Nem-determinizmus

Hajnal Péter

Bolyai Intézet, TTIK, SZTE, Szeged

2020. ősz

Számolás vs érvelés

Számolás vs érvelés

Definiáltuk egy számítás bonyolultságát.

Számolás vs érvelés

Definiáltuk egy számítás bonyolultságát. Például egy L döntési feladat akkor tartozott a $\mathcal{P}$ nyelvosztályhoz, ha létezett egy ezt eldöntő Turing-gép egy garanciával, hogy tetszőleges ω inputon az output/döntés $|\omega|$-nak polinomjaként függő lépésszámon belül megtörténik.

Számolás vs érvelés

Definiáltuk egy számítás bonyolultságát. Például egy L döntési feladat akkor tartozott a $\mathcal{P}$ nyelvosztályhoz, ha létezett egy ezt eldöntő Turing-gép egy garanciával, hogy tetszőleges ω inputon az output/döntés $|\omega|$-nak polinomjaként függő lépésszámon belül megtörténik.

Néha azonban nem akarjuk a teljes számolást elvégezni.

Számolás vs érvelés

Definiáltuk egy számítás bonyolultságát. Például egy L döntési feladat akkor tartozott a $\mathcal{P}$ nyelvosztályhoz, ha létezett egy ezt eldöntő Turing-gép egy garanciával, hogy tetszőleges ω inputon az output/döntés $|\omega|$-nak polinomjaként függő lépésszámon belül megtörténik.

Néha azonban nem akarjuk a teljes számolást elvégezni. Beérjük azzal, hogy valahogy demonstrálja/láttassa/bizonyítsa a gép, hogy $\omega \in L$ (feltéve, hogy ez így van).

Számolás vs érvelés

Definiáltuk egy számítás bonyolultságát. Például egy L döntési feladat akkor tartozott a $\mathcal{P}$ nyelvosztályhoz, ha létezett egy ezt eldöntő Turing-gép egy garanciával, hogy tetszőleges ω inputon az output/döntés $|\omega|$-nak polinomjaként függő lépésszámon belül megtörténik.

Néha azonban nem akarjuk a teljes számolást elvégezni. Beérjük azzal, hogy valahogy demonstrálja/láttassa/bizonyítsa a gép, hogy $\omega \in L$ (feltéve, hogy ez így van).

Az eddig tárgyalt kiszámíthatósági fogalom olyan volt, hogy ismert input esetén egyértelmű volt, milyen konfigurációsorozatot követve fut a gép.

Számolás vs érvelés

Definiáltuk egy számítás bonyolultságát. Például egy L döntési feladat akkor tartozott a $\mathcal{P}$ nyelvosztályhoz, ha létezett egy ezt eldöntő Turing-gép egy garanciával, hogy tetszőleges ω inputon az output/döntés $|\omega|$-nak polinomjaként függő lépésszámon belül megtörténik.

Néha azonban nem akarjuk a teljes számolást elvégezni. Beérjük azzal, hogy valahogy demonstrálja/láttassa/bizonyítsa a gép, hogy $\omega \in L$ (feltéve, hogy ez így van).

Az eddig tárgyalt kiszámíthatósági fogalom olyan volt, hogy ismert input esetén egyértelmű volt, milyen konfigurációsorozatot követve fut a gép.

Az emberi agy nem ilyen (gondoljuk mi). A gondolkozás/érvelés nem így számol.

Determinizmus vs nem-determinizmus

Determinizmus vs nem-determinizmus

Az eredeti kiszámíthatósághoz egy jelzőt teszünk: a definiált Turing-gép egy determinisztikus gép.

Determinizmus vs nem-determinizmus

Az eredeti kiszámíthatósághoz egy jelzőt teszünk: a definiált Turing-gép egy determinisztikus gép.

Vannak nem-deterministikus gépek is. Az alábbiakban a nem-deterministikus Turing-gépekre két alternatív definíciót is adunk.

Nem-determinizmus: I. változat

Nem-determinizmus: I. változat

Hasonlóan mint a determinisztikus Turing-gépeknél, itt is vannak szalagok, fejek, állapotok, stb. Mi most csak az egy munkaszalagos változatot írjuk le.

Nem-determinizmus: I. változat

Hasonlóan mint a determinisztikus Turing-gépeknél, itt is vannak szalagok, fejek, állapotok, stb. Mi most csak az egy munkaszalagos változatot írjuk le.

Definíció: Nem-determinisztikus TG

A nem-determinisztikus TG átmeneti függvénye:

$$\delta : \Sigma \times \Gamma \times S \to \mathcal{P}(\{\leftarrow, ., \rightarrow\} \times \Gamma \times \{\leftarrow, ., \rightarrow\} \times S) \setminus \{\emptyset\}.$$

Nem-determinizmus: I. változat

Nem-determinizmus: I. változat

Azaz a konfiguráció látott része függvényében nem egy update-szabály adott, hanem update-szabályok egy nem-üres halmaza.

Nem-determinizmus: I. változat

Azaz a konfiguráció látott része függvényében nem egy update-szabály adott, hanem update-szabályok egy nem-üres halmaza.

Adott konfigurációra nem szükségszerűen egy rákövetkező konfiguráció adható meg, hanem rákövetkező konfigurációk egy halmaza (az átmenetifüggvény által leírt halmaz mindegyik eleme egy-egy lehetséges rákövetkező konfigurációt ad).

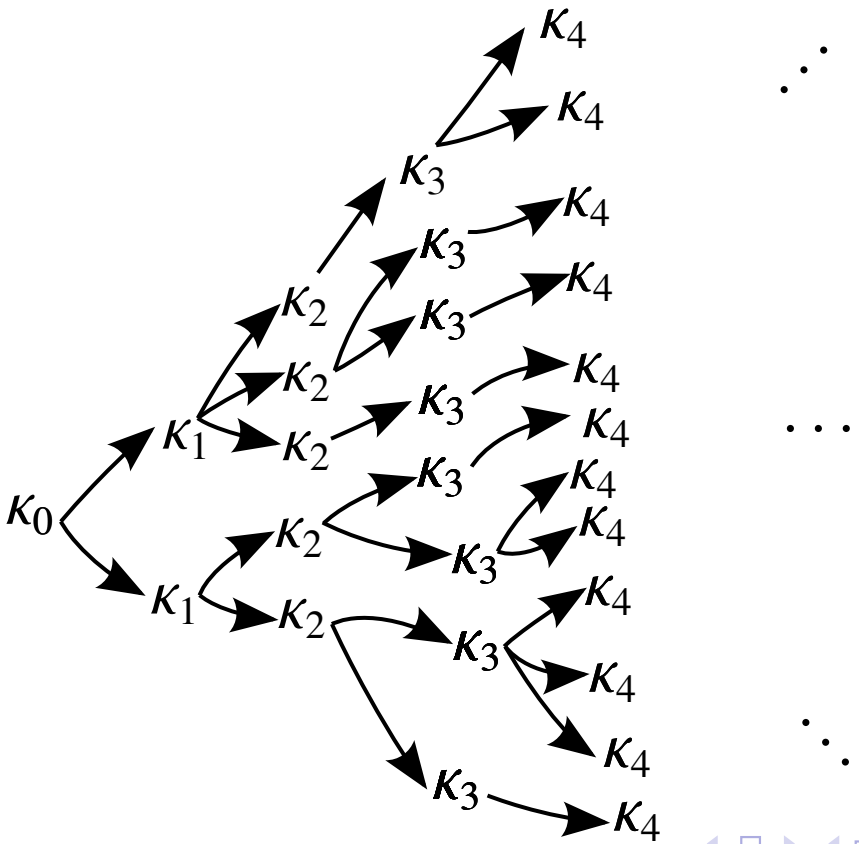
Nem-determinizmus: I. változat

Nem-determinizmus: I. változat

Így az ω inputhoz tartozó futás nem meghatározott (idegen szóval nem-determinisztikus), azaz a $\kappa_0(\omega)$ kezdőkonfigurációból több lehetséges konfiguráció felé mehetünk. Így egy $\kappa_0(\omega)$-ban gyökereztetett fa írja le a gép lehetséges futásait.

Nem-determinizmus: I. változat

Így az ω inputhoz tartozó futás nem meghatározott (idegen szóval nem-determinisztikus), azaz a $\kappa_0(\omega)$ kezdőkonfigurációból több lehetséges konfiguráció felé mehetünk. Így egy $\kappa_0(\omega)$-ban gyökereztetett fa írja le a gép lehetséges futásait.



Nem-determinizmus: I. változat

Nem-determinizmus: I. változat

A nem-determinizmus megértéséhez nagyon fontos, hogy tisztázzuk mikor is számít ki egy gép egy nyelvet.

Nem-determinizmus: I. változat

A nem-determinizmus megértéséhez nagyon fontos, hogy tisztázzuk mikor is számít ki egy gép egy nyelvet.

Definíció

Ahhoz, hogy egy nem-determinisztikus gép egy nyelvet elfogadjon minden inputon minden futásának le kell állnia.

Nem-determinizmus: I. változat

A nem-determinizmus megértéséhez nagyon fontos, hogy tisztázzuk mikor is számít ki egy gép egy nyelvet.

Definíció

Ahhoz, hogy egy nem-determinisztikus gép egy nyelvet elfogadjon minden inputon minden futásának le kell állnia.

Az ω inputot pontosan akkor fogadja el a T nem-determinisztikus Turing-gép, ha létezik ELFOGAD állapotba vezető futása.

Nem-determinizmus: I. változat

A nem-determinizmus megértéséhez nagyon fontos, hogy tisztázzuk mikor is számít ki egy gép egy nyelvet.

Definíció

Ahhoz, hogy egy nem-determinisztikus gép egy nyelvet elfogadjon minden inputon minden futásának le kell állnia.

Az ω inputot pontosan akkor fogadja el a T nem-determinisztikus Turing-gép, ha létezik ELFOGAD állapotba vezető futása.

Azaz ω elvetése ekvivalens azzal, hogy ω-n minden futása ELVET állapothoz vezet.

Nem-determinizmus: II. változat

Definíció

Ebben az esetben egy plusz szalagunk lesz az input- és munkaszalagok között, az úgynevezett tanú/bizonyítás szalag.

Nem-determinizmus: II. változat

Definíció

Ebben az esetben egy plusz szalagunk lesz az input- és munkaszalagok között, az úgynevezett tanú/bizonyítás szalag.

Ez a szalag csak olvasható és a fej csak jobbra tud mozogni rajta.

Nem-determinizmus: II. változat

Definíció

Ebben az esetben egy plusz szalagunk lesz az input- és munkaszalagok között, az úgynevezett tanú/bizonyítás szalag.

Ez a szalag csak olvasható és a fej csak jobbra tud mozogni rajta.

Az átmeneti függvényt ugyanúgy definiáljuk, mint a determinisztikus esetben, és a futás is determinisztikus lesz, azaz ω és τ (a tanúszalag tartalma) egyértelműen meghatároz egy konfigurációsorozatot:

$$\kappa_0 = \kappa_0(\omega, \tau) \to \kappa_1 \to \kappa_2 \to \dots.$$

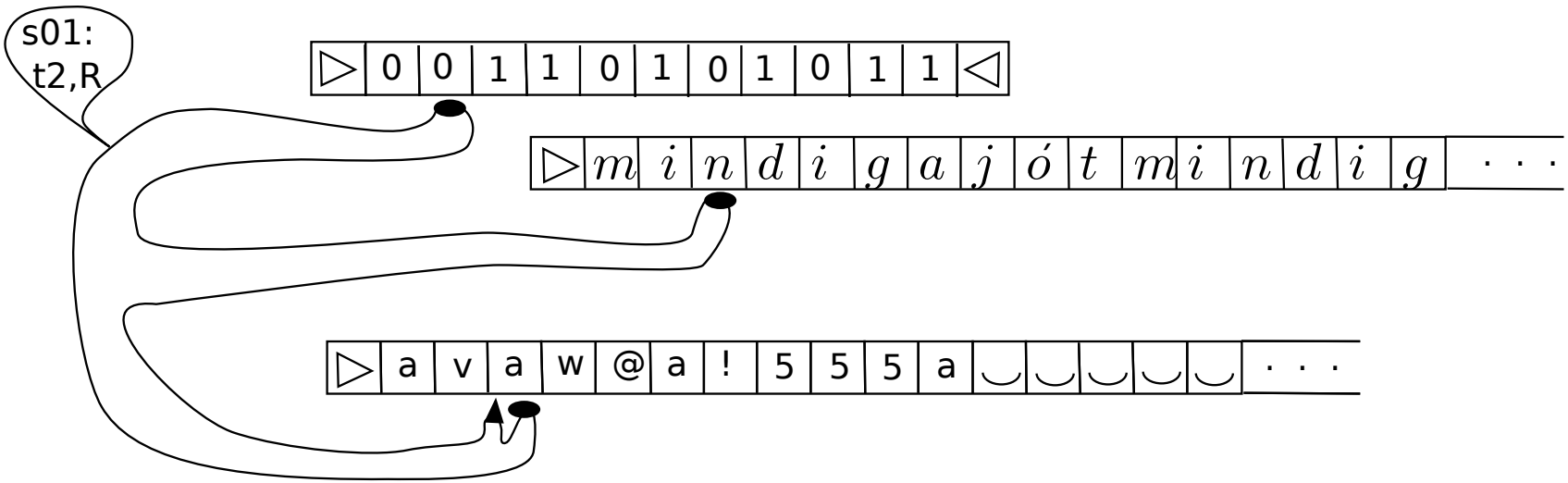
Nem-determinizmus: II. változat képen

Nem-determinizmus: II. változat képen

Az alábbi ábra a II. nem-determinisztikus gép egy konfigurációjának „fényképe”.

Nem-determinizmus: II. változat képen

Az alábbi ábra a II. nem-determinisztikus gép egy konfigurációjának „fényképe”.



Nem-determinizmus: II. változat

Nem-determinizmus: II. változat

A probléma ismét az, hogy mikor mondjuk, hogy egy nem-determisztikus gép elfogad egy L nyelvet.

Nem-determinizmus: II. változat

A probléma ismét az, hogy mikor mondjuk, hogy egy nem-determisztikus gép elfogad egy L nyelvet. Ehhez szükséges, hogy a gép minden ω inputon tetszőleges tanúszalag-tartalom mellett leálljon.

Nem-determinizmus: II. változat

A probléma ismét az, hogy mikor mondjuk, hogy egy nem-determisztikus gép elfogad egy L nyelvet. Ehhez szükséges, hogy a gép minden ω inputon tetszőleges tanúszalag-tartalom mellett leálljon. Az ω inputot pontosan akkor fogadja el egy nem-determinisztikus Turing-gép, ha van olyan τ tanúszalag tartalom, amelyre a futás ELFOGAD állapotba kerül.

Nem-determinizmus: II. változat

A probléma ismét az, hogy mikor mondjuk, hogy egy nem-determisztikus gép elfogad egy L nyelvet. Ehhez szükséges, hogy a gép minden ω inputon tetszőleges tanúszalag-tartalom mellett leálljon. Az ω inputot pontosan akkor fogadja el egy nem-determinisztikus Turing-gép, ha van olyan τ tanúszalag tartalom, amelyre a futás ELFOGAD állapotba kerül.

Egy τ tanú során az ω inputon ELVET állapotba juthatunk. Ez nem jelenti szükségszerűen, hogy az input rossz.

Nem-determinizmus: II. változat

A probléma ismét az, hogy mikor mondjuk, hogy egy nem-determisztikus gép elfogad egy L nyelvet. Ehhez szükséges, hogy a gép minden ω inputon tetszőleges tanúszalag-tartalom mellett leálljon. Az ω inputot pontosan akkor fogadja el egy nem-determinisztikus Turing-gép, ha van olyan τ tanúszalag tartalom, amelyre a futás ELFOGAD állapotba kerül.

Egy τ tanú során az ω inputon ELVET állapotba juthatunk. Ez nem jelenti szükségszerűen, hogy az input rossz. Ha $\omega \in L$, akkor a jelentése, hogy τ egy rossz választás/nem meggyőző tanú. Emiatt gyakran a NEM-STIMMEL nevet adjuk az ELVET állapotnak a nem-determinisztikus esetben.

Nem-determinizmus: II. változat

A probléma ismét az, hogy mikor mondjuk, hogy egy nem-determisztikus gép elfogad egy L nyelvet. Ehhez szükséges, hogy a gép minden ω inputon tetszőleges tanúszalag-tartalom mellett leálljon. Az ω inputot pontosan akkor fogadja el egy nem-determinisztikus Turing-gép, ha van olyan τ tanúszalag tartalom, amelyre a futás ELFOGAD állapotba kerül.

Egy τ tanú során az ω inputon ELVET állapotba juthatunk. Ez nem jelenti szükségszerűen, hogy az input rossz. Ha $\omega \in L$, akkor a jelentése, hogy τ egy rossz választás/nem meggyőző tanú. Emiatt gyakran a NEM-STIMMEL nevet adjuk az ELVET állapotnak a nem-determinisztikus esetben. Az elvetés akkor történik, ha minden τ tanúszalag-tartalom NEM-STIMMEL állapothoz vezet.

Nem-determinizmus: A két változat viszonya

Nem-determinizmus: A két változat viszonya

A két változatban egy lényeges különbség, hogy az elsőben a nem-determinizmus a futás során „szétszórt”, az utolsó lépés előtt sem meghatározott a végső állapot.

Nem-determinizmus: A két változat viszonya

A két változatban egy lényeges különbség, hogy az elsőben a nem-determinizmus a futás során „szétszórt”, az utolsó lépés előtt sem meghatározott a végső állapot.

A második változatban a nem-determinizmus a τ választásával jelentkezik. Azaz mindenek előtt lerögzítjük esetleges döntéseinket, alternatíváinkat. A futás ezekután determinisztikus lesz.

Nem-determinizmus: A két változat viszonya

A két változatban egy lényeges különbség, hogy az elsőben a nem-determinizmus a futás során „szétszórt", az utolsó lépés előtt sem meghatározott a végső állapot.

A második változatban a nem-determinizmus a τ választásával jelentkezik. Azaz mindenek előtt lerögzítjük esetleges döntéseinket, alternatíváinkat. A futás ezekután determinisztikus lesz.

Tétel

L pontosan akkor eldönthető I-nem-determinisztikus TG-gel, ha eldönthető II-nem-determinisztikus TG-gel.

Nem-determinizmus: A két változat viszonya

A két változatban egy lényeges különbség, hogy az elsőben a nem-determinizmus a futás során „szétszórt", az utolsó lépés előtt sem meghatározott a végső állapot.

A második változatban a nem-determinizmus a τ választásával jelentkezik. Azaz mindenek előtt lerögzítjük esetleges döntéseinket, alternatíváinkat. A futás ezekután determinisztikus lesz.

Tétel

L pontosan akkor eldönthető I-nem-determinisztikus TG-gel, ha eldönthető II-nem-determinisztikus TG-gel.

A tételt nem bizonyítjuk, de az érdeklődő hallgató könnyen megteheti ezt.

A nem-determinizmus ereje

A nem-determinizmus ereje

Felmerül a kérdés, hogy a nem-determinizmus ad-e plusz kiszámíthatósági erőt. Nem.

A nem-determinizmus ereje

Felmerül a kérdés, hogy a nem-determinizmus ad-e plusz kiszámíthatósági erőt. Nem.

Tétel

Egy L nyelvhez akkor és csak akkor van olyan nem-determinisztikus Turing-gép, amely L-et elfogadja el, ha L eldönthető.

A nem-determinizmus ereje

Felmerül a kérdés, hogy a nem-determinizmus ad-e plusz kiszámíthatósági erőt. Nem.

Tétel

Egy L nyelvhez akkor és csak akkor van olyan nem-determinisztikus Turing-gép, amely L-et elfogadja el, ha L eldönthető.

Az egyik irány nyilvánvaló:

A nem-determinizmus ereje

Felmerül a kérdés, hogy a nem-determinizmus ad-e plusz kiszámíthatósági erőt. Nem.

