ALGORITMUSOK ÉS BONYOLULTSÁGELMÉLET

Matematika MSc hallgatók számára

4. Előadás

Előadó: Hajnal Péter
Jegyzetelő: Nemes Anna 2011. február 22.

Emlékeztető. Az eddigi nylevosztályokról a következő tartalmazások nyilvánvalók:

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

Az $\mathcal{L}, \mathcal{P}, \mathcal{EXP}$ nyelvosztályok determinisztikus gépekhez kapcsolódnak, ebből adódóan zártak a komplementerképzésre. Az $\mathcal{NL}, \mathcal{NP}, \mathcal{NPSPACE}, \mathcal{NEXP}$ osztályok nem-determinisztikus nyelvcsaládok (ezt hangsúlyozza a kezdő $\mathcal{N}$ betű). Ezek esetén a *co* $\mathcal{NL}$, *co* $\mathcal{NP}$, *co* $\mathcal{NPSPACE}$, *co* $\mathcal{NEXP}$ osztályok érdekesek.

Definíció.

$$\mathcal{TIME}(t(n)) = \{L : \text{létezik } L\text{-et elfogadó } T