Tétel

Egy L nyelvhez akkor és csak akkor van olyan nem-determinisztikus Turing-gép, amely L-et elfogadja el, ha L eldönthető.

Az egyik irány nyilvánvaló: Ha L eldönthető, akkor van olyan T (determinisztikus) Turing-gép, amely L-et fogadja el. T felfogható egy speciális nem-determinisztikus gépnek, amely ugyancsak L-et fogadja el.

A nem-determinizmus ereje

Felmerül a kérdés, hogy a nem-determinizmus ad-e plusz kiszámíthatósági erőt. Nem.

Tétel

Egy L nyelvhez akkor és csak akkor van olyan nem-determinisztikus Turing-gép, amely L-et elfogadja el, ha L eldönthető.

Az egyik irány nyilvánvaló: Ha L eldönthető, akkor van olyan T (determinisztikus) Turing-gép, amely L-et fogadja el. T felfogható egy speciális nem-determinisztikus gépnek, amely ugyancsak L-et fogadja el.

Csak a fordított irányt kell belátni.

Bizonyítás

Bizonyítás

Tegyük fel, hogy van olyan T nem-determinisztikus (II. értelemben) Turing-gép, amely L-et fogadja el.

Bizonyítás

Tegyük fel, hogy van olyan T nem-determinisztikus (II. értelemben) Turing-gép, amely L-et fogadja el. Ekkor konstruálunk egy determinisztikus $\widetilde{T}$ gépet, amely szintén L-et fogadja el:

Bizonyítás

Tegyük fel, hogy van olyan T nem-determinisztikus (II. értelemben) Turing-gép, amely L-et fogadja el. Ekkor konstruálunk egy determinisztikus $\widetilde{T}$ gépet, amely szintén L-et fogadja el:

Feltesszük, hogy T-nek egy munkaszalagja volt.

Bizonyítás

Tegyük fel, hogy van olyan T nem-determinisztikus (II. értelemben) Turing-gép, amely L-et fogadja el. Ekkor konstruálunk egy determinisztikus $\widetilde{T}$ gépet, amely szintén L-et fogadja el:

Feltesszük, hogy T-nek egy munkaszalagja volt. $\widetilde{T}$ gépnek három lesz, amelyből az első a tanúszalagot „szimulálja”.

Bizonyítás

Tegyük fel, hogy van olyan T nem-determinisztikus (II. értelemben) Turing-gép, amely L-et fogadja el. Ekkor konstruálunk egy determinisztikus $\widetilde{T}$ gépet, amely szintén L-et fogadja el:

Feltesszük, hogy T-nek egy munkaszalagja volt. $\widetilde{T}$ gépnek három lesz, amelyből az első a tanúszalagot „szimulálja”. A második a hátralévő tanú kezdőszeleteket tartalmazza.

Bizonyítás

Tegyük fel, hogy van olyan T nem-determinisztikus (II. értelemben) Turing-gép, amely L-et fogadja el. Ekkor konstruálunk egy determinisztikus $\widetilde{T}$ gépet, amely szintén L-et fogadja el:

Feltesszük, hogy T-nek egy munkaszalagja volt. $\widetilde{T}$ gépnek három lesz, amelyből az első a tanúszalagot „szimulálja”. A második a hátralévő tanú kezdőszeleteket tartalmazza. A harmadik szalag a T gép szimulálásához szükséges munkaterületet adja.

Bizonyítás

Tegyük fel, hogy van olyan T nem-determinisztikus (II. értelemben) Turing-gép, amely L-et fogadja el. Ekkor konstruálunk egy determinisztikus $\widetilde{T}$ gépet, amely szintén L-et fogadja el:

Feltesszük, hogy T-nek egy munkaszalagja volt. $\widetilde{T}$ gépnek három lesz, amelyből az első a tanúszalagot „szimulálja”. A második a hátralévő tanú kezdőszeleteket tartalmazza. A harmadik szalag a T gép szimulálásához szükséges munkaterületet adja.

Kezdetben a második szalag csak az üres tanú töredéket tartalmazza.

Bizonyítás

Tegyük fel, hogy van olyan T nem-determinisztikus (II. értelemben) Turing-gép, amely L-et fogadja el. Ekkor konstruálunk egy determinisztikus $\widetilde{T}$ gépet, amely szintén L-et fogadja el:

Feltesszük, hogy T-nek egy munkaszalagja volt. $\widetilde{T}$ gépnek három lesz, amelyből az első a tanúszalagot „szimulálja”. A második a hátralévő tanú kezdőszeleteket tartalmazza. A harmadik szalag a T gép szimulálásához szükséges munkaterületet adja.

Kezdetben a második szalag csak az üres tanú töredéket tartalmazza. Később tanú kezdőszeletek egy sorozatát tartalmazza, amelyeket eddig nem tesztelt.

Bizonyítás

Tegyük fel, hogy van olyan T nem-determinisztikus (II. értelemben) Turing-gép, amely L-et fogadja el. Ekkor konstruálunk egy determinisztikus $\widetilde{T}$ gépet, amely szintén L-et fogadja el:

Feltesszük, hogy T-nek egy munkaszalagja volt. $\widetilde{T}$ gépnek három lesz, amelyből az első a tanúszalagot „szimulálja". A második a hátralévő tanú kezdőszeleteket tartalmazza. A harmadik szalag a T gép szimulálásához szükséges munkaterületet adja.

Kezdetben a második szalag csak az üres tanú töredéket tartalmazza. Később tanú kezdőszeletek egy sorozatát tartalmazza, amelyeket eddig nem tesztelt.

A START állapot után $\widetilde{T}$ a második szalag első tanú kezdőszeletét az első szalagra másolja, majd a "töredék-vége" jelet írja mögé, közben a második szalagról letörli és a "TANÚ-TÖREDÉK-JELEN" állapotba kerül.

Bizonyítás (folytatás)

Bizonyítás (folytatás)

A $\widetilde{T}$ gép T-t szimulálja, de a tanúszalag tartalmat az első munkaszalagról veszi, munkáját a harmadik szalagon végzi. T "START$_T$" állapotát a "TANÚ-TÖREDÉK-JELEN" állapot veszi át.

Bizonyítás (folytatás)

A $\widetilde{T}$ gép T-t szimulálja, de a tanúszalag tartalmat az első munkaszalagról veszi, munkáját a harmadik szalagon végzi. T "START$_T$" állapotát a "TANÚ-TÖREDÉK-JELEN" állapot veszi át.

A szimulálás végeredményére három lehetőség van:

(1) A szimulálás során elolvassuk a "töredék-vége" jelet.

(2) A szimulálás során NEM-STIMMEL$_T$ állapotba jutunk.

(3) A szimulálás során ELFOGAD$_T$ állapotba jutunk.

Bizonyítás: (1) eset

Bizonyítás: (1) eset

(1) az aktuális tanú τ_t töredék nem elég hosszú, a szimulálás "túlcsordul".

Bizonyítás: (1) eset

(1) az aktuális tanú τ_t töredék nem elég hosszú, a szimulálás "túlcsordul".

τ_t összes egy karakterrel való meghosszabbítását vesszük és a második szalagra az aktuális kezdőszeletek listájára felvesszük, majd "KÉREM-A-KÖVETKEZŐT" állapotba kerül.

Bizonyítás: (1) eset

(1) az aktuális tanú τ_t töredék nem elég hosszú, a szimulálás "túlcsordul".

τ_t összes egy karakterrel való meghosszabbítását vesszük és a második szalagra az aktuális kezdőszeletek listájára felvesszük, majd "KÉREM-A-KÖVETKEZŐT" állapotba kerül.

$\widetilde{T}$ ismét a második szalag első tanú kezdőszeletét az első szalagra másolja, majd a "töredék-vége" jelet írja mögé, közben a második szalagról letörli és a "TANÚ-TÖREDÉK-JELEN" állapotba kerül.

Bizonyítás: (2) eset és (3) eset

Bizonyítás: (2) eset és (3) eset

(2) esetén a tanú töredéket végigteszteltük.

Bizonyítás: (2) eset és (3) eset

(2) esetén a tanú töredéket végigteszteltük. A második szalagot nem írjuk felül, hanem az első ott lévő tanú kezdőszelettel folytatjuk a futást:

Bizonyítás: (2) eset és (3) eset

(2) esetén a tanú töredéket végigteszteltük. A második szalagot nem írjuk felül, hanem az első ott lévő tanú kezdőszelettel folytatjuk a futást: Átmásoljuk az első szalagra (ezzel eltűnik a második szalagról) és a szimulálás újra elkezdődik.

Bizonyítás: (2) eset és (3) eset

(2) esetén a tanú töredéket végigteszteltük. A második szalagot nem írjuk felül, hanem az első ott lévő tanú kezdőszelettel folytatjuk a futást: Átmásoljuk az első szalagra (ezzel eltűnik a második szalagról) és a szimulálás újra elkezdődik.

(2) esetén előfordulhat, hogy a második szalag tartalma kiürül.

Bizonyítás: (2) eset és (3) eset

(2) esetén a tanú töredéket végigteszteltük. A második szalagot nem írjuk felül, hanem az első ott lévő tanú kezdőszelettel folytatjuk a futást: Átmásoljuk az első szalagra (ezzel eltűnik a második szalagról) és a szimulálás újra elkezdődik.

(2) esetén előfordulhat, hogy a második szalag tartalma kiürül. Ekkor $\widetilde{T}$ leáll ELVET állapottal.

Bizonyítás: (2) eset és (3) eset

(2) esetén a tanú töredéket végigteszteltük. A második szalagot nem írjuk felül, hanem az első ott lévő tanú kezdőszelettel folytatjuk a futást: Átmásoljuk az első szalagra (ezzel eltűnik a második szalagról) és a szimulálás újra elkezdődik.

(2) esetén előfordulhat, hogy a második szalag tartalma kiürül. Ekkor $\widetilde{T}$ leáll ELVET állapottal.

(3) egyszerű: Ekkor ELFOGAD állapotba jutunk.

Szünet

Nem-determinisztikus számításon alapuló nyelvosztályok

Nem-determinisztikus számításon alapuló nyelvosztályok

A nem-determinizmus kétféle változata közül bármelyikre alapozhatjuk definícióinkat (úgy, hogy ugyanahhoz a nyelvosztályokhoz jussunk). Mi a második (tanúszalagos) szemléletet követjük.

Nem-determinisztikus számításon alapuló nyelvosztályok

A nem-determinizmus kétféle változata közül bármelyikre alapozhatjuk definícióinkat (úgy, hogy ugyanahhoz a nyelvosztályokhoz jussunk). Mi a második (tanúszalagos) szemléletet követjük.

Definíció

Legyen T egy II. értelemben vett nem-determinisztikus Turing-gép. Ekkor $TIME(\omega, \tau; T)$ és $SPACE(\omega, \tau; T)$ a determinisztikus eset lemásolásával definiálható.
Legyen

$$NTIME(\omega; T) = \begin{cases} \min\{ TIME(\omega, \tau; T) : \text{ahol } \tau \text{ olyan,} \\ \qquad \text{hogy } \omega\text{-n } T \text{ ELFOGAD állapotba jut}\}, & \omega \in L, \\ \min\{ TIME(\omega, \tau; T) : \text{ahol } \tau \in \Sigma^*\}, & \omega \notin L. \end{cases}$$

Nem-determinisztikus számításon alapuló nyelvosztályok

Nem-determinisztikus számításon alapuló nyelvosztályok

Azaz az elfogadó futások közül a „legzseniálisabb” tanúszalag-tartalom szabja meg az idő korlátot.

Nem-determinisztikus számításon alapuló nyelvosztályok

Azaz az elfogadó futások közül a „legzseniálisabb” tanúszalag-tartalom szabja meg az idő korlátot.

Definíció

$$NSPACE(\omega; T) = \begin{cases} \min\{SPACE(\omega, \tau; T) : \text{ahol } \tau \text{ olyan, hogy } \omega\text{-n} \\ \qquad T \text{ ELFOGAD állapotba jut}\}, & \omega \in L, \\ \min\{SPACE(\omega, \tau; T) : \text{ ahol } \tau \in \Sigma^*\}, & \omega \notin L. \end{cases}$$

A leggyakoribb osztályok

Definíció

A leggyakoribb osztályok

Definíció

$$\mathcal{NP} = \{L : \text{létezik olyan } T \text{ nem-determinisztikus Turing-gép, ami } L\text{-et fogadja el, továbbá létezik olyan } i \in \mathbb{N}, \text{ hogy minden } \omega\text{-ra } NTIME(\omega; T) \leq |\omega|^i + i.\}$$

A leggyakoribb osztályok

Definíció

$$\mathcal{NP} = \{L : \text{létezik olyan } T \text{ nem-determinisztikus Turing-gép, ami } L\text{-et fogadja el, továbbá létezik olyan } i \in \mathbb{N}, \text{ hogy minden } \omega\text{-ra } NTIME(\omega; T) \leq |\omega|^i + i.\}$$

$$\mathcal{NEXP} = \{L : \text{létezik olyan } T \text{ nem-determinisztikus Turing-gép, ami } L\text{-et fogadja el, továbbá létezik olyan } i \in \mathbb{N}, \text{ hogy minden } \omega\text{-ra } NTIME(\omega; T) \leq 2^{|\omega|^i + i}.\}$$

A leggyakoribb osztályok

Definíció

A leggyakoribb osztályok

Definíció

$$\mathcal{NL} = \{L : \text{létezik olyan } T \text{ nem-determinisztikus Turing-gép, ami } L\text{-et fogadja el, továbbá létezik olyan } i \in \mathbb{N}, \text{ hogy minden } \omega\text{-ra } \mathit{NSPACE}(\omega; T) \leq i \log(|\omega| + 1).\}$$

A leggyakoribb osztályok

Definíció

$$\mathcal{NL} = \{L : \text{létezik olyan } T \text{ nem-determinisztikus Turing-gép, ami } L\text{-et fogadja el, továbbá létezik olyan } i \in \mathbb{N}, \text{ hogy minden } \omega\text{-ra } NSPACE(\omega; T) \leq i \log(|\omega| + 1).\}$$

$$\mathcal{NPSPACE} = \{L : \text{létezik olyan } T \text{ nem-determinisztikus Turing-gép, ami } L\text{-et fogadja el, továbbá létezik olyan } i \in \mathbb{N}, \text{ hogy minden } \omega\text{-ra } NSPACE(\omega; T) \leq |\omega|^i + i.\}$$

A keggyakoribb osztályok

Definíció

A keggyakoribb osztályok

Definíció

$$\mathcal{NEXPSPACE} = \{L : \text{létezik olyan } T \text{ nem-determinisztikus Turing-gép, ami } L\text{-et fogadja el, továbbá létezik olyan } i \in \mathbb{N}, \text{ hogy minden } \omega\text{-ra } NSPACE(\omega; T) \leq 2^{|\omega|^i + i}.\}$$

A keggyakoribb osztályok

Definíció

$$\mathcal{NEXPSPACE} = \{L : \text{létezik olyan } T \text{ nem-determinisztikus Turing-gép, ami } L\text{-et fogadja el, továbbá létezik olyan } i \in \mathbb{N}, \text{ hogy minden } \omega\text{-ra } NSPACE(\omega; T) \leq 2^{|\omega|^i + i}.\}$$

Ismét megjegyezzük, hogy definiált osztályok robusztusak:

A keggyakoribb osztályok

Definíció

$$\mathcal{NEXPSPACE} = \{L : \text{létezik olyan } T \text{ nem-determinisztikus Turing-gép, ami } L\text{-et fogadja el, továbbá létezik olyan } i \in \mathbb{N}, \text{ hogy minden } \omega\text{-ra } NSPACE(\omega; T) \leq 2^{|\omega|^i + i}.\}$$

Ismét megjegyezzük, hogy definiált osztályok robusztusak: ha a Turing-gép definícióját kissé megváltoztatjuk, akkor a megfelelő osztályok ugyanazok maradnak.

A keggyakoribb osztályok

Definíció

$$\mathcal{NEXPSPACE} = \{L : \text{létezik olyan } T \text{ nem-determinisztikus Turing-gép, ami } L\text{-et fogadja el, továbbá létezik olyan } i \in \mathbb{N}, \text{ hogy minden } \omega\text{-ra } NSPACE(\omega; T) \leq 2^{|\omega|^i + i}.\}$$

Ismét megjegyezzük, hogy definiált osztályok robusztusak: ha a Turing-gép definícióját kissé megváltoztatjuk, akkor a megfelelő osztályok ugyanazok maradnak.

A fenti definíciókat a nem-determinizmus I. változatára alapítva is bevezethettük volna.

Komplementálás és determinizmus

Komplementálás és determinizmus

Észrevétel

A determinisztikus osztályok zártak a komplementálásra.

Komplementálás és determinizmus

Észrevétel

A determinisztikus osztályok zártak a komplementálásra.

Példa

Ha $L \in_T \mathcal{P}$, akkor $\overline{L} = \Sigma^* - L$ is $\mathcal{P}$-hez tartozik.

Komplementálás és determinizmus

Észrevétel

A determinisztikus osztályok zártak a komplementálásra.

Példa

Ha $L \in_T \mathcal{P}$, akkor $\overline{L} = \Sigma^* - L$ is $\mathcal{P}$-hez tartozik.

Az állítás bizonyításához legyen $\widetilde{T}$ az a Turing-gép, amit T-ből az alábbi egyszerű változtatással kapunk: Az átmeneti függvényt úgy írjuk át, hogy ha T-nél az ELFOGAD állapotba vezet, akkor $\widetilde{T}$ (minden más megtartásával) az ELVET állapotba jusson. Illetve fordítva.

Komplementálás és determinizmus

Észrevétel

A determinisztikus osztályok zártak a komplementálásra.

Példa

Ha $L \in_T \mathcal{P}$, akkor $\overline{L} = \Sigma^* - L$ is $\mathcal{P}$-hez tartozik.

Az állítás bizonyításához legyen $\widetilde{T}$ az a Turing-gép, amit T-ből az alábbi egyszerű változtatással kapunk: Az átmeneti függvényt úgy írjuk át, hogy ha T-nél az ELFOGAD állapotba vezet, akkor $\widetilde{T}$ (minden más megtartásával) az ELVET állapotba jusson. Illetve fordítva.

Ezzel $\widetilde{T}$ pontosan a T által elvetett inputokat fogadja el. Azaz a kiszámított nyelv a komplementer nyelv lesz.

Komplementálás és determinizmus

Észrevétel

A determinisztikus osztályok zártak a komplementálásra.

Példa

Ha $L \in_T \mathcal{P}$, akkor $\overline{L} = \Sigma^* - L$ is $\mathcal{P}$-hez tartozik.

Az állítás bizonyításához legyen $\widetilde{T}$ az a Turing-gép, amit T-ből az alábbi egyszerű változtatással kapunk: Az átmeneti függvényt úgy írjuk át, hogy ha T-nél az ELFOGAD állapotba vezet, akkor $\widetilde{T}$ (minden más megtartásával) az ELVET állapotba jusson. Illetve fordítva.

Ezzel $\widetilde{T}$ pontosan a T által elvetett inputokat fogadja el. Azaz a kiszámított nyelv a komplementer nyelv lesz.

T és $\widetilde{T}$ bonyolultsága ugyanaz lesz.

Komplementálás és nem-determinizmus

Komplementálás és nem-determinizmus

A fenti észrevétel a nem-determinisztikus esetben távolról sem nyilvánvaló, ha egyáltalán igaz.

Komplementálás és nem-determinizmus

A fenti észrevétel a nem-determinisztikus esetben távolról sem nyilvánvaló, ha egyáltalán igaz.

A következő definíciók jogosak.

Komplementálás és nem-determinizmus

A fenti észrevétel a nem-determinisztikus esetben távolról sem nyilvánvaló, ha egyáltalán igaz.

A következő definíciók jogosak.

Definíció

Komplementálás és nem-determinizmus

A fenti észrevétel a nem-determinisztikus esetben távolról sem nyilvánvaló, ha egyáltalán igaz.

A következő definíciók jogosak.

Definíció

$$co\ \mathcal{NP} = \{\overline{L} : L \in \mathcal{NP}\},$$
$$co\ \mathcal{NEXP} = \{\overline{L} : L \in \mathcal{NEXP}\},$$
$$co\ \mathcal{NL} = \{\overline{L} : L \in \mathcal{NL}\},$$
$$co\ \mathcal{NPSPACE} = \{\overline{L} : L \in \mathcal{NPSPACE}\},$$
$$co\ \mathcal{NEXPSPACE} = \{\overline{L} : L \in \mathcal{NEXPSPACE}\},$$

Nyilvánvaló tartalmazások

Nyilvánvaló tartalmazások

Több idő, több nyelv.

Nyilvánvaló tartalmazások

Több idő, több nyelv. Több tár, több nyelv.

Nyilvánvaló tartalmazások

Több idő, több nyelv. Több tár, több nyelv.

A nem determinizmus ereje több nyelv.

Nyilvánvaló tartalmazások

Több idő, több nyelv. Több tár, több nyelv.

A nem determinizmus ereje több nyelv.

Ezen állítások nyilvánvalóak a definíciókból (amennyiben a „több” azt jelenti, hogy legalább annyi).

Nyilvánvaló tartalmazások

Több idő, több nyelv. Több tár, több nyelv.

A nem determinizmus ereje több nyelv.

Ezen állítások nyilvánvalóak a definíciókból (amennyiben a „több” azt jelenti, hogy legalább annyi).

Az is természetes, hogy korlátozott idő korlátozott tárfelhasználást is jelent.

Nyilvánvaló tartalmazások

Több idő, több nyelv. Több tár, több nyelv.

A nem determinizmus ereje több nyelv.

Ezen állítások nyilvánvalóak a definíciókból (amennyiben a „több” azt jelenti, hogy legalább annyi).

Az is természetes, hogy korlátozott idő korlátozott tárfelhasználást is jelent.

Ezek alapján az eddigi nyelvosztályokról a következő tartalmazások nyilvánvalók:

Nyilvánvaló tartalmazások

Több idő, több nyelv. Több tár, több nyelv.

A nem determinizmus ereje több nyelv.

Ezen állítások nyilvánvalóak a definíciókból (amennyiben a „több” azt jelenti, hogy legalább annyi).

Az is természetes, hogy korlátozott idő korlátozott tárfelhasználást is jelent.

Ezek alapján az eddigi nyelvosztályokról a következő tartalmazások nyilvánvalók:

$$\begin{array}{ccccccccc}
\mathcal{NL} & & \mathcal{NP} & \subseteq & \mathcal{NPSPACE} & & \mathcal{NEXP} & \subseteq & \mathcal{NEXPSPACE} \\
\cup| & & \cup| & & \cup| & & \cup| & & \cup| \\
\mathcal{L} & \subseteq & \mathcal{P} & \subseteq & \mathcal{PSPACE} & \subseteq & \mathcal{EXP} & \subseteq & \mathcal{EXPSPACE}
\end{array}$$

Technikai definíciók

Technikai definíciók

Definíció

Egy $t(n) : \mathbb{N} \to \mathbb{N}$ függvényt szép időfüggvénynek nevezzük, ha van olyan Turing-gép, hogy minden n hosszú inputon pontosan $t(n)$ ideig fut.

Technikai definíciók

Definíció

Egy $t(n) : \mathbb{N} \to \mathbb{N}$ függvényt szép időfüggvénynek nevezzük, ha van olyan Turing-gép, hogy minden n hosszú inputon pontosan $t(n)$ ideig fut.

Definíció

Egy $s(n) : \mathbb{N} \to \mathbb{N}$ függvényt szép tárfüggvénynek nevezzük, ha van olyan Turing-gép, amely minden n hosszú inputon leáll és pontosan $s(n)$ mezőt érint a munkaszalagon.

Technikai definíciók

Definíció

Egy $t(n) : \mathbb{N} \to \mathbb{N}$ függvényt szép időfüggvénynek nevezzük, ha van olyan Turing-gép, hogy minden n hosszú inputon pontosan $t(n)$ ideig fut.

Definíció

Egy $s(n) : \mathbb{N} \to \mathbb{N}$ függvényt szép tárfüggvénynek nevezzük, ha van olyan Turing-gép, amely minden n hosszú inputon leáll és pontosan $s(n)$ mezőt érint a munkaszalagon.

A fentiek technikai feltételek. Azonban az eddig használt függvények mind szépek.

Technikai definíciók

Definíció

Egy $t(n) : \mathbb{N} \to \mathbb{N}$ függvényt szép időfüggvénynek nevezzük, ha van olyan Turing-gép, hogy minden n hosszú inputon pontosan $t(n)$ ideig fut.

Definíció

Egy $s(n) : \mathbb{N} \to \mathbb{N}$ függvényt szép tárfüggvénynek nevezzük, ha van olyan Turing-gép, amely minden n hosszú inputon leáll és pontosan $s(n)$ mezőt érint a munkaszalagon.

A fentiek technikai feltételek. Azonban az eddig használt függvények mind szépek. Például idő esetén a polinom függvények, 2^n, vagy tár esetén $\lceil \log_2 n \rceil$ is.

Példa

Példa

Példa

Feltesszük, hogy $t(n)$ szép időfüggvény. Legyen T egy tetszőleges Turing-gép. Legyen ω egy tetszőleges n hosszú input.

Példa

Példa

Feltesszük, hogy $t(n)$ szép időfüggvény. Legyen T egy tetszőleges Turing-gép. Legyen ω egy tetszőleges n hosszú input.

ω-t átmásoljuk egy munkaszalagra, majd a szemek/kezek balra visszamennek ($2n$ idő).

Példa

Példa

Feltesszük, hogy $t(n)$ szép időfüggvény. Legyen T egy tetszőleges Turing-gép. Legyen ω egy tetszőleges n hosszú input.

ω-t átmásoljuk egy munkaszalagra, majd a szemek/kezek balra visszamennek ($2n$ idő).

Innen párhuzamosan szimuláljuk a T gép futását és a munkaszalag segítségével egy $t(n)$ szépségét mutató W gépet. A T gép ELFOGAD/ELVET állapota leállítja gépünket.

Példa

Példa

Feltesszük, hogy $t(n)$ szép időfüggvény. Legyen T egy tetszőleges Turing-gép. Legyen ω egy tetszőleges n hosszú input.

ω-t átmásoljuk egy munkaszalagra, majd a szemek/kezek balra visszamennek ($2n$ idő).

Innen párhuzamosan szimuláljuk a T gép futását és a munkaszalag segítségével egy $t(n)$ szépségét mutató W gépet. A T gép ELFOGAD/ELVET állapota leállítja gépünket.

W leálló állapotát CSÖRÖG-nek nevezzük.

Példa

Példa

Feltesszük, hogy $t(n)$ szép időfüggvény. Legyen T egy tetszőleges Turing-gép. Legyen ω egy tetszőleges n hosszú input.

ω-t átmásoljuk egy munkaszalagra, majd a szemek/kezek balra visszamennek ($2n$ idő).

Innen párhuzamosan szimuláljuk a T gép futását és a munkaszalag segítségével egy $t(n)$ szépségét mutató W gépet. A T gép ELFOGAD/ELVET állapota leállítja gépünket.

W leálló állapotát CSÖRÖG-nek nevezzük. W-re úgy gondolunk mint egy órára.

Példa

Példa

Feltesszük, hogy $t(n)$ szép időfüggvény. Legyen T egy tetszőleges Turing-gép. Legyen ω egy tetszőleges n hosszú input.

ω-t átmásoljuk egy munkaszalagra, majd a szemek/kezek balra visszamennek ($2n$ idő).

Innen párhuzamosan szimuláljuk a T gép futását és a munkaszalag segítségével egy $t(n)$ szépségét mutató W gépet. A T gép ELFOGAD/ELVET állapota leállítja gépünket.

W leálló állapotát CSÖRÖG-nek nevezzük. W-re úgy gondolunk mint egy órára. A CSÖRÖG állapot az egész szimulációt leállítja.

Példa

Példa

Feltesszük, hogy $t(n)$ szép időfüggvény. Legyen T egy tetszőleges Turing-gép. Legyen ω egy tetszőleges n hosszú input.

ω-t átmásoljuk egy munkaszalagra, majd a szemek/kezek balra visszamennek ($2n$ idő).

Innen párhuzamosan szimuláljuk a T gép futását és a munkaszalag segítségével egy $t(n)$ szépségét mutató W gépet. A T gép ELFOGAD/ELVET állapota leállítja gépünket.

W leálló állapotát CSÖRÖG-nek nevezzük. W-re úgy gondolunk mint egy órára. A CSÖRÖG állapot az egész szimulációt leállítja.

Így az új gépünk garantáltan $2n + t(n)$ időben leáll ($t(n) + 2n$ és $t(n)$ nagyságrendileg megegyezik). T számításait elvégzi, ha azok a $t(n)$ időbe befférnek.

Példa

Példa

Példa

Feltesszük, hogy $s(n)$ szép tárfüggvény. Legyen T egy tetszőleges Turing-gép. Legyen ω egy tetszőleges n hosszú input.

Példa

Példa

Feltesszük, hogy $s(n)$ szép tárfüggvény. Legyen T egy tetszőleges Turing-gép. Legyen ω egy tetszőleges n hosszú input.

Először szimuláljuk az $s(n)$ szépségét mutató W gépet. Majd a használt mezőket felülírjuk egy üres jellel és mögéjük teszünk egy EDDIG karaktert.

Példa

Példa

Feltesszük, hogy $s(n)$ szép tárfüggvény. Legyen T egy tetszőleges Turing-gép. Legyen ω egy tetszőleges n hosszú input.

Először szimuláljuk az $s(n)$ szépségét mutató W gépet. Majd a használt mezőket felülírjuk egy üres jellel és mögéjük teszünk egy EDDIG karaktert.

Ezekután elkezdjük a T gép szimulálását. Ha az EDDIG karaktert olvassuk, akkor leállunk SOK-MEMÓRIA állapottal.

Példa

Példa

Feltesszük, hogy $s(n)$ szép tárfüggvény. Legyen T egy tetszőleges Turing-gép. Legyen ω egy tetszőleges n hosszú input.

Először szimuláljuk az $s(n)$ szépségét mutató W gépet. Majd a használt mezőket felülírjuk egy üres jellel és mögéjük teszünk egy EDDIG karaktert.

Ezekután elkezdjük a T gép szimulálását. Ha az EDDIG karaktert olvassuk, akkor leállunk SOK-MEMÓRIA állapottal.

A T gép ELFOGAD/ELVET állapota leállítja gépünket. Így az új gépünk garantáltan $1 + s(n)$ tárat használ ($s(n)$ és $s(n) + 1$ nagyságrendileg megegyezik).

Példa

Példa

Feltesszük, hogy $s(n)$ szép tárfüggvény. Legyen T egy tetszőleges Turing-gép. Legyen ω egy tetszőleges n hosszú input.

Először szimuláljuk az $s(n)$ szépségét mutató W gépet. Majd a használt mezőket felülírjuk egy üres jellel és mögéjük teszünk egy EDDIG karaktert.

Ezekután elkezdjük a T gép szimulálását. Ha az EDDIG karaktert olvassuk, akkor leállunk SOK-MEMÓRIA állapottal.

A T gép ELFOGAD/ELVET állapota leállítja gépünket. Így az új gépünk garantáltan $1 + s(n)$ tárat használ ($s(n)$ és $s(n) + 1$ nagyságrendileg megegyezik).

Az új gép T számításait elvégzi, ha azok a $s(n)$ tárba befférnek.

Példa

Példa

Példa

Legyen T egy nem-determinisztikus gép, amely $t(n)$ időigényű és az L nyelvet számolja ki.

Példa

Példa

Legyen T egy nem-determinisztikus gép, amely $t(n)$ időigényű és az L nyelvet számolja ki. Ennek lehet sok olyan futása lehet, aminek idejéről nem tudunk semmit.

Példa

Példa

Legyen T egy nem-determinisztikus gép, amely $t(n)$ időigényű és az L nyelvet számolja ki. Ennek lehet sok olyan futása lehet, aminek idejéről nem tudunk semmit.

A fenti példa alapján HA $t(n)$ szép, akkor feltehető, hogy gépünk minden futása $t(n) \approx t(n) + 2n$ lépés alatt megáll.

Példa

Példa

Legyen T egy nem-determinisztikus gép, amely $t(n)$ időigényű és az L nyelvet számolja ki. Ennek lehet sok olyan futása lehet, aminek idejéről nem tudunk semmit.

A fenti példa alapján HA $t(n)$ szép, akkor feltehető, hogy gépünk minden futása $t(n) \approx t(n) + 2n$ lépés alatt megáll. A leállított futások nem változtatják meg az elfogadott nyelvet.

Példa

Példa

Legyen T egy nem-determinisztikus gép, amely $t(n)$ időigényű és az L nyelvet számolja ki. Ennek lehet sok olyan futása lehet, aminek idejéről nem tudunk semmit.

A fenti példa alapján HA $t(n)$ szép, akkor feltehető, hogy gépünk minden futása $t(n) \approx t(n) + 2n$ lépés alatt megáll. A leállított futások nem változtatják meg az elfogadott nyelvet.

Az időbonyolultsági feltétel miatt $\omega \in L$ esetén lesz olyan ELFOGAD állapothoz vezető futás, ami CSÖRÖG előtt végetér. Azaz a szimuláló gép is „észleli” ezt.

Példa

Példa

Példa

Legyen $s(n)$ egy szép tárfüggvény. Legyen T egy $s(n)$ tárigényű gép. Ekkor elérhetjük, hogy gépünknek pontosan kétféle leálló konfigurációja legyen:

Példa

Példa

Legyen $s(n)$ egy szép tárfüggvény. Legyen T egy $s(n)$ tárigényű gép. Ekkor elérhetjük, hogy gépünknek pontosan kétféle leálló konfigurációja legyen:

Futtassuk T-t. A számítás végén azonban „tartsuk meg magunknak” az eredményt és az ELFOGAD/ELVET állapotok bejelentése előtt töröljük le a munkaszalagot $s(n)$ hosszban (a szépség miatt ez könnyen megtehető).

Példa

Példa

Legyen $s(n)$ egy szép tárfüggvény. Legyen T egy $s(n)$ tárigényű gép. Ekkor elérhetjük, hogy gépünknek pontosan kétféle leálló konfigurációja legyen:

Futtassuk T-t. A számítás végén azonban „tartsuk meg magunknak" az eredményt és az ELFOGAD/ELVET állapotok bejelentése előtt töröljük le a munkaszalagot $s(n)$ hosszban (a szépség miatt ez könnyen megtehető).

Minden szem/kéz mozogjon balra. Ezek után érjük el a kiszámított eredménynek megfelelő leálló állapotot.

Példa

Példa

Legyen $s(n)$ egy szép tárfüggvény. Legyen T egy $s(n)$ tárigényű gép. Ekkor elérhetjük, hogy gépünknek pontosan kétféle leálló konfigurációja legyen:

Futtassuk T-t. A számítás végén azonban „tartsuk meg magunknak" az eredményt és az ELFOGAD/ELVET állapotok bejelentése előtt töröljük le a munkaszalagot $s(n)$ hosszban (a szépség miatt ez könnyen megtehető).

Minden szem/kéz mozogjon balra. Ezek után érjük el a kiszámított eredménynek megfelelő leálló állapotot.

Kétféle leálló konfigurációnk („fénykép" a gépről) lett és természetesen gépünk ugyanazt számolja ki mint az erdeti.

Észrevétel

Észrevétel

Megjegyezzük, hogy a szép időfüggvényt (legyen W az ezt bizonyító Turing-gép) tárkijelölésre is használhatjuk:

Észrevétel

Megjegyezzük, hogy a szép időfüggvényt (legyen W az ezt bizonyító Turing-gép) tárkijelölésre is használhatjuk:

W szimulációja mellett a többi munkaszalagon a balra állított szem/kéz folyamatosan jobbra haladjon a CSÖRÖG állapotig.

Észrevétel

Megjegyezzük, hogy a szép időfüggvényt (legyen W az ezt bizonyító Turing-gép) tárkijelölésre is használhatjuk:

W szimulációja mellett a többi munkaszalagon a balra állított szem/kéz folyamatosan jobbra haladjon a CSÖRÖG állapotig.

Ekkor a W által használt szalagokon túl pontosan $t(n)$ mező lett kijelölve a további szalagokon.

Szünet

Cél

Cél

Célunk a következő tartalmazási lánc belátása:

$$\mathcal{L} \subset \mathcal{NL} \overset{(2)}{\subset} \mathcal{P} \subset \mathcal{NP} \overset{(1)}{\subset} \mathcal{PSPACE} \subset \mathcal{NPSPACE} \overset{(2)}{\subset} \mathcal{EXP} \subset \mathcal{NEXP}.$$

Cél

Célunk a következő tartalmazási lánc belátása:

$$\mathcal{L} \subset \mathcal{NL} \overset{(2)}{\subset} \mathcal{P} \subset \mathcal{NP} \overset{(1)}{\subset} \mathcal{PSPACE} \subset \mathcal{NPSPACE} \overset{(2)}{\subset} \mathcal{EXP} \subset \mathcal{NEXP}.$$

Megszámoztuk a még bizonyítatlan tartalmazásokat.

Cél

Célunk a következő tartalmazási lánc belátása:

$$\mathcal{L} \subset \mathcal{NL} \overset{(2)}{\subset} \mathcal{P} \subset \mathcal{NP} \overset{(1)}{\subset} \mathcal{PSPACE} \subset \mathcal{NPSPACE} \overset{(2)}{\subset} \mathcal{EXP} \subset \mathcal{NEXP}.$$

Megszámoztuk a még bizonyítatlan tartalmazásokat.

Az alábbiakban belátjuk ezeket. Célunk azonban nem a minél rövidebb indoklás, hanem az eredmények összefoglalása és későbbi módszerek bevezetése.

Bemelegítés

Bemelegítés

Trivialitás A

$\mathcal{TIME}(t(n)) \subset \mathcal{SPACE}(t(n))$.

Bemelegítés

Trivialitás A

$\mathcal{TIME}(t(n)) \subset \mathcal{SPACE}(t(n))$.

Valóban, az időkorlát korlátozza azt, hogy a munkaszalag szem/keze milyen messze tud elmozogni.

Bemelegítés

Trivialitás A

$\mathcal{TIME}(t(n)) \subset \mathcal{SPACE}(t(n))$.

Valóban, az időkorlát korlátozza azt, hogy a munkaszalag szem/keze milyen messze tud elmozogni.

Trivialitás B

(i) $\mathcal{TIME}(t(n)) \subset \mathcal{NTIME}(t(n))$.

(ii) $\mathcal{SPACE}(s(n)) \subset \mathcal{NSPACE}(s(n))$.

Bemelegítés

Trivialitás A

$\mathcal{TIME}(t(n)) \subset \mathcal{SPACE}(t(n))$.

Valóban, az időkorlát korlátozza azt, hogy a munkaszalag szem/keze milyen messze tud elmozogni.

Trivialitás B

(i) $\mathcal{TIME}(t(n)) \subset \mathcal{NTIME}(t(n))$.

(ii) $\mathcal{SPACE}(s(n)) \subset \mathcal{NSPACE}(s(n))$.

Valóban, a determinizmus felfogható, mint a nem-determinisztikusság egy speciális esete.

Észrevétel

Észrevétel

Észrevétel

$\mathcal{NTIME}(t(n)) \subset \mathcal{SPACE}(t(n))$, ahol $t(n)$ szép időfüggvény.

Észrevétel

Észrevétel

$\mathcal{NTIME}(t(n)) \subset \mathcal{SPACE}(t(n))$, ahol $t(n)$ szép időfüggvény.

Legyen $L \in_T \mathcal{NTIME}(t(n))$.

Észrevétel

Észrevétel

$\mathcal{NTIME}(t(n)) \subset \mathcal{SPACE}(t(n))$, ahol $t(n)$ szép időfüggvény.

Legyen $L \in_T \mathcal{NTIME}(t(n))$. Azaz T egy tanúszalagos bizonyító Turing-gép.

Észrevétel

Észrevétel

$\mathcal{NTIME}(t(n)) \subset \mathcal{SPACE}(t(n))$, ahol $t(n)$ szép időfüggvény.

Legyen $L \in_T \mathcal{NTIME}(t(n))$. Azaz T egy tanúszalagos bizonyító Turing-gép. Azaz T az L nyelvet fogadja el és minden ω inputra $t(|\omega|)$ az időbonyolultsága.

Észrevétel

Észrevétel

$\mathcal{NTIME}(t(n)) \subset \mathcal{SPACE}(t(n))$, ahol $t(n)$ szép időfüggvény.

Legyen $L \in_T \mathcal{NTIME}(t(n))$. Azaz T egy tanúszalagos bizonyító Turing-gép. Azaz T az L nyelvet fogadja el és minden ω inputra $t(|\omega|)$ az időbonyolultsága.

Az állítás bizonyításához megadunk (T-re alapulva) egy $\widetilde{T}$ egy determinisztikus Turing-gépet, amely ugyanazt a nyelvet fogadja el és tár korlátja $t(n)$ lesz.

Észrevétel

Észrevétel

$\mathcal{NTIME}(t(n)) \subset \mathcal{SPACE}(t(n))$, ahol $t(n)$ szép időfüggvény.

Legyen $L \in_T \mathcal{NTIME}(t(n))$. Azaz T egy tanúszalagos bizonyító Turing-gép. Azaz T az L nyelvet fogadja el és minden ω inputra $t(|\omega|)$ az időbonyolultsága.

Az állítás bizonyításához megadunk (T-re alapulva) egy $\widetilde{T}$ egy determinisztikus Turing-gépet, amely ugyanazt a nyelvet fogadja el és tár korlátja $t(n)$ lesz.

Ehhez megtartjuk T leírásához szükséges munkaszalagokat és hozzáadunk egyet, amely a tanú szalag szerepét tölti be és még egyet, ami egy óra szerepét tölti be ($t(n)$ szép időfüggvény).

Észrevétel

Észrevétel

$\mathcal{NTIME}(t(n)) \subset \mathcal{SPACE}(t(n))$, ahol $t(n)$ szép időfüggvény.

Legyen $L \in_T \mathcal{NTIME}(t(n))$. Azaz T egy tanúszalagos bizonyító Turing-gép. Azaz T az L nyelvet fogadja el és minden ω inputra $t(|\omega|)$ az időbonyolultsága.

Az állítás bizonyításához megadunk (T-re alapulva) egy $\widetilde{T}$ egy determinisztikus Turing-gépet, amely ugyanazt a nyelvet fogadja el és tár korlátja $t(n)$ lesz.

Ehhez megtartjuk T leírásához szükséges munkaszalagokat és hozzáadunk egyet, amely a tanú szalag szerepét tölti be és még egyet, ami egy óra szerepét tölti be ($t(n)$ szép időfüggvény).

Persze az új gép a nem-determinisztikus gépek „zsenialitását”/tippelő tulajdonságát nem birtokolja.

A terv

A terv

$\widetilde{T}$ működésének leírásához megadjuk, hogyan néz ki egy futása.

A terv

$\widetilde{T}$ működésének leírásához megadjuk, hogyan néz ki egy futása. Ebből az átmenetifüggvény (formális leírása) kiolvasható.

A terv

$\widetilde{T}$ működésének leírásához megadjuk, hogyan néz ki egy futása. Ebből az átmenetifüggvény (formális leírása) kiolvasható.

Feltesszük, hogy az inputunk hossza n.

$\widetilde{T}$: Inicializáló fázis

$\widetilde{T}$: Inicializáló fázis

A tanúszalag szerepét betöltő munkaszalagon kijelölünk $t(n)$ számú mezőt, melyet egy Γ-beli speciális határolójellel lezárunk.

$\widetilde{T}$: Inicializáló fázis

A tanúszalag szerepét betöltő munkaszalagon kijelölünk $t(n)$ számú mezőt, melyet egy Γ-beli speciális határolójellel lezárunk. Ez egy csak erre a célra használt karakter.

$\widetilde{T}$: Inicializáló fázis

A tanúszalag szerepét betöltő munkaszalagon kijelölünk $t(n)$ számú mezőt, melyet egy Γ-beli speciális határolójellel lezárunk. Ez egy csak erre a célra használt karakter. Ezen karakter olvasásakor tudjuk, hogy a tár korlát betartása mellett nem léphetünk jobbra.

$\widetilde{T}$: Inicializáló fázis

A tanúszalag szerepét betöltő munkaszalagon kijelölünk $t(n)$ számú mezőt, melyet egy Γ-beli speciális határolójellel lezárunk. Ez egy csak erre a célra használt karakter. Ezen karakter olvasásakor tudjuk, hogy a tár korlát betartása mellett nem léphetünk jobbra.

$t(n)$ szép időfüggvény, azaz vehetünk egy órát, ami $t(n)$ lépés után „csörög” (és persze újra felhúzható).

$\widetilde{T}$: Inicializáló fázis

A tanúszalag szerepét betöltő munkaszalagon kijelölünk $t(n)$ számú mezőt, melyet egy Γ-beli speciális határolójellel lezárunk. Ez egy csak erre a célra használt karakter. Ezen karakter olvasásakor tudjuk, hogy a tár korlát betartása mellett nem léphetünk jobbra.

$t(n)$ szép időfüggvény, azaz vehetünk egy órát, ami $t(n)$ lépés után „csörög” (és persze újra felhúzható).

Ennek segítségével a tárterület kijelölése könnyen megoldható: a munka szalag felett a csörgésig jobbra mozgunk.

$\widetilde{T}$: Inicializáló fázis

A tanúszalag szerepét betöltő munkaszalagon kijelölünk $t(n)$ számú mezőt, melyet egy Γ-beli speciális határolójellel lezárunk. Ez egy csak erre a célra használt karakter. Ezen karakter olvasásakor tudjuk, hogy a tár korlát betartása mellett nem léphetünk jobbra.

$t(n)$ szép időfüggvény, azaz vehetünk egy órát, ami $t(n)$ lépés után „csörög” (és persze újra felhúzható).

Ennek segítségével a tárterület kijelölése könnyen megoldható: a munka szalag felett a csörgésig jobbra mozgunk.

A tanúszalag szerepét betöltő munkaszalagra felírjuk az első lehetséges $t(n)$ karaktert, ami egy tanú-kezdet lehet.

$\widetilde{T}$: Inicializáló fázis

A tanúszalag szerepét betöltő munkaszalagon kijelölünk $t(n)$ számú mezőt, melyet egy Γ-beli speciális határolójellel lezárunk. Ez egy csak erre a célra használt karakter. Ezen karakter olvasásakor tudjuk, hogy a tár korlát betartása mellett nem léphetünk jobbra.

$t(n)$ szép időfüggvény, azaz vehetünk egy órát, ami $t(n)$ lépés után „csörög” (és persze újra felhúzható).

Ennek segítségével a tárterület kijelölése könnyen megoldható: a munka szalag felett a csörgésig jobbra mozgunk.

A tanúszalag szerepét betöltő munkaszalagra felírjuk az első lehetséges $t(n)$ karaktert, ami egy tanú-kezdet lehet.
// Több karakterre nincs szükségünk mert $t(n)$ időkorlátos gép nem tud többet elolvasni.

$\widetilde{T}$: Szimuláló fázis

$\widetilde{T}$: Szimuláló fázis

A T Turing-gép munkaszalagjainak megfelelő szalagokon szimuláljuk T futását az első tanún $t(n)$ ideig. A szimuláció vagy ELFOGAD, vagy NEM-STIMMEL állapottal ér véget, vagy letelik az idő/kifutunk a $t(n)$ időből. Ez utóbbit is a tesztelt tanú elvetéseként (NEM-STIMMEL állapot) fogjuk fel.
Ha a szimuláció ELFOGAD álapotba jutott, akkor mi is ELFOGADjuk az inputot, $\widetilde{T}$ is leáll. Ha NEM-STIMMEL állapotba jutott, akkor a tanú szalag szerepét betöltő szalagon a következő lehetséges $t(n)$ hosszú tanúkezdettel írjuk felül eddigi tartalmát. A többi szalag tartalmát letöröljük. Megismételjük a Szimuláló fázist. Ha a következő tanú-kezdet generálása nem lehetséges, mert az összes tanú-kezdetet teszteltük (a tanúk kimerültek), akkor ELVET állapottal leállunk.

$\widetilde{T}$: Tulajdonságok

$\widetilde{T}$: Tulajdonságok

Ekkor a következő két állítás egyszerűen adódik az előzőekből:

$\widetilde{T}$: Tulajdonságok

Ekkor a következő két állítás egyszerűen adódik az előzőekből:

(1) $\widetilde{T}$ az L nyelvet számolja ki,

(2) $\widetilde{T}$ tárigénye legfeljebb $t(n)$.

$\widetilde{T}$: Tulajdonságok

Ekkor a következő két állítás egyszerűen adódik az előzőekből:

(1) $\widetilde{T}$ az L nyelvet számolja ki,
(2) $\widetilde{T}$ tárigénye legfeljebb $t(n)$.

Ezzel az észrevételt igazoltuk.

$\widetilde{T}$: Tulajdonságok

Ekkor a következő két állítás egyszerűen adódik az előzőekből:

(1) $\widetilde{T}$ az L nyelvet számolja ki,
(2) $\widetilde{T}$ tárigénye legfeljebb $t(n)$.

Ezzel az észrevételt igazoltuk.

Ezzel speciálisan adódott a (1)-vel jelölt tartalmazás.

Észrevétel

Észrevétel

A nem-determinizmus tárgyalása előtt láttuk, hogy

$$\mathcal{SPACE}(s(n)) \subseteq \cup_{c \in \mathbb{N}} \mathcal{TIME}(c^{s(n)+\log(n+1)}).$$

Észrevétel

A nem-determinizmus tárgyalása előtt láttuk, hogy

$$\mathcal{SPACE}(s(n)) \subseteq \cup_{c \in \mathbb{N}} \mathcal{TIME}(c^{s(n)+\log(n+1)}).$$

Most ezt terjesztjük ki a nem-determinisztikus esetre.

Észrevétel

A nem-determinizmus tárgyalása előtt láttuk, hogy

$$\mathcal{SPACE}(s(n)) \subseteq \cup_{c \in \mathbb{N}} \mathcal{TIME}(c^{s(n)+\log(n+1)}).$$

Most ezt terjesztjük ki a nem-determinisztikus esetre.

Az észrevételt viszonylag gyorsan beláthatnánk. Mi egy lassabb utat választunk. A módszerünk a későbbiekben nagyon fontos lesz.

Észrevétel

A nem-determinizmus tárgyalása előtt láttuk, hogy

$$\mathcal{SPACE}(s(n)) \subseteq \cup_{c \in \mathbb{N}} \mathcal{TIME}(c^{s(n)+\log(n+1)}).$$

Most ezt terjesztjük ki a nem-determinisztikus esetre.

Az észrevételt viszonylag gyorsan beláthatnánk. Mi egy lassabb utat választunk. A módszerünk a későbbiekben nagyon fontos lesz. A bizonyítás gondolatmenete a későbbiekben fontos lesz.

Észrevétel

A nem-determinizmus tárgyalása előtt láttuk, hogy

$$\mathcal{SPACE}(s(n)) \subseteq \cup_{c\in\mathbb{N}}\mathcal{TIME}(c^{s(n)+\log(n+1)}).$$

Most ezt terjesztjük ki a nem-determinisztikus esetre.

Az észrevételt viszonylag gyorsan beláthatnánk. Mi egy lassabb utat választunk. A módszerünk a későbbiekben nagyon fontos lesz. A bizonyítás gondolatmenete a későbbiekben fontos lesz.

Észrevétel

$\mathcal{NSPACE}(s(n)) \subseteq \bigcup_{c\in\mathbb{N}} \mathcal{TIME}(c^{s(n)+\log(n+1)})$, ahol $s(n)$ szép tárfüggvény.

Előkészületek

Előkészületek

Legyen $L \in_T \mathcal{NSPACE}(s(n))$.

Előkészületek

Legyen $L \in_T \mathcal{NSPACE}(s(n))$. Azaz T (I. értelemben vett) nem-determinisztikus Turing-gép, vagyis az átmeneti függvény nem-determinisztikus, a futás „szétágazó" lehet. T kiszámolja L-et és tárigénye $s(n)$.

Előkészületek

Legyen $L \in_T \mathcal{NSPACE}(s(n))$. Azaz T (I. értelemben vett) nem-determinisztikus Turing-gép, vagyis az átmeneti függvény nem-determinisztikus, a futás „szétágazó" lehet. T kiszámolja L-et és tárigénye $s(n)$.

T-ről feltehető, hogy leálláskor az input- illetve a munkafej a szalag elejére áll, továbbá a munkaszalag első $s(n)$ karaktere üres (a gép leradirozza a munkaterületét).

Előkészületek

Legyen $L \in_T \mathcal{NSPACE}(s(n))$. Azaz T (I. értelemben vett) nem-determinisztikus Turing-gép, vagyis az átmeneti függvény nem-determinisztikus, a futás „szétágazó" lehet. T kiszámolja L-et és tárigénye $s(n)$.

T-ről feltehető, hogy leálláskor az input- illetve a munkafej a szalag elejére áll, továbbá a munkaszalag első $s(n)$ karaktere üres (a gép leradirozza a munkaterületét). Így a leálláskor két konfiguráció fordulhat elő.

Előkészületek

Legyen $L \in_T \mathcal{NSPACE}(s(n))$. Azaz T (I. értelemben vett) nem-determinisztikus Turing-gép, vagyis az átmeneti függvény nem-determinisztikus, a futás „szétágazó" lehet. T kiszámolja L-et és tárigénye $s(n)$.

T-ről feltehető, hogy leálláskor az input- illetve a munkafej a szalag elejére áll, továbbá a munkaszalag első $s(n)$ karaktere üres (a gép leradirozza a munkaterületét). Így a leálláskor két konfiguráció fordulhat elő. Speciálisan elfogadó futás során tudjuk mi az utolsó konfiguráció.

Redukált konfiguráció

Redukált konfiguráció

Redukált konfiguráció $s(n)$ tárigényű I-nem-determinisztikus Turing-gép esetén adott ω inputra nézve a következő komponenseket tartalmazza:

(1) input- és munkafej pozíciója,
(2) munkaszalag első $s(n)$ karaktere,
(3) a gép állapota.

Redukált konfiguráció

Redukált konfiguráció $s(n)$ tárigényű I-nem-determinisztikus Turing-gép esetén adott ω inputra nézve a következő komponenseket tartalmazza:

(1) input- és munkafej pozíciója,
(2) munkaszalag első $s(n)$ karaktere,
(3) a gép állapota.

Tulajdonképpen csak az inputszalag tartalmát és a munkaszalag garantáltan olvasatlan/érintetlen részét takartuk le a (teljes) konfigurációból.

Redukált konfiguráció

Redukált konfiguráció $s(n)$ tárigényű I-nem-determinisztikus Turing-gép esetén adott ω inputra nézve a következő komponenseket tartalmazza:

(1) input- és munkafej pozíciója,
(2) munkaszalag első $s(n)$ karaktere,
(3) a gép állapota.

Tulajdonképpen csak az inputszalag tartalmát és a munkaszalag garantáltan olvasatlan/érintetlen részét takartuk le a (teljes) konfigurációból.

A redukált konfigurációk halmaza legyen V. Ekkor

$$|V| \leq \alpha_T \cdot (n+1) \cdot \beta_T^{s(n)}.$$

Redukált konfiguráció

Redukált konfiguráció $s(n)$ tárigényű I-nem-determinisztikus Turing-gép esetén adott ω inputra nézve a következő komponenseket tartalmazza:

(1) input- és munkafej pozíciója,
(2) munkaszalag első $s(n)$ karaktere,
(3) a gép állapota.

Tulajdonképpen csak az inputszalag tartalmát és a munkaszalag garantáltan olvasatlan/érintetlen részét takartuk le a (teljes) konfigurációból.

A redukált konfigurációk halmaza legyen V. Ekkor

$$|V| \leq \alpha_T \cdot (n+1) \cdot \beta_T^{s(n)}.$$

A κ konfigurációból természetesen kiolvasható a megfelelő redukált $\rho = \mathrm{red}(\kappa)$ konfiguráció.

Redukált konfiguráció

Redukált konfiguráció $s(n)$ tárigényű I-nem-determinisztikus Turing-gép esetén adott ω inputra nézve a következő komponenseket tartalmazza:

(1) input- és munkafej pozíciója,
(2) munkaszalag első $s(n)$ karaktere,
(3) a gép állapota.

Tulajdonképpen csak az inputszalag tartalmát és a munkaszalag garantáltan olvasatlan/érintetlen részét takartuk le a (teljes) konfigurációból.

A redukált konfigurációk halmaza legyen V. Ekkor

$$|V| \leq \alpha_T \cdot (n+1) \cdot \beta_T^{s(n)}.$$

A κ konfigurációból természetesen kiolvasható a megfelelő redukált $\rho = \mathrm{red}(\kappa)$ konfiguráció. Ha ismert az ω input, akkor megfordítva is igaz: ω és a ρ redukált konfiguráció meghatározza a $\kappa = \mathrm{konf}(\omega, \rho)$ teljes konfigurációt.

A gráf

A gráf

Definíció

Legyen T egy I. nemdeterminisztikus Turing-gép és ω egy inputja. Ekkor $\overrightarrow{G}_{\omega,T}$ a (T,ω)-hoz tartozó redukált konfigurációk gráfja. Ez egy irányított gráf, ahol a csúcsok halmaza a fenti V halmaz, továbbá $\overrightarrow{uv}$ akkor és csak akkor él, ha az konf(ω,u) konfiguráció után az átmeneti függvény megengedi a konf(ω,v) konfigurációt.

A gráf

Definíció

Legyen T egy I. nemdeterminisztikus Turing-gép és ω egy inputja. Ekkor $\overrightarrow{G}_{\omega,T}$ a (T,ω)-hoz tartozó redukált konfigurációk gráfja. Ez egy irányított gráf, ahol a csúcsok halmaza a fenti V halmaz, továbbá $\overrightarrow{uv}$ akkor és csak akkor él, ha az konf(ω,u) konfiguráció után az átmeneti függvény megengedi a konf(ω,v) konfigurációt.

Legyen V speciális eleme $v_0 = \text{red}(\kappa_0(\omega))$ a kezdő konfiguráció redukáltja.

A gráf

Definíció

Legyen T egy I. nemdeterminisztikus Turing-gép és ω egy inputja. Ekkor $\overrightarrow{G}_{\omega,T}$ a (T,ω)-hoz tartozó redukált konfigurációk gráfja. Ez egy irányított gráf, ahol a csúcsok halmaza a fenti V halmaz, továbbá $\overrightarrow{uv}$ akkor és csak akkor él, ha az konf(ω,u) konfiguráció után az átmeneti függvény megengedi a konf(ω,v) konfigurációt.

Legyen V speciális eleme $v_0 = \text{red}(\kappa_0(\omega))$ a kezdő konfiguráció redukáltja.

Legyen v_1 az elfogadó leállásnak megfelelő konfiguráció redukáltja.

A gráf

Definíció

Legyen T egy I. nemdeterminisztikus Turing-gép és ω egy inputja. Ekkor $\overrightarrow{G}_{\omega,T}$ a (T,ω)-hoz tartozó redukált konfigurációk gráfja. Ez egy irányított gráf, ahol a csúcsok halmaza a fenti V halmaz, továbbá $\overrightarrow{uv}$ akkor és csak akkor él, ha az konf(ω,u) konfiguráció után az átmeneti függvény megengedi a konf(ω,v) konfigurációt.

Legyen V speciális eleme $v_0 = \text{red}(\kappa_0(\omega))$ a kezdő konfiguráció redukáltja.

Legyen v_1 az elfogadó leállásnak megfelelő konfiguráció redukáltja.

Megjegyezzük, hogy determinisztikus gép esetén is bevezethetők a fenti fogalmak. Ekkor a definiált irányított gráf minden pontjának kifoka 1 lenne.

Észrevétel

Észrevétel

Észrevétel

$\omega \in L$ pontosan akkor teljesül, ha $\overrightarrow{G}_{\omega,T}$-ben létezik $v_0 v_1$ irányított út.

Észrevétel

Észrevétel

$\omega \in L$ pontosan akkor teljesül, ha $\overrightarrow{G}_{\omega,T}$-ben létezik $v_0 v_1$ irányított út.

Valóban:

Észrevétel

Észrevétel

$\omega \in L$ pontosan akkor teljesül, ha $\overrightarrow{G}_{\omega,T}$-ben létezik $v_0 v_1$ irányított út.

Valóban: $\omega \in L$ ekvivalens elfogadó futás létezésével ω-n. A futások párbaállíthatók a v_0-ből induló irányított utakkal. Egy futás elfogadó, ha a megfelelő út v_1-be vezet.

Észrevétel

Észrevétel

$\omega \in L$ pontosan akkor teljesül, ha $\overrightarrow{G}_{\omega,T}$-ben létezik $v_0 v_1$ irányított út.

Valóban: $\omega \in L$ ekvivalens elfogadó futás létezésével ω-n. A futások párbaállíthatók a v_0-ből induló irányított utakkal. Egy futás elfogadó, ha a megfelelő út v_1-be vezet.

A fent definiált gráf hatékonyan kiszámítható.

Lemma

Lemma

Lemma

Van olyan $T_1(T)$ determinisztikus gép, amely ω inputon kiszámítja a $\overrightarrow{G}_{\omega,T}, v_0, v_1$ hármasnak a kódját.

Lemma

Lemma

Van olyan $T_1(T)$ determinisztikus gép, amely ω inputon kiszámítja a $\overrightarrow{G}_{\omega,T}, v_0, v_1$ hármasnak a kódját.

Továbbá $T_1(T)$ determinisztikus tárigénye

$$\alpha_T(s(n) + \log(n+1)).$$

Lemma

Lemma

Van olyan $T_1(T)$ determinisztikus gép, amely ω inputon kiszámítja a $\overrightarrow{G}_{\omega,T}, v_0, v_1$ hármasnak a kódját.

Továbbá $T_1(T)$ determinisztikus tárigénye

$$\alpha_T(s(n) + \log(n+1)).$$

A következő állítás lényege, hogy a T, ω mögött álló gráf kiszámítására a munkaszalagon nagyon kevés hely felhasználására van szükségünk.

Lemma

Lemma

Van olyan $T_1(T)$ determinisztikus gép, amely ω inputon kiszámítja a $\overrightarrow{G}_{\omega,T}, v_0, v_1$ hármasnak a kódját.

Továbbá $T_1(T)$ determinisztikus tárigénye

$$\alpha_T(s(n) + \log(n+1)).$$

A következő állítás lényege, hogy a T, ω mögött álló gráf kiszámítására a munkaszalagon nagyon kevés hely felhasználására van szükségünk. A „spórolásnak” ebben a pillanatban nincs értelme. Jelentősége később lesz.

Lemma

Lemma

Van olyan $T_1(T)$ determinisztikus gép, amely ω inputon kiszámítja a $\overrightarrow{G}_{\omega,T}, v_0, v_1$ hármasnak a kódját.

Továbbá $T_1(T)$ determinisztikus tárigénye

$$\alpha_T(s(n) + \log(n+1)).$$

A következő állítás lényege, hogy a T, ω mögött álló gráf kiszámítására a munkaszalagon nagyon kevés hely felhasználására van szükségünk. A „spórolásnak" ebben a pillanatban nincs értelme. Jelentősége később lesz.

A megadott tár arra elegendő, hogy a munkaszalagon konstans számú redukált konfiguráció kódját írjuk le.

A Lemma bizonyítása

A Lemma bizonyítása

• Számunkra két redukált konfiguráció számára kell hely. A futás elején kijelölünk két blokkot a munkaszalag elején, amik egy-egy redukált konfiguráció tárolására szolgálnak ($s(n)$ szép tárfüggvény).

A Lemma bizonyítása

• Számunkra két redukált konfiguráció számára kell hely. A futás elején kijelölünk két blokkot a munkaszalag elején, amik egy-egy redukált konfiguráció tárolására szolgálnak ($s(n)$ szép tárfüggvény).

• Az első blokkban felsoroljuk az összes lehetséges kódszót. Ezek a jelöltek arra, hogy gráfunk egy csúcsát kódolják. Mindegyik jelöltnél eldöntjük, hogy redukált konfiguráció kódja-e. (Egy természetes kódolási szabály lerögzítése után az a feladat könnyen megoldható.)

A Lemma bizonyítása

• Számunkra két redukált konfiguráció számára kell hely. A futás elején kijelölünk két blokkot a munkaszalag elején, amik egy-egy redukált konfiguráció tárolására szolgálnak ($s(n)$ szép tárfüggvény).

• Az első blokkban felsoroljuk az összes lehetséges kódszót. Ezek a jelöltek arra, hogy gráfunk egy csúcsát kódolják. Mindegyik jelöltnél eldöntjük, hogy redukált konfiguráció kódja-e. (Egy természetes kódolási szabály lerögzítése után az a feladat könnyen megoldható.)

• Ha nem, akkor a következő jelöltre térünk át. Ha igen, akkor átmásoljuk az outputszalagra, majd egy ':' rakunk. Ezután fogjuk kiírni a ki-szomszédok sorozatát.

A Lemma bizonyítása (folytatás)

A Lemma bizonyítása (folytatás)

• A munkaszalag második blokkjában szintén elkezdjük a redukált konfigurációk felsorolását. Ha a munkaszalagon x és y redukált konfigurációk kódja szerepel, akkor eldöntjük, hogy x-ből vezet-e él y-hoz. Az inputszalagon ott van ω, továbbá a T Turing-gép — véges leírással — ismert számunkra. Ezen részfeladat implementációja ismét egyszerű.

A Lemma bizonyítása (folytatás)

- A munkaszalag második blokkjában szintén elkezdjük a redukált konfigurációk felsorolását. Ha a munkaszalagon x és y redukált konfigurációk kódja szerepel, akkor eldöntjük, hogy x-ből vezet-e él y-hoz. Az inputszalagon ott van ω, továbbá a T Turing-gép — véges leírással — ismert számunkra. Ezen részfeladat implementációja ismét egyszerű.

- Ha azt kapjuk, hogy vezet él, akkor y-t az outputszalagra másoljuk Ha azt kapjuk, hogy nem vezet él, akkor egyből a következő y keresésére térünk.

A Lemma bizonyítása (folytatás)

• A munkaszalag második blokkjában szintén elkezdjük a redukált konfigurációk felsorolását. Ha a munkaszalagon x és y redukált konfigurációk kódja szerepel, akkor eldöntjük, hogy x-ből vezet-e él y-hoz. Az inputszalagon ott van ω, továbbá a T Turing-gép — véges leírással — ismert számunkra. Ezen részfeladat implementációja ismét egyszerű.

• Ha azt kapjuk, hogy vezet él, akkor y-t az outputszalagra másoljuk Ha azt kapjuk, hogy nem vezet él, akkor egyből a következő y keresésére térünk.

• Ha az y kimerül, akkor az következő x keresésére térünk át. Ha az x-ek is kimerülnek, akkor kiszámoltuk $\overrightarrow{G}_{\omega,T}$ kódját.

A Lemma bizonyítása (folytatás)

A Lemma bizonyítása (folytatás)

- v_0 és v_1 kódjának felírása az outputszalagra szintén könnyen megoldható.

A Lemma bizonyítása (folytatás)

- v_0 és v_1 kódjának felírása az outputszalagra szintén könnyen megoldható.
- A részfeladatok megoldását nem részleteztük. Megvalósításuk során az adott tárkorlátot nem kell túllépnünk.

A Lemma bizonyítása (folytatás)

- v_0 és v_1 kódjának felírása az outputszalagra szintén könnyen megoldható.
- A részfeladatok megoldását nem részleteztük. Megvalósításuk során az adott tárkorlátot nem kell túllépnünk.
- A Lemma adódott.

A várva várt bionyítás

A várva várt bionyítás

- $L \in_T \mathcal{NSPACE}(s(n))$.

A várva várt bionyítás

- $L \in_T \mathcal{NSPACE}(s(n))$. Futassuk $T_1(T)$-t ω-n és írjuk le $\overrightarrow{G}_{\omega,T}, v_0, v_1$ kódját egy plusz munkaszalgra.

A várva várt bionyítás

- $L \in_T \mathcal{NSPACE}(s(n))$. Futassuk $T_1(T)$-t ω-n és írjuk le $\overrightarrow{G}_{\omega,T}, v_0, v_1$ kódját egy plusz munkaszalgra.
- Ennek a determinisztikus eljárásnak a tárigényét előbb becsültük, de nekünk a futási idejére kell becslés.

A várva várt bionyítás

- $L \in_T \mathcal{NSPACE}(s(n))$. Futassuk $T_1(T)$-t ω-n és írjuk le $\overrightarrow{G}_{\omega,T}, v_0, v_1$ kódját egy plusz munkaszalgra.
- Ennek a determinisztikus eljárásnak a tárigényét előbb becsültük, de nekünk a futási idejére kell becslés. Futási ideje legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$.

A várva várt bionyítás

• $L \in_T \mathcal{NSPACE}(s(n))$. Futassuk $T_1(T)$-t ω-n és írjuk le $\overrightarrow{G}_{\omega,T}, v_0, v_1$ kódját egy plusz munkaszalgra.

• Ennek a determinisztikus eljárásnak a tárigényét előbb becsültük, de nekünk a futási idejére kell becslés. Futási ideje legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$. A kiszámolt kódszó hossza is legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$.

A várva várt bionyítás

- $L \in_T \mathcal{NSPACE}(s(n))$. Futassuk $T_1(T)$-t ω-n és írjuk le $\overrightarrow{G}_{\omega,T}, v_0, v_1$ kódját egy plusz munkaszalgra.
- Ennek a determinisztikus eljárásnak a tárigényét előbb becsültük, de nekünk a futási idejére kell becslés. Futási ideje legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$. A kiszámolt kódszó hossza is legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$.
- Döntsük el, hogy $\overrightarrow{G}_{\omega,T}$-ben van-e v_0v_1 irányított út.

A várva várt bionyítás

- $L \in_T \mathcal{NSPACE}(s(n))$. Futassuk $T_1(T)$-t ω-n és írjuk le $\overrightarrow{G}_{\omega,T}, v_0, v_1$ kódját egy plusz munkaszalgra.
- Ennek a determinisztikus eljárásnak a tárigényét előbb becsültük, de nekünk a futási idejére kell becslés. Futási ideje legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$. A kiszámolt kódszó hossza is legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$.
- Döntsük el, hogy $\overrightarrow{G}_{\omega,T}$-ben van-e v_0v_1 irányított út.
- Erre számtalan megoldás létezik.

A várva várt bionyítás

- $L \in_T \mathcal{NSPACE}(s(n))$. Futassuk $T_1(T)$-t ω-n és írjuk le $\overrightarrow{G}_{\omega,T}, v_0, v_1$ kódját egy plusz munkaszalgra.
- Ennek a determinisztikus eljárásnak a tárigényét előbb becsültük, de nekünk a futási idejére kell becslés. Futási ideje legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$. A kiszámolt kódszó hossza is legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$.
- Döntsük el, hogy $\overrightarrow{G}_{\omega,T}$-ben van-e v_0v_1 irányított út.
- Erre számtalan megoldás létezik. Például szélességi keresés.

A várva várt bionyítás

- $L \in_T \mathcal{NSPACE}(s(n))$. Futassuk $T_1(T)$-t ω-n és írjuk le $\overrightarrow{G}_{\omega,T}, v_0, v_1$ kódját egy plusz munkaszalgra.
- Ennek a determinisztikus eljárásnak a tárigényét előbb becsültük, de nekünk a futási idejére kell becslés. Futási ideje legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$. A kiszámolt kódszó hossza is legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$.
- Döntsük el, hogy $\overrightarrow{G}_{\omega,T}$-ben van-e v_0v_1 irányított út.
- Erre számtalan megoldás létezik. Például szélességi keresés.
- Az algoritmus Turing-gépen megvalósítható (T_2).

A várva várt bionyítás

• $L \in_T \mathcal{NSPACE}(s(n))$. Futassuk $T_1(T)$-t ω-n és írjuk le $\overrightarrow{G}_{\omega,T}, v_0, v_1$ kódját egy plusz munkaszalgra.

• Ennek a determinisztikus eljárásnak a tárigényét előbb becsültük, de nekünk a futási idejére kell becslés. Futási ideje legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$. A kiszámolt kódszó hossza is legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$.

• Döntsük el, hogy $\overrightarrow{G}_{\omega,T}$-ben van-e v_0v_1 irányított út.

• Erre számtalan megoldás létezik. Például szélességi keresés.

• Az algoritmus Turing-gépen megvalósítható (T_2). Futási ideje (különösebb ötlet nélkül) polinomiális az input hosszában.

A várva várt bionyítás

- $L \in_T \mathcal{NSPACE}(s(n))$. Futassuk $T_1(T)$-t ω-n és írjuk le $\overrightarrow{G}_{\omega,T}, v_0, v_1$ kódját egy plusz munkaszalgra.
- Ennek a determinisztikus eljárásnak a tárigényét előbb becsültük, de nekünk a futási idejére kell becslés. Futási ideje legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$. A kiszámolt kódszó hossza is legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$.
- Döntsük el, hogy $\overrightarrow{G}_{\omega,T}$-ben van-e v_0v_1 irányított út.
- Erre számtalan megoldás létezik. Például szélességi keresés.
- Az algoritmus Turing-gépen megvalósítható (T_2). Futási ideje (különösebb ötlet nélkül) polinomiális az input hosszában.
- $T_1(T)$ és T_2 együtt éppen az L nyelvet dönti el és időigénye $2^{\gamma_T(s(n)+\log(n+1))}$. Ez adja a bizonyítandót.

A várva várt bionyítás

- $L \in_T \mathcal{NSPACE}(s(n))$. Futassuk $T_1(T)$-t ω-n és írjuk le $\overrightarrow{G}_{\omega,T}, v_0, v_1$ kódját egy plusz munkaszalgra.
- Ennek a determinisztikus eljárásnak a tárigényét előbb becsültük, de nekünk a futási idejére kell becslés. Futási ideje legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$. A kiszámolt kódszó hossza is legfeljebb $2^{\beta_T(s(n)+\log(n+1))}$.
- Döntsük el, hogy $\overrightarrow{G}_{\omega,T}$-ben van-e v_0v_1 irányított út.
- Erre számtalan megoldás létezik. Például szélességi keresés.
- Az algoritmus Turing-gépen megvalósítható (T_2). Futási ideje (különösebb ötlet nélkül) polinomiális az input hosszában.
- $T_1(T)$ és T_2 együtt éppen az L nyelvet dönti el és időigénye $2^{\gamma_T(s(n)+\log(n+1))}$. Ez adja a bizonyítandót.
- Észrevételünkből a két (2)-vel jelölt tartalmazás is adódik.

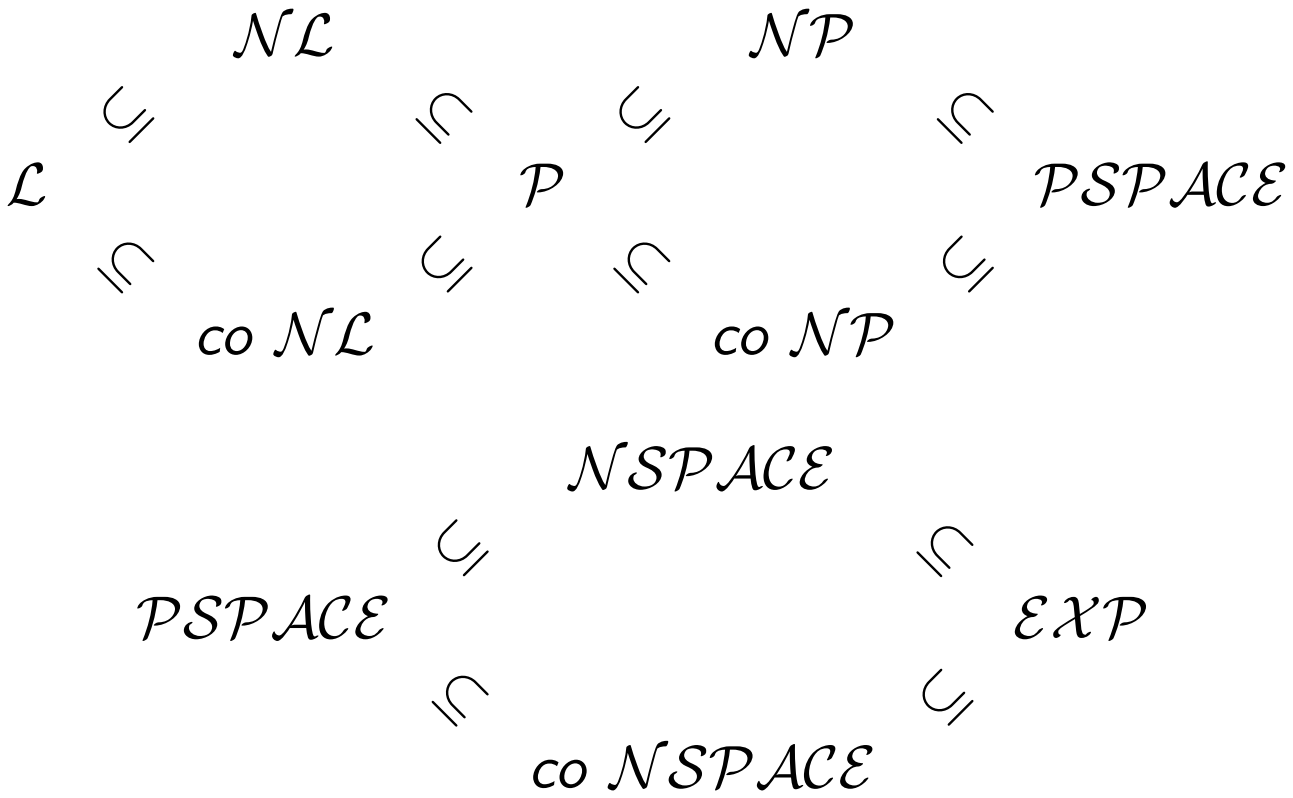
Eddigi tudásunk

Eddigi tudásunk

A bizonyított tartalmazásokat kiegészíthetjük a nem-determinisztikus osztályok komplementer nyelveinek osztályaival:

Eddigi tudásunk

A bizonyított tartalmazásokat kiegészíthetjük a nem-determinisztikus osztályok komplementer nyelveinek osztályaival:



További egybeesések

További egybeesések

$$\mathcal{NL} = \mathrm{co}\mathcal{NL},$$

További egybeesések

$$\mathcal{NL} = \text{co}\mathcal{NL},$$

$$\mathcal{PSPACE} = \mathcal{NPSPACE} = \text{co}\mathcal{NPSPACE}.$$

Ezeket az összefüggéseket később tárgyaljuk, ha időnk megengedi.

További valódi tartalmazások

További valódi tartalmazások

Az is igaz, hogy az idő, illetve tárkorlát lényeges emelésével bővebb osztályhoz jutunk:

$$\mathcal{L} \subsetneq \mathcal{PSPACE},$$

$$\mathcal{P} \subsetneq \mathcal{EXP}.$$

További valódi tartalmazások

Az is igaz, hogy az idő, illetve tárkorlát lényeges emelésével bővebb osztályhoz jutunk:

$$\mathcal{L} \subsetneq \mathcal{PSPACE},$$

$$\mathcal{P} \subsetneq \mathcal{EXP}.$$

Ennél több azonban nem ismert.

További valódi tartalmazások

Az is igaz, hogy az idő, illetve tárkorlát lényeges emelésével bővebb osztályhoz jutunk:

$$\mathcal{L} \subsetneq \mathcal{PSPACE},$$

$$\mathcal{P} \subsetneq \mathcal{EXP}.$$

Ennél több azonban nem ismert.

Azt a kérdést, hogy „A $\mathcal{P} \subseteq \mathcal{NP}$ tartalmazás valódi, vagy egyenlőség áll fenn?” sokan a XXI. századi matematika központi problémájának tartják.

Szünet

IDEÁL-ELEM-TESZT

IDEÁL-ELEM-TESZT

IDEÁL-ELEM-TESZT

Inputja egy végesen generált ideál a $\mathbb{Q}[x_1, x_2, \ldots, x_n]$ polinomgyűrűben és egy p polinom. Az ideál $g_1, g_2, \ldots, g_N$ generáló polinomokkal adott. A kérdés, hogy p az ideálhoz tartozik-e.

IDEÁL-ELEM-TESZT

IDEÁL-ELEM-TESZT

Inputja egy végesen generált ideál a $\mathbb{Q}[x_1, x_2, \ldots, x_n]$ polinomgyűrűben és egy p polinom. Az ideál $g_1, g_2, \ldots, g_N$ generáló polinomokkal adott. A kérdés, hogy p az ideálhoz tartozik-e.

Könnyű leírni az ideált: az $\alpha_1 g_1 + \alpha_2 g_2 + \ldots + \alpha_N g_N$ alakú polinomok, ahol az α_i együtthatók is polinomok.

IDEÁL-ELEM-TESZT

IDEÁL-ELEM-TESZT

Inputja egy végesen generált ideál a $\mathbb{Q}[x_1, x_2, \ldots, x_n]$ polinomgyűrűben és egy p polinom. Az ideál $g_1, g_2, \ldots, g_N$ generáló polinomokkal adott. A kérdés, hogy p az ideálhoz tartozik-e.

Könnyű leírni az ideált: az $\alpha_1 g_1 + \alpha_2 g_2 + \ldots + \alpha_N g_N$ alakú polinomok, ahol az α_i együtthatók is polinomok. Hogy egy hatékony nem-determinisztikus algoritmust adjunk ez alapján kellene egy becslés az ideálhoz tartozást bizonyító együtthatókra (fokaikra és együtthatóikra).

IDEÁL-ELEM-TESZT

IDEÁL-ELEM-TESZT

Inputja egy végesen generált ideál a $\mathbb{Q}[x_1, x_2, \ldots, x_n]$ polinomgyűrűben és egy p polinom. Az ideál $g_1, g_2, \ldots, g_N$ generáló polinomokkal adott. A kérdés, hogy p az ideálhoz tartozik-e.

Könnyű leírni az ideált: az $\alpha_1 g_1 + \alpha_2 g_2 + \ldots + \alpha_N g_N$ alakú polinomok, ahol az α_i együtthatók is polinomok. Hogy egy hatékony nem-determinisztikus algoritmust adjunk ez alapján kellene egy becslés az ideálhoz tartozást bizonyító együtthatókra (fokaikra és együtthatóikra). Ez nem egyszerű.

IDEÁL-ELEM-TESZT

IDEÁL-ELEM-TESZT

Inputja egy végesen generált ideál a $\mathbb{Q}[x_1, x_2, \ldots, x_n]$ polinomgyűrűben és egy p polinom. Az ideál $g_1, g_2, \ldots, g_N$ generáló polinomokkal adott. A kérdés, hogy p az ideálhoz tartozik-e.

Könnyű leírni az ideált: az $\alpha_1 g_1 + \alpha_2 g_2 + \ldots + \alpha_N g_N$ alakú polinomok, ahol az α_i együtthatók is polinomok. Hogy egy hatékony nem-determinisztikus algoritmust adjunk ez alapján kellene egy becslés az ideálhoz tartozást bizonyító együtthatókra (fokaikra és együtthatóikra). Ez nem egyszerű.

A Gröbner-bázisok elméletén alapulva $\mathcal{EXPSPACE}$ bonyolultságú algoritmus adható a problémára.

IDEÁL-ELEM-TESZT

IDEÁL-ELEM-TESZT

Inputja egy végesen generált ideál a $\mathbb{Q}[x_1, x_2, \ldots, x_n]$ polinomgyűrűben és egy p polinom. Az ideál $g_1, g_2, \ldots, g_N$ generáló polinomokkal adott. A kérdés, hogy p az ideálhoz tartozik-e.

Könnyű leírni az ideált: az $\alpha_1 g_1 + \alpha_2 g_2 + \ldots + \alpha_N g_N$ alakú polinomok, ahol az α_i együtthatók is polinomok. Hogy egy hatékony nem-determinisztikus algoritmust adjunk ez alapján kellene egy becslés az ideálhoz tartozást bizonyító együtthatókra (fokaikra és együtthatóikra). Ez nem egyszerű.

A Gröbner-bázisok elméletén alapulva $\mathcal{EXPSPACE}$ bonyolultságú algoritmus adható a problémára. Azaz

$$\text{IDEÁL-ELEM-TESZT} \in \mathcal{EXPSPACE}.$$

IDEÁL-TELJESSÉG

IDEÁL-TELJESSÉG

IDEÁL-TELJESSÉG

Inputja egy végesen generált ideál a $\mathbb{Q}[x_1, x_2, \ldots, x_n]$ polinomgyűrűben. Az ideál $g_1, g_2, \ldots, g_N$ generáló polinomokkal van leírva. A kérdés, hogy az ideál a teljes gyűrű-e, azaz 1 az ideálhoz tartozik-e.

IDEÁL-TELJESSÉG

IDEÁL-TELJESSÉG

Inputja egy végesen generált ideál a $\mathbb{Q}[x_1, x_2, \ldots, x_n]$ polinomgyűrűben. Az ideál $g_1, g_2, \ldots, g_N$ generáló polinomokkal van leírva. A kérdés, hogy az ideál a teljes gyűrű-e, azaz 1 az ideálhoz tartozik-e.

Ez nyilván az előző probléma egy speciális esete. Bonyolultsága legfeljebb akkora mint az előző kérdésé.

IDEÁL-TELJESSÉG

IDEÁL-TELJESSÉG

Inputja egy végesen generált ideál a $\mathbb{Q}[x_1, x_2, \ldots, x_n]$ polinomgyűrűben. Az ideál $g_1, g_2, \ldots, g_N$ generáló polinomokkal van leírva. A kérdés, hogy az ideál a teljes gyűrű-e, azaz 1 az ideálhoz tartozik-e.

Ez nyilván az előző probléma egy speciális esete. Bonyolultsága legfeljebb akkora mint az előző kérdésé.

A Gröbner-bázisok elméletén alapulva $\mathcal{PSPACE}$ bonyolultságú algoritmus adható a problémára.

IDEÁL-TELJESSÉG

IDEÁL-TELJESSÉG

Inputja egy végesen generált ideál a $\mathbb{Q}[x_1, x_2, \ldots, x_n]$ polinomgyűrűben. Az ideál $g_1, g_2, \ldots, g_N$ generáló polinomokkal van leírva. A kérdés, hogy az ideál a teljes gyűrű-e, azaz 1 az ideálhoz tartozik-e.

Ez nyilván az előző probléma egy speciális esete. Bonyolultsága legfeljebb akkora mint az előző kérdésé.

A Gröbner-bázisok elméletén alapulva $\mathcal{PSPACE}$ bonyolultságú algoritmus adható a problémára. Azaz az algoritmuselmélet ki tudja használni a specialitását a problémának (az előző kérdéshez képest).

$$\text{IDEÁL-TELJESSÉG} \in \mathcal{PSPACE}.$$

SLIDING-BLOCK-PUZZLE

SLIDING-BLOCK-PUZZLE

SLIDING-BLOCK-PUZZLE

Inputja egy $n \times m$ táblázatban (mint alappályán) elhelyezett egymást át nem fedő téglalapok. A téglalapok a pályát nem fedik le teljesen, így lehetőség van tologatásukra (az alaptábla oldalaival párhuzamosan, az át nem fedés betartásával). El kell döntenünk, hogy az input/kiinduló konfigurációból tologatásokkal el tudunk-e jutni egy célkonfigurációba. Azaz elérhető-e egy célkonfiguráció-halmaz egy eleme (mondjuk az egyik téglalapot egy adott pozícióba vihetjük-e)?

SLIDING-BLOCK-PUZZLE

SLIDING-BLOCK-PUZZLE

Inputja egy $n \times m$ táblázatban (mint alappályán) elhelyezett egymást át nem fedő téglalapok. A téglalapok a pályát nem fedik le teljesen, így lehetőség van tologatásukra (az alaptábla oldalaival párhuzamosan, az át nem fedés betartásával). El kell döntenünk, hogy az input/kiinduló konfigurációból tologatásokkal el tudunk-e jutni egy célkonfigurációba. Azaz elérhető-e egy célkonfiguráció-halmaz egy eleme (mondjuk az egyik téglalapot egy adott pozícióba vihetjük-e)?

SLIDING-BLOCK-PUZZLE

SLIDING-BLOCK-PUZZLE

Inputja egy $n \times m$ táblázatban (mint alappályán) elhelyezett egymást át nem fedő téglalapok. A téglalapok a pályát nem fedik le teljesen, így lehetőség van tologatásukra (az alaptábla oldalaival párhuzamosan, az át nem fedés betartásával). El kell döntenünk, hogy az input/kiinduló konfigurációból tologatásokkal el tudunk-e jutni egy célkonfigurációba. Azaz elérhető-e egy célkonfiguráció-halmaz egy eleme (mondjuk az egyik téglalapot egy adott pozícióba vihetjük-e)?

HAMILTON

HAMILTON

HAMILTON

HAMILTON probléma inputja egy gráf. El kell döntenünk, hogy van-e benne Hamilton-kör.

HAMILTON

HAMILTON

HAMILTON probléma inputja egy gráf. El kell döntenünk, hogy van-e benne Hamilton-kör.

Leírunk egy nem-determinisztikus TG-et: A tanúszalagon elvárhatjuk a csúcsok egy felsorolását. Ellenőriznünk kell, hogy az egymásutáni csúcsok szomszédosak a gráfban és az első, illetve utolsó csúcs is összekötött. Ellenőrizni kell azt az „ígéretet” is, hogy minden csúcsot pontosan egyszer soroltunk fel.

HAMILTON

HAMILTON

HAMILTON probléma inputja egy gráf. El kell döntenünk, hogy van-e benne Hamilton-kör.

Leírunk egy nem-determinisztikus TG-et: A tanúszalagon elvárhatjuk a csúcsok egy felsorolását. Ellenőriznünk kell, hogy az egymásutáni csúcsok szomszédosak a gráfban és az első, illetve utolsó csúcs is összekötött. Ellenőrizni kell azt az „ígéretet” is, hogy minden csúcsot pontosan egyszer soroltunk fel.

Teszteink nyilván polinomiális időben megvalósíthatók. Ha ezen tesztek mindegyike stimmel, akkor a tanú bizonyítja, hogy inputgráfunkban van Hamilton-kör. Másrészt nyilván minden Hamilton-körrel rendelkező gráfhoz található bizonyító tanú. Kaptuk, hogy

$$\text{HAMILTON} \in \mathcal{NP}.$$

HAMILTON (folytatás)

HAMILTON (folytatás)

coNP-beliséghez a Hamilton-kör hiányát kellene hatékonyan megindokolnunk.

HAMILTON (folytatás)

coNP-beliséghez a Hamilton-kör hiányát kellene hatékonyan megindokolnunk.

Könnyű egy polinomiális Turing-gépet tervezni, ami teszteli, hogy a tanúszalag tartalma egy U csúcshalmaz-e és $G - U$ komponenseinek száma nagyobb-e mint U elemszáma.

HAMILTON (folytatás)

coNP-beliséghez a Hamilton-kör hiányát kellene hatékonyan megindokolnunk.

Könnyű egy polinomiális Turing-gépet tervezni, ami teszteli, hogy a tanúszalag tartalma egy U csúcshalmaz-e és $G - U$ komponenseinek száma nagyobb-e mint U elemszáma.

Ha igen, akkor biztosak lehetünk, hogy gráfunkban nincs Hamilton-kör.

HAMILTON (folytatás)

coNP-beliséghez a Hamilton-kör hiányát kellene hatékonyan megindokolnunk.

Könnyű egy polinomiális Turing-gépet tervezni, ami teszteli, hogy a tanúszalag tartalma egy U csúcshalmaz-e és $G - U$ komponenseinek száma nagyobb-e mint U elemszáma.

Ha igen, akkor biztosak lehetünk, hogy gráfunkban nincs Hamilton-kör. Valóban, U elhagyása után a Hamilton-kör megmaradt ívei garantálnák, hogy $|U|$-nál nem több komponensünk van.

HAMILTON (folytatás)

coNP-beliséghez a Hamilton-kör hiányát kellene hatékonyan megindokolnunk.

Könnyű egy polinomiális Turing-gépet tervezni, ami teszteli, hogy a tanúszalag tartalma egy U csúcshalmaz-e és $G - U$ komponenseinek száma nagyobb-e mint U elemszáma.

Ha igen, akkor biztosak lehetünk, hogy gráfunkban nincs Hamilton-kör. Valóban, U elhagyása után a Hamilton-kör megmaradt ívei garantálnák, hogy $|U|$-nál nem több komponensünk van.

A fent leírt gép azonban NEM bioznyítja a HAMILTON nyelv $co\mathcal{NP}$beliségét.

HAMILTON (folytatás)

coNP-beliséghez a Hamilton-kör hiányát kellene hatékonyan megindokolnunk.

Könnyű egy polinomiális Turing-gépet tervezni, ami teszteli, hogy a tanúszalag tartalma egy U csúcshalmaz-e és $G - U$ komponenseinek száma nagyobb-e mint U elemszáma.

Ha igen, akkor biztosak lehetünk, hogy gráfunkban nincs Hamilton-kör. Valóban, U elhagyása után a Hamilton-kör megmaradt ívei garantálnák, hogy $|U|$-nál nem több komponensünk van.

A fent leírt gép azonban NEM bioznyítja a HAMILTON nyelv $co\mathcal{NP}$beliségét. Nem igaz, hogy Hamilton-kör hiányát ilyen módon biztos igazolni tudjuk.

HAMILTON (folytatás)

coNP-beliséghez a Hamilton-kör hiányát kellene hatékonyan megindokolnunk.

Könnyű egy polinomiális Turing-gépet tervezni, ami teszteli, hogy a tanúszalag tartalma egy U csúcshalmaz-e és $G - U$ komponenseinek száma nagyobb-e mint U elemszáma.

Ha igen, akkor biztosak lehetünk, hogy gráfunkban nincs Hamilton-kör. Valóban, U elhagyása után a Hamilton-kör megmaradt ívei garantálnák, hogy $|U|$-nál nem több komponensünk van.

A fent leírt gép azonban NEM bioznyítja a HAMILTON nyelv $co\mathcal{NP}$beliségét. Nem igaz, hogy Hamilton-kör hiányát ilyen módon biztos igazolni tudjuk. A Petersen-gráfban nincs Hamilton-kör.

HAMILTON (folytatás)

coNP-beliséghez a Hamilton-kör hiányát kellene hatékonyan megindokolnunk.

Könnyű egy polinomiális Turing-gépet tervezni, ami teszteli, hogy a tanúszalag tartalma egy U csúcshalmaz-e és $G - U$ komponenseinek száma nagyobb-e mint U elemszáma.

Ha igen, akkor biztosak lehetünk, hogy gráfunkban nincs Hamilton-kör. Valóban, U elhagyása után a Hamilton-kör megmaradt ívei garantálnák, hogy $|U|$-nál nem több komponensünk van.

A fent leírt gép azonban NEM bioznyítja a HAMILTON nyelv $co\mathcal{NP}$beliségét. Nem igaz, hogy Hamilton-kör hiányát ilyen módon biztos igazolni tudjuk. A Petersen-gráfban nincs Hamilton-kör. A fenti gép nem fogadná el.

HAMILTON (folytatás)

coNP-beliséghez a Hamilton-kör hiányát kellene hatékonyan megindokolnunk.

Könnyű egy polinomiális Turing-gépet tervezni, ami teszteli, hogy a tanúszalag tartalma egy U csúcshalmaz-e és $G - U$ komponenseinek száma nagyobb-e mint U elemszáma.

Ha igen, akkor biztosak lehetünk, hogy gráfunkban nincs Hamilton-kör. Valóban, U elhagyása után a Hamilton-kör megmaradt ívei garantálnák, hogy $|U|$-nál nem több komponensünk van.

A fent leírt gép azonban NEM bioznyítja a HAMILTON nyelv $co\mathcal{NP}$beliségét. Nem igaz, hogy Hamilton-kör hiányát ilyen módon biztos igazolni tudjuk. A Petersen-gráfban nincs Hamilton-kör. A fenti gép nem fogadná el.

Igazából nem ismert, hogy HAMILTON a $co\mathcal{NP}$ nyelvhez tartozik-e.

LP-TESZTELÉS

LP-TESZTELÉS

LP-TESZTELÉS probléma

Inputja egy $A \in \mathbb{Q}^{m \times n}$ mátrix és egy $b \in \mathbb{Q}^{m \times 1}$ (oszlop)vektor.

LP-TESZTELÉS

LP-TESZTELÉS probléma

Inputja egy $A \in \mathbb{Q}^{m \times n}$ mátrix és egy $b \in \mathbb{Q}^{m \times 1}$ (oszlop)vektor.

El kell döntenünk, hogy az $Ax = b$ egyenletrendszernek $(x = (x_1, x_2, \ldots, x_n)^T)$ van-e nem negatív megoldása.

LP-TESZTELÉS

LP-TESZTELÉS probléma

Inputja egy $A \in \mathbb{Q}^{m \times n}$ mátrix és egy $b \in \mathbb{Q}^{m \times 1}$ (oszlop)vektor.

El kell döntenünk, hogy az $Ax = b$ egyenletrendszernek $(x = (x_1, x_2, \ldots, x_n)^T)$ van-e nem negatív megoldása.

Valójában az egészek felett is dolgozhatunk.

LP-TESZTELÉS

LP-TESZTELÉS probléma

Inputja egy $A \in \mathbb{Q}^{m \times n}$ mátrix és egy $b \in \mathbb{Q}^{m \times 1}$ (oszlop)vektor.

El kell döntenünk, hogy az $Ax = b$ egyenletrendszernek $(x = (x_1, x_2, \ldots, x_n)^T)$ van-e nem negatív megoldása.

Valójában az egészek felett is dolgozhatunk. Az inputban szereplő számok nevezőinek legkisebb közös többszörösével megszorozhatjuk egyenleteinket.

LP-TESZTELÉS

LP-TESZTELÉS probléma

Inputja egy $A \in \mathbb{Q}^{m \times n}$ mátrix és egy $b \in \mathbb{Q}^{m \times 1}$ (oszlop)vektor.

El kell döntenünk, hogy az $Ax = b$ egyenletrendszernek $(x = (x_1, x_2, \ldots, x_n)^T)$ van-e nem negatív megoldása.

Valójában az egészek felett is dolgozhatunk. Az inputban szereplő számok nevezőinek legkisebb közös többszörösével megszorozhatjuk egyenleteinket. Az eredetivel ekvivalens egyenletrendszer együtthatói leírásának összhossza az eredeti inputméret polinomjával (négyzetével) becsülhető.

LP-TESZTELÉS probléma (folytatás)

LP-TESZTELÉS probléma (folytatás)

Az $\mathcal{NP}$-beliség egyszerűnek tűnik.

LP-TESZTELÉS probléma (folytatás)

Az $\mathcal{NP}$-beliség egyszerűnek tűnik. A tanúszalagra fel kell írni egy megoldást.

LP-TESZTELÉS probléma (folytatás)

Az $\mathcal{NP}$-beliség egyszerűnek tűnik. A tanúszalagra fel kell írni egy megoldást. A gép csak ellenőrzi ezt.

LP-TESZTELÉS probléma (folytatás)

Az $\mathcal{NP}$-beliség egyszerűnek tűnik. A tanúszalagra fel kell írni egy megoldást. A gép csak ellenőrzi ezt.

A probléma, hogy az ellenőrzés csak a tanúszámok méretében lesz polinomiális (szemben az inputszámokkal).

LP-TESZTELÉS probléma (folytatás)

Az $\mathcal{NP}$-beliség egyszerűnek tűnik. A tanúszalagra fel kell írni egy megoldást. A gép csak ellenőrzi ezt.

A probléma, hogy az ellenőrzés csak a tanúszámok méretében lesz polinomiális (szemben az inputszámokkal).

Azaz vigyáznunk kell, hogy tanúnk ne legyen lényegesen hosszabb az input méreténél. Ilyen tanú létezik. Ennek indoklását itt nem végezzük el.

LP-TESZTELÉS probléma (folytatás)

Az $\mathcal{NP}$-beliség egyszerűnek tűnik. A tanúszalagra fel kell írni egy megoldást. A gép csak ellenőrzi ezt.

A probléma, hogy az ellenőrzés csak a tanúszámok méretében lesz polinomiális (szemben az inputszámokkal).

Azaz vigyáznunk kell, hogy tanúnk ne legyen lényegesen hosszabb az input méreténél. Ilyen tanú létezik. Ennek indoklását itt nem végezzük el.

Tétel

$$\text{LP-TESZTELÉS} \in \mathcal{NP}.$$

LP-TESZTELÉS probléma (folytatás)

LP-TESZTELÉS probléma (folytatás)

Egy egyenletrendszer nem negatív számok körében való meg nem oldhatóságára ismertetünk egy módszert.

LP-TESZTELÉS probléma (folytatás)

Egy egyenletrendszer nem negatív számok körében való meg nem oldhatóságára ismertetünk egy módszert.

Az egyenleteink számszorosa, ezek összege a kiinduló rendszer egy következménye.

LP-TESZTELÉS probléma (folytatás)

Egy egyenletrendszer nem negatív számok körében való meg nem oldhatóságára ismertetünk egy módszert.

Az egyenleteink számszorosa, ezek összege a kiinduló rendszer egy következménye.

Ha ezt a következtetést úgy végezzük, hogy a bal oldalon szereplő kikombinált lineáris kifejezésben minden együttható nem negatív legyen, míg a jobb oldalon egy negatív szám adódjon, akkor nagyon transzparens lesz, hogy a következtett egyenletnek nincs nem negatív megoldása.

LP-TESZTELÉS probléma (folytatás)

Egy egyenletrendszer nem negatív számok körében való meg nem oldhatóságára ismertetünk egy módszert.

Az egyenleteink számszorosa, ezek összege a kiinduló rendszer egy következménye.

Ha ezt a következtetést úgy végezzük, hogy a bal oldalon szereplő kikombinált lineáris kifejezésben minden együttható nem negatív legyen, míg a jobb oldalon egy negatív szám adódjon, akkor nagyon transzparens lesz, hogy a következtett egyenletnek nincs nem negatív megoldása. Így az eredeti egyenletrendszernek sincs.

LP-TESZTELÉS probléma (folytatás)

Egy egyenletrendszer nem negatív számok körében való meg nem oldhatóságára ismertetünk egy módszert.

Az egyenleteink számszorosa, ezek összege a kiinduló rendszer egy következménye.

Ha ezt a következtetést úgy végezzük, hogy a bal oldalon szereplő kikombinált lineáris kifejezésben minden együttható nem negatív legyen, míg a jobb oldalon egy negatív szám adódjon, akkor nagyon transzparens lesz, hogy a következtett egyenletnek nincs nem negatív megoldása. Így az eredeti egyenletrendszernek sincs.

Az előzőekben nem ismertetett gondolatmenethez hasonlóan belátható, hogy a bizonyító következmények között olyan is van, ami kezelhető együtthatókkal kikombinálható.

LP-TESZTELÉS probléma (folytatás)

LP-TESZTELÉS probléma (folytatás)

Így a tanúszalagról leolvasható és tesztelhető polinomiális időben. A fent ismertetett stratégia akkor vezet $\mathcal{NP}$ algoritmushoz, ha igaz, hogy nem megoldható inputrendszer esetén ilyen bizonyítás található is rá.

LP-TESZTELÉS probléma (folytatás)

Így a tanúszalagról leolvasható és tesztelhető polinomiális időben. A fent ismertetett stratégia akkor vezet $\mathcal{NP}$ algoritmushoz, ha igaz, hogy nem megoldható inputrendszer esetén ilyen bizonyítás található is rá. Ez igaz, ez a jól-ismert Farkas-lemma.

LP-TESZTELÉS probléma (folytatás)

Így a tanúszalagról leolvasható és tesztelhető polinomiális időben. A fent ismertetett stratégia akkor vezet $\mathcal{NP}$ algoritmushoz, ha igaz, hogy nem megoldható inputrendszer esetén ilyen bizonyítás található is rá. Ez igaz, ez a jól-ismert Farkas-lemma.

Tehát

$$\text{LP-TESZTELÉS} \in co\mathcal{NP}.$$

LP-TESZTELÉS probléma (folytatás)

Jelenleg több olyan lineáris programozási algoritmus is van, ami polinomiális időben fut. Azaz

$$\text{LP-TESZTELÉS} \in \mathcal{P}.$$

LP-TESZTELÉS probléma (folytatás)

Jelenleg több olyan lineáris programozási algoritmus is van, ami polinomiális időben fut. Azaz

$$\text{LP-TESZTELÉS} \in \mathcal{P}.$$

• Megjegyezzük, hogy az LP-TESZTELÉS a lineáris programozás optimalizálási probléma egyik döntési változata.

LP-TESZTELÉS probléma (folytatás)

Jelenleg több olyan lineáris programozási algoritmus is van, ami polinomiális időben fut. Azaz

$$\text{LP-TESZTELÉS} \in \mathcal{P}.$$

- Megjegyezzük, hogy az LP-TESZTELÉS a lineáris programozás optimalizálási probléma egyik döntési változata.
- $\mathcal{NP}$-belisége klasszikus becsléseken alapul.

LP-TESZTELÉS probléma (folytatás)

Jelenleg több olyan lineáris programozási algoritmus is van, ami polinomiális időben fut. Azaz

$$\text{LP-TESZTELÉS} \in \mathcal{P}.$$

- Megjegyezzük, hogy az LP-TESZTELÉS a lineáris programozás optimalizálási probléma egyik döntési változata.
- $\mathcal{NP}$-belisége klasszikus becsléseken alapul.
- $co\mathcal{NP}$-belisége a Farkas-lemmán alapul, amit 1902-ben publikált Farkas Gyula.

LP-TESZTELÉS probléma (folytatás)

Jelenleg több olyan lineáris programozási algoritmus is van, ami polinomiális időben fut. Azaz

$$\text{LP-TESZTELÉS} \in \mathcal{P}.$$

• Megjegyezzük, hogy az LP-TESZTELÉS a lineáris programozás optimalizálási probléma egyik döntési változata.

• $\mathcal{NP}$-belisége klasszikus becsléseken alapul.

• $co\mathcal{NP}$-belisége a Farkas-lemmán alapul, amit 1902-ben publikált Farkas Gyula.

• A LP optimalizálás mind a mai napig ünnepelt szimplex algoritmusát 1947-ben jelent meg (Dantzig).

LP-TESZTELÉS probléma (folytatás)

Jelenleg több olyan lineáris programozási algoritmus is van, ami polinomiális időben fut. Azaz

$$\text{LP-TESZTELÉS} \in \mathcal{P}.$$

• Megjegyezzük, hogy az LP-TESZTELÉS a lineáris programozás optimalizálási probléma egyik döntési változata.

• $\mathcal{NP}$-belisége klasszikus becsléseken alapul.

• $co\mathcal{NP}$-belisége a Farkas-lemmán alapul, amit 1902-ben publikált Farkas Gyula.

• A LP optimalizálás mind a mai napig ünnepelt szimplex algoritmusát 1947-ben jelent meg (Dantzig).

• 1972-ben Klee és Minty bizonyította hogy az algoritmus nem polinomiális (valójában exponenciális futási idejű).

LP-TESZTELÉS probléma (folytatás)

Jelenleg több olyan lineáris programozási algoritmus is van, ami polinomiális időben fut. Azaz

$$\text{LP-TESZTELÉS} \in \mathcal{P}.$$

- Megjegyezzük, hogy az LP-TESZTELÉS a lineáris programozás optimalizálási probléma egyik döntési változata.
- $\mathcal{NP}$-belisége klasszikus becsléseken alapul.
- $co\mathcal{NP}$-belisége a Farkas-lemmán alapul, amit 1902-ben publikált Farkas Gyula.
- A LP optimalizálás mind a mai napig ünnepelt szimplex algoritmusát 1947-ben jelent meg (Dantzig).
- 1972-ben Klee és Minty bizonyította hogy az algoritmus nem polinomiális (valójában exponenciális futási idejű).
- Az első polinomiális algoritmust Kachian adta 1979-ben.

PRÍM-TESZTELÉS

PRÍM-TESZTELÉS

PRÍM-TESZTELÉS

A probléma inputja egy n pozitív egész (mondjuk 10-es számrendszerben kódolva). El kell döntenünk, hogy príme-e.

PRÍM-TESZTELÉS

PRÍM-TESZTELÉS

A probléma inputja egy n pozitív egész (mondjuk 10-es számrendszerben kódolva). El kell döntenünk, hogy príme-e.

A PRÍMTESZTELÉS-sel kapcsolatban az egyszerű feladat a nem prímség bizonyítása. Ehhez csak egy valódi osztót kell előhoznunk tanúként. Könnyű ellenőrizni az oszthatóságot (és a valódiságot is).

PRÍM-TESZTELÉS

PRÍM-TESZTELÉS

A probléma inputja egy n pozitív egész (mondjuk 10-es számrendszerben kódolva). El kell döntenünk, hogy príme-e.

A PRÍMTESZTELÉS-sel kapcsolatban az egyszerű feladat a nem prímség bizonyítása. Ehhez csak egy valódi osztót kell előhoznunk tanúként. Könnyű ellenőrizni az oszthatóságot (és a valódiságot is).

Kapjuk, hogy

$$\text{PRÍMTESZTELÉS} \in co\mathcal{NP}.$$

PRÍM-TESZTELÉS (folytatás)

PRÍM-TESZTELÉS (folytatás)

Pratt bizonyítási sémája (1975) mutatja, hogy

$$\text{PRÍMTESZTELÉS} \in \mathcal{NP}.$$

PRÍM-TESZTELÉS (folytatás)

Pratt bizonyítási sémája (1975) mutatja, hogy

$$\text{PRÍMTESZTELÉS} \in \mathcal{NP}.$$

Agrawal—Kayal—Saxena-prímteszt a következő tételhez vezet:

$$\text{PRÍMTESZTELÉS} \in \mathcal{P}.$$

TELJES-PÁROSÍTÁS-TESZTELÉS

TELJES-PÁROSÍTÁS-TESZTELÉS

TELJES-PÁROSÍTÁS-TESZTELÉS

Az input egy egyszerű gráf. El kell döntenünk, hogy az input tartalmaz-e teljes párosítást.

TELJES-PÁROSÍTÁS-TESZTELÉS

TELJES-PÁROSÍTÁS-TESZTELÉS

Az input egy egyszerű gráf. El kell döntenünk, hogy az input tartalmaz-e teljes párosítást.

Az input kódolását nem tárgyaljuk. Azonban azt megjegyezzük, hogy a fenti értelemben v, a csúcsszám is vehető az input méretének (kódja hossza helyett).

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

Először egy nemdeterminisztikus algoritmust írunk le. A nemdeterminizmus második értelmezését használjuk. Azaz egy tanúszalag tartalma segítségével döntünk az elfogadásról. A tanúszalag tartalma csúcspárok egy M halmaza lesz.

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

Először egy nemdeterminisztikus algoritmust írunk le. A nemdeterminizmus második értelmezését használjuk. Azaz egy tanúszalag tartalma segítségével döntünk az elfogadásról. A tanúszalag tartalma csúcspárok egy M halmaza lesz.

A T gép azt teszteli, hogy a csúcspárok éllel összekötött párok-e, és minden csúcs pontosan egy párban szerepel-e. Ha mindkétszer igen a válasz, akkor ELFOGAD állapotba kerülünk. Ha valamelyik teszten elbukik a tanú, akkor NEM-STIMMEL állapotba kerülünk.

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

Először egy nemdeterminisztikus algoritmust írunk le. A nemdeterminizmus második értelmezését használjuk. Azaz egy tanúszalag tartalma segítségével döntünk az elfogadásról. A tanúszalag tartalma csúcspárok egy M halmaza lesz.

A T gép azt teszteli, hogy a csúcspárok éllel összekötött párok-e, és minden csúcs pontosan egy párban szerepel-e. Ha mindkétszer igen a válasz, akkor ELFOGAD állapotba kerülünk. Ha valamelyik teszten elbukik a tanú, akkor NEM-STIMMEL állapotba kerülünk.

Egy teljes párosítás létezése esetén könnyű bizonyító tanút megadnunk. Ha nincs teljes párosítás, akkor mindegyik tanú elbukik.

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

Először egy nemdeterminisztikus algoritmust írunk le. A nemdeterminizmus második értelmezését használjuk. Azaz egy tanúszalag tartalma segítségével döntünk az elfogadásról. A tanúszalag tartalma csúcspárok egy M halmaza lesz.

A T gép azt teszteli, hogy a csúcspárok éllel összekötött párok-e, és minden csúcs pontosan egy párban szerepel-e. Ha mindkétszer igen a válasz, akkor ELFOGAD állapotba kerülünk. Ha valamelyik teszten elbukik a tanú, akkor NEM-STIMMEL állapotba kerülünk.

Egy teljes párosítás létezése esetén könnyű bizonyító tanút megadnunk. Ha nincs teljes párosítás, akkor mindegyik tanú elbukik.

A tesztek polinom időben könnyen elvégezhetők. Így kaptuk, hogy

$$\text{TELJES-PÁROSÍTÁS-TESZTELÉS} \in \mathcal{NP}.$$

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

Tutte-tétel ismeretében egyszerű dolgunk van ha teljes párosítás nem létét szeretnénk bizonyítani.

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

Tutte-tétel ismeretében egyszerű dolgunk van ha teljes párosítás nem létét szeretnénk bizonyítani.

A tanúszalag tartalma legyen egy T ponthalmaz. A gép az $\omega \equiv G$ gráf és $\tau \equiv T$ ponthalmaz esetében meghatározza $G - T$ komponenseit, megszámolja páratlan pontszámúakat és ezt a számot összehasonlítja $|T|$-vel.

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

Tutte-tétel ismeretében egyszerű dolgunk van ha teljes párosítás nem létét szeretnénk bizonyítani.

A tanúszalag tartalma legyen egy T ponthalmaz. A gép az $\omega \equiv G$ gráf és $\tau \equiv T$ ponthalmaz esetében meghatározza $G - T$ komponenseit, megszámolja páratlan pontszámúakat és ezt a számot összehasonlítja $|T|$-vel. Amennyiben T elemszáma kisebb a páratlan pontszámú komponensek számánál a gép ELFOGAD állapotba kerül (a komplementer nyelvhez definiáljuk a gépet; az elfogadás azt jelenti, hogy a komplementer nyelv eleme, azaz nincs benne teljes párosítás).

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

Tutte-tétel ismeretében egyszerű dolgunk van ha teljes párosítás nem létét szeretnénk bizonyítani.

A tanúszalag tartalma legyen egy T ponthalmaz. A gép az $\omega \equiv G$ gráf és $\tau \equiv T$ ponthalmaz esetében meghatározza $G - T$ komponenseit, megszámolja páratlan pontszámúakat és ezt a számot összehasonlítja $|T|$-vel. Amennyiben T elemszáma kisebb a páratlan pontszámú komponensek számánál a gép ELFOGAD állapotba kerül (a komplementer nyelvhez definiáljuk a gépet; az elfogadás azt jelenti, hogy a komplementer nyelv eleme, azaz nincs benne teljes párosítás). Gépünk minden más esetben NEM-STIMMEL állapotba kerül.

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

Tutte-tétel ismeretében egyszerű dolgunk van ha teljes párosítás nem létét szeretnénk bizonyítani.

A tanúszalag tartalma legyen egy T ponthalmaz. A gép az $\omega \equiv G$ gráf és $\tau \equiv T$ ponthalmaz esetében meghatározza $G - T$ komponenseit, megszámolja páratlan pontszámúakat és ezt a számot összehasonlítja $|T|$-vel. Amennyiben T elemszáma kisebb a páratlan pontszámú komponensek számánál a gép ELFOGAD állapotba kerül (a komplementer nyelvhez definiáljuk a gépet; az elfogadás azt jelenti, hogy a komplementer nyelv eleme, azaz nincs benne teljes párosítás). Gépünk minden más esetben NEM-STIMMEL állapotba kerül.

A gép polinomiális megvalósíthatóságának igazolása az olvasó feladata.

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

Az algoritmus korrektsége (G-ben akkor és csak akkor nincs teljes párosítás, ha alkalmas T tanú ezt bizonyítja) éppen Tutte-tételének állítása.

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

Az algoritmus korrektsége (G-ben akkor és csak akkor nincs teljes párosítás, ha alkalmas T tanú ezt bizonyítja) éppen Tutte-tételének állítása.

Így kapjuk a következőt

$$\text{TELJES-PÁROSÍTÁS-TESZTELÉS} \in co\mathcal{NP}.$$

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

Az Edmonds-algoritmus Turing-gép megvalósítása egy polinomiális algoritmus.

TELJES-PÁROSÍTÁS-TESZTELÉS (folytatás)

Az Edmonds-algoritmus Turing-gép megvalósítása egy polinomiális algoritmus.

Ez (az igen összetett) algoritmus az előző két eredménynél erősebb állításhoz vezet:

$$\text{TELJES-PÁROSÍTÁS-TESZTELÉS} \in \mathcal{P}.$$

IRÁNYÍTOTT-ELÉRHETŐSÉG

IRÁNYÍTOTT-ELÉRHETŐSÉG

IRÁNYÍTOTT-ELÉRHETŐSÉG

Adott $\overrightarrow{G}$ egyszerű irányított gráf és s, t két csúcsa. Döntsük el, hogy van-e irányított st séta $\overrightarrow{G}$-ben.

IRÁNYÍTOTT-ELÉRHETŐSÉG

IRÁNYÍTOTT-ELÉRHETŐSÉG

Adott $\overrightarrow{G}$ egyszerű irányított gráf és s, t két csúcsa. Döntsük el, hogy van-e irányított st séta $\overrightarrow{G}$-ben.

Természetesen az $(\overrightarrow{G}, s, t)$ inputot kódolnunk kell.
IRÁNYÍTOTT-ELÉRHETŐSÉG azon kódok halmaz, amelyek gráf komponense tartalmaz st sétát.

IRÁNYÍTOTT-ELÉRHETŐSÉG

IRÁNYÍTOTT-ELÉRHETŐSÉG

Adott $\overrightarrow{G}$ egyszerű irányított gráf és s, t két csúcsa. Döntsük el, hogy van-e irányított st séta $\overrightarrow{G}$-ben.

Természetesen az $(\overrightarrow{G}, s, t)$ inputot kódolnunk kell.
IRÁNYÍTOTT-ELÉRHETŐSÉG azon kódok halmaz, amelyek gráf komponense tartalmaz st sétát.

A szélességi keresés algoritmusa adja, hogy

$$\text{IRÁNYÍTOTT-ELÉRHETŐSÉG} \in \mathcal{P}.$$

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Az algoritmus, amit adunk, az egy nemdeterminisztikus algoritmus lesz.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Az algoritmus, amit adunk, az egy nemdeterminisztikus algoritmus lesz.

Azt is mondhatnánk, hogy egy eltévedt sétáló algoritmusa a $\overrightarrow{G}$ gráfban.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Az algoritmus, amit adunk, az egy nemdeterminisztikus algoritmus lesz.

Azt is mondhatnánk, hogy egy eltévedt sétáló algoritmusa a $\overrightarrow{G}$ gráfban. A sétálás folyamán azt nézzük, hogy t-ben vagyunk-e, illetve számoljuk, hány lépést tettünk eddig meg.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Az algoritmus, amit adunk, az egy nemdeterminisztikus algoritmus lesz.

Azt is mondhatnánk, hogy egy eltévedt sétáló algoritmusa a $\overrightarrow{G}$ gráfban. A sétálás folyamán azt nézzük, hogy t-ben vagyunk-e, illetve számoljuk, hány lépést tettünk eddig meg.

Ha elértük t-t, akkor ELFOGAD állapottal leállunk.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Az algoritmus, amit adunk, az egy nemdeterminisztikus algoritmus lesz.

Azt is mondhatnánk, hogy egy eltévedt sétáló algoritmusa a $\overrightarrow{G}$ gráfban. A sétálás folyamán azt nézzük, hogy t-ben vagyunk-e, illetve számoljuk, hány lépést tettünk eddig meg.

Ha elértük t-t, akkor ELFOGAD állapottal leállunk.

Ha nem értük el t-t, akkor megnézzük, hogy tettünk-e v lépést. Ha igen, akkor NEM-STIMMEL állapottal leállunk.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Az algoritmus, amit adunk, az egy nemdeterminisztikus algoritmus lesz.

Azt is mondhatnánk, hogy egy eltévedt sétáló algoritmusa a $\overrightarrow{G}$ gráfban. A sétálás folyamán azt nézzük, hogy t-ben vagyunk-e, illetve számoljuk, hány lépést tettünk eddig meg.

Ha elértük t-t, akkor ELFOGAD állapottal leállunk.

Ha nem értük el t-t, akkor megnézzük, hogy tettünk-e v lépést. Ha igen, akkor NEM-STIMMEL állapottal leállunk.

Ha még nem tettünk ennyi lépést, akkor nemdeterminisztikus lépésekkel felírunk egy csúcsot.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Az algoritmus, amit adunk, az egy nemdeterminisztikus algoritmus lesz.

Azt is mondhatnánk, hogy egy eltévedt sétáló algoritmusa a $\overrightarrow{G}$ gráfban. A sétálás folyamán azt nézzük, hogy t-ben vagyunk-e, illetve számoljuk, hány lépést tettünk eddig meg.

Ha elértük t-t, akkor ELFOGAD állapottal leállunk.

Ha nem értük el t-t, akkor megnézzük, hogy tettünk-e v lépést. Ha igen, akkor NEM-STIMMEL állapottal leállunk.

Ha még nem tettünk ennyi lépést, akkor nemdeterminisztikus lépésekkel felírunk egy csúcsot. Ellenőrizzük, hogy az előző csúcsból ide léphetünk-e egy élen keresztül.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Az algoritmus, amit adunk, az egy nemdeterminisztikus algoritmus lesz.

Azt is mondhatnánk, hogy egy eltévedt sétáló algoritmusa a $\overrightarrow{G}$ gráfban. A sétálás folyamán azt nézzük, hogy t-ben vagyunk-e, illetve számoljuk, hány lépést tettünk eddig meg.

Ha elértük t-t, akkor ELFOGAD állapottal leállunk.

Ha nem értük el t-t, akkor megnézzük, hogy tettünk-e v lépést. Ha igen, akkor NEM-STIMMEL állapottal leállunk.

Ha még nem tettünk ennyi lépést, akkor nemdeterminisztikus lépésekkel felírunk egy csúcsot. Ellenőrizzük, hogy az előző csúcsból ide léphetünk-e egy élen keresztül. Ha nem, akkor ismét NEM-STIMMEL állapotba jutunk.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Az algoritmus, amit adunk, az egy nemdeterminisztikus algoritmus lesz.

Azt is mondhatnánk, hogy egy eltévedt sétáló algoritmusa a $\overrightarrow{G}$ gráfban. A sétálás folyamán azt nézzük, hogy t-ben vagyunk-e, illetve számoljuk, hány lépést tettünk eddig meg.

Ha elértük t-t, akkor ELFOGAD állapottal leállunk.

Ha nem értük el t-t, akkor megnézzük, hogy tettünk-e v lépést. Ha igen, akkor NEM-STIMMEL állapottal leállunk.

Ha még nem tettünk ennyi lépést, akkor nemdeterminisztikus lépésekkel felírunk egy csúcsot. Ellenőrizzük, hogy az előző csúcsból ide léphetünk-e egy élen keresztül. Ha nem, akkor ismét NEM-STIMMEL állapotba jutunk. Ha igen, akkor az előző csúcsot töröljük (!).

IRÁNYÍTOTT-ELÉRHETŐSÉG

IRÁNYÍTOTT-ELÉRHETŐSÉG

Persze a törölt csúcs helyét a séta későbbi részére fenntartjuk.

IRÁNYÍTOTT-ELÉRHETŐSÉG

Persze a törölt csúcs helyét a séta későbbi részére fenntartjuk. Így elérjük, hogy a tárban a futás minden pillanatában legfeljebb két csúcs van és egy számláló, ami értéke legfeljebb v.

IRÁNYÍTOTT-ELÉRHETŐSÉG

Persze a törölt csúcs helyét a séta későbbi részére fenntartjuk. Így elérjük, hogy a tárban a futás minden pillanatában legfeljebb két csúcs van és egy számláló, ami értéke legfeljebb v. A szükséges tárigény $\mathcal{O}(\log v)$.

IRÁNYÍTOTT-ELÉRHETŐSÉG

Persze a törölt csúcs helyét a séta későbbi részére fenntartjuk. Így elérjük, hogy a tárban a futás minden pillanatában legfeljebb két csúcs van és egy számláló, ami értéke legfeljebb v. A szükséges tárigény $\mathcal{O}(\log v)$.

Így kaptuk, hogy

$$\text{IRÁNYÍTOTT-ELÉRHETŐSÉG} \in \mathcal{NL}.$$

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Egy újabb determinisztikus algoritmust adunk, aminek tárfelhasználása lesz nagyon takarékos:

$\overrightarrow{st}$-IRÁNYÍTOTT-ELÉRHETŐSÉG

$$\in \cup_{\alpha\in\mathbb{N}}\mathcal{SPACE}(\alpha \log_2^2 n) = \mathcal{SPACE}(\log^2 n).$$

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Egy újabb determinisztikus algoritmust adunk, aminek tárfelhasználása lesz nagyon takarékos:

$$\overrightarrow{st}\text{-IRÁNYÍTOTT-ELÉRHETŐSÉG}$$
$$\in \cup_{\alpha\in\mathbb{N}}\mathcal{SPACE}(\alpha \log_2^2 n) = \mathcal{SPACE}(\log^2 n).$$

Ezt a következő rekurzív algoritmus bizonyítja.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-algoritmus:
KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG$(x, y, 2^\ell)$

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-algoritmus: KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG($x, y, 2^\ell$)

// Adott x és y csúcsok esetén teszteli, hogy van-e köztük legfeljebb 2^ℓ lépéses séta. A sétára gondolhatunk úgy, mint egy „lusta” séta. Minden lépésnél két lehetőségünk van: vagy egy szomszédba mozgunk, vagy maradunk. Lusta sétánál feltehetjük, hogy a hossz pontosan 2^ℓ.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-algoritmus: KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG($x, y, 2^\ell$)

// Adott x és y csúcsok esetén teszteli, hogy van-e köztük legfeljebb 2^ℓ lépéses séta. A sétára gondolhatunk úgy, mint egy „lusta” séta. Minden lépésnél két lehetőségünk van: vagy egy szomszédba mozgunk, vagy maradunk. Lusta sétánál feltehetjük, hogy a hossz pontosan 2^ℓ.

Ha $\ell = 0$, akkor teszteljük, hogy $x = y$ vagy $\overrightarrow{xy}$ egy él. Ha a teszt sikerül, akkor ELFOGAD állapottal, különben ELVET állapottal leállunk.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-algoritmus: KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG($x, y, 2^\ell$)

// Adott x és y csúcsok esetén teszteli, hogy van-e köztük legfeljebb 2^ℓ lépéses séta. A sétára gondolhatunk úgy, mint egy „lusta” séta. Minden lépésnél két lehetőségünk van: vagy egy szomszédba mozgunk, vagy maradunk. Lusta sétánál feltehetjük, hogy a hossz pontosan 2^ℓ.

Ha $\ell = 0$, akkor teszteljük, hogy $x = y$ vagy $\overrightarrow{xy}$ egy él. Ha a teszt sikerül, akkor ELFOGAD állapottal, különben ELVET állapottal leállunk.

Ha $\ell > 0$, akkor Összes $k \in V$ esetén

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-algoritmus:
KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG($x, y, 2^\ell$)

// Adott x és y csúcsok esetén teszteli, hogy van-e köztük legfeljebb 2^ℓ lépéses séta. A sétára gondolhatunk úgy, mint egy „lusta" séta. Minden lépésnél két lehetőségünk van: vagy egy szomszédba mozgunk, vagy maradunk. Lusta sétánál feltehetjük, hogy a hossz pontosan 2^ℓ.

Ha $\ell = 0$, akkor teszteljük, hogy $x = y$ vagy $\overrightarrow{xy}$ egy él. Ha a teszt sikerül, akkor ELFOGAD állapottal, különben ELVET állapottal leállunk.

Ha $\ell > 0$, akkor Összes $k \in V$ esetén
// k a lusta séta középső pontja, azaz x-ből k-ba $2^{\ell-1}$ lusta lépés vezet és k-ból y-ba is $2^{\ell-1}$ lusta lépés vezet. Az összes lehetőséget végigpróbáljuk.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-algoritmus (folytatás)

(1) KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG$(x, k, 2^{\ell-1})$

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-algoritmus (folytatás)

(1) KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG$(x, k, 2^{\ell-1})$
ha NEM, akkor következő k és vissza (1)-hez

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-algoritmus (folytatás)

(1) KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG$(x, k, 2^{\ell-1})$
ha NEM, akkor következő k és vissza (1)-hez
ha NEM, és nincs következő k (V kimerült) akkor ELVET.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-algoritmus (folytatás)

(1) KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG$(x, k, 2^{\ell-1})$

ha NEM, akkor következő k és vissza (1)-hez

ha NEM, és nincs következő k (V kimerült) akkor ELVET.

ha IGEN, akkor

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-algoritmus (folytatás)

(1) KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG$(x, k, 2^{\ell-1})$
ha NEM, akkor következő k és vissza (1)-hez
ha NEM, és nincs következő k (V kimerült) akkor ELVET.
ha IGEN, akkor
(2) KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG$(k, y, 2^{\ell-1})$

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-algoritmus (folytatás)

(1) KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG$(x, k, 2^{\ell-1})$
ha NEM, akkor következő k és vissza (1)-hez
ha NEM, és nincs következő k (V kimerült) akkor ELVET.
ha IGEN, akkor

(2) KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG$(k, y, 2^{\ell-1})$
ha IGEN, akkor ELFOGAD állapot és leáll

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-algoritmus (folytatás)

(1) KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG$(x, k, 2^{\ell-1})$
- ha NEM, akkor következő k és vissza (1)-hez
- ha NEM, és nincs következő k (V kimerült) akkor ELVET.
- ha IGEN, akkor

(2) KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG$(k, y, 2^{\ell-1})$
- ha IGEN, akkor ELFOGAD állapot és leáll
- ha NEM, akkor következő k és vissza (1)-hez

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-algoritmus (folytatás)

(1) KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG$(x, k, 2^{\ell-1})$
- ha NEM, akkor következő k és vissza (1)-hez
- ha NEM, és nincs következő k (V kimerült) akkor ELVET.
- ha IGEN, akkor

(2) KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG$(k, y, 2^{\ell-1})$
- ha IGEN, akkor ELFOGAD állapot és leáll
- ha NEM, akkor következő k és vissza (1)-hez
- ha NEM, és nincs következő k (V kimerült) akkor vissza (1) NEM ágára.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-algoritmus (folytatás)

(1) KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG$(x, k, 2^{\ell-1})$
  ha NEM, akkor következő k és vissza (1)-hez
  ha NEM, és nincs következő k (V kimerült) akkor ELVET.
  ha IGEN, akkor

(2) KORLÁTOZOTT-IRÁNYÍTOTT-ELÉRHETŐSÉG$(k, y, 2^{\ell-1})$
  ha IGEN, akkor ELFOGAD állapot és leáll
  ha NEM, akkor következő k és vissza (1)-hez
  ha NEM, és nincs következő k (V kimerült) akkor vissza (1) NEM ágára.

A fenti algoritmus $\ell = \lceil \log_2 |V(G)| \rceil$ paraméterrel futtatva megoldja az elérhetőséget. Az algoritmust Savitch Turing-gépen implementálta tár-takarékos módon.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-tétel

A fenti rekurzív algoritmus Turing-gépen megvalósítható úgy, hogy minden konfigurációban a munkaszalagon legfeljebb ℓ (a rekurzió mélysége sok) darab blokkot használunk, ahol egy blokk hossza $\mathcal{O}(\log |V|)$ (véges sok csúcs tárolására alkalmas hely).

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

Savitch-tétel

A fenti rekurzív algoritmus Turing-gépen megvalósítható úgy, hogy minden konfigurációban a munkaszalagon legfeljebb ℓ (a rekurzió mélysége sok) darab blokkot használunk, ahol egy blokk hossza $\mathcal{O}(\log |V|)$ (véges sok csúcs tárolására alkalmas hely).

A pontos megvalósításhoz/implementációhoz csak ötleteket adunk:

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

- A rekurzióban felmerülő kérdéseket struktúráját egy fával reprezentálhatjuk.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

- A rekurzióban felmerülő kérdéseket struktúráját egy fával reprezentálhatjuk.
- A fa gyökere az IRÁNYÍTOTT-ELÉRHETŐSÉG probléma, azaz az, hogy van-e 2^ℓ hosszú lusta séta?

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

- A rekurzióban felmerülő kérdéseket struktúráját egy fával reprezentálhatjuk.
- A fa gyökere az IRÁNYÍTOTT-ELÉRHETŐSÉG probléma, azaz az, hogy van-e 2^ℓ hosszú lusta séta?
- Minden p=(u-ból v-be vezet-e 2^ℓ hosszú lusta séta) probléma két részfeladatra bomlik.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

- A rekurzióban felmerülő kérdéseket struktúráját egy fával reprezentálhatjuk.
- A fa gyökere az IRÁNYÍTOTT-ELÉRHETŐSÉG probléma, azaz az, hogy van-e 2^ℓ hosszú lusta séta?
- Minden p=(u-ból v-be vezet-e 2^ℓ hosszú lusta séta) probléma két részfeladatra bomlik.
- Egy középső w csúcsra $p_{bal}(w)$= (vezet-e u-ból w-be $2^{\ell-1}$ hosszú lusta séta),

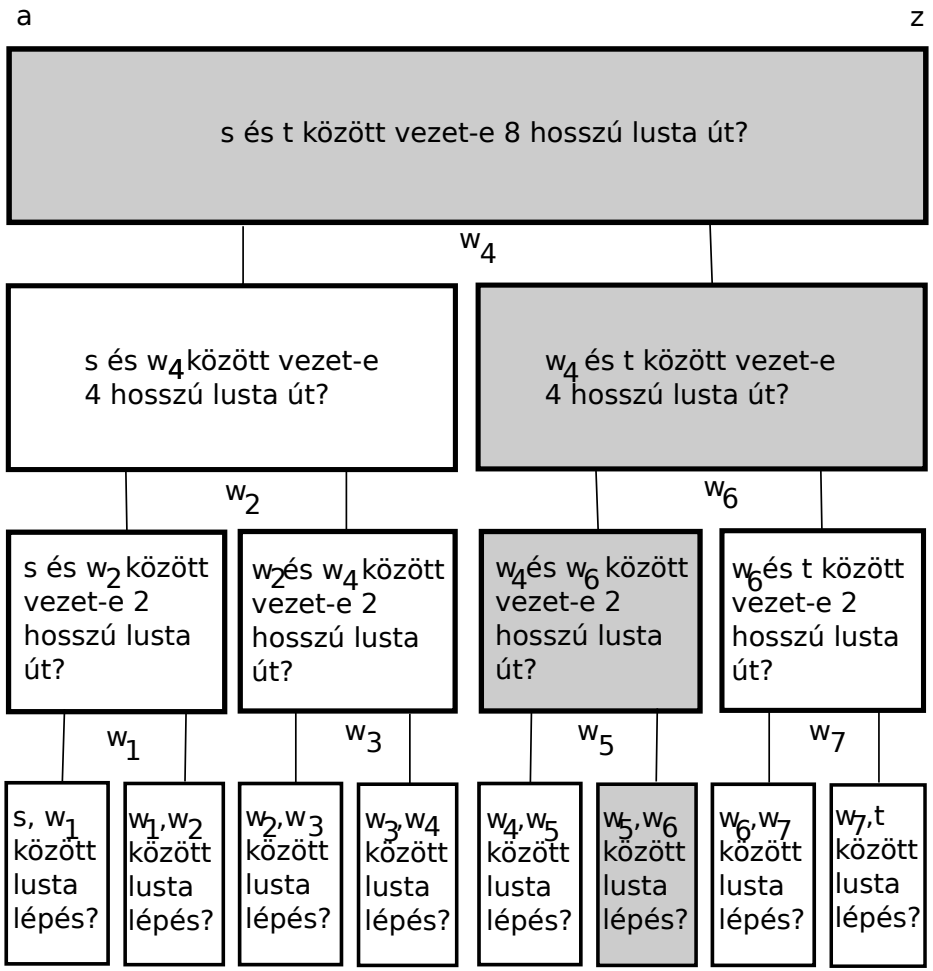
IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

- A rekurzióban felmerülő kérdéseket struktúráját egy fával reprezentálhatjuk.
- A fa gyökere az IRÁNYÍTOTT-ELÉRHETŐSÉG probléma, azaz az, hogy van-e 2^ℓ hosszú lusta séta?
- Minden p=(u-ból v-be vezet-e 2^ℓ hosszú lusta séta) probléma két részfeladatra bomlik.
- Egy középső w csúcsra $p_{bal}(w)$= (vezet-e u-ból w-be $2^{\ell-1}$ hosszú lusta séta), illetve $p_{jobb}(w)$= (vezet-e w-ból v-be $2^{\ell-1}$ hosszú lusta séta) a p probléma két részfeladata.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

- A rekurzióban felmerülő kérdéseket struktúráját egy fával reprezentálhatjuk.
- A fa gyökere az IRÁNYÍTOTT-ELÉRHETŐSÉG probléma, azaz az, hogy van-e 2^ℓ hosszú lusta séta?
- Minden p=(u-ból v-be vezet-e 2^ℓ hosszú lusta séta) probléma két részfeladatra bomlik.
- Egy középső w csúcsra $p_{bal}(w)$= (vezet-e u-ból w-be $2^{\ell-1}$ hosszú lusta séta), illetve $p_{jobb}(w)$= (vezet-e w-ból v-be $2^{\ell-1}$ hosszú lusta séta) a p probléma két részfeladata.
- A két feladat egymás *testvére*.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)



A munkaszalag tartalma mindig egy feladat (csúcs a fában) a gyökérhez vezetett úttal együtt. Egy példát kiemeltünk sötétítéssel.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

A Savitch-implementáció lényege, hogy az ábrán lévő út leírása belefér a tétel által igényelt memóriába.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

A Savitch-implementáció lényege, hogy az ábrán lévő út leírása belefér a tétel által igényelt memóriába.

Továbbá az út update-elése megoldható egy Turing-géppel.

IRÁNYÍTOTT-ELÉRHETŐSÉG (folytatás)

A Savitch-implementáció lényege, hogy az ábrán lévő út leírása belefér a tétel által igényelt memóriába.

Továbbá az út update-elése megoldható egy Turing-géppel.

A részletek teljes kidolgozása meghaladja a kurzus kereteit.

Vége van!

Köszönöm a figyelmet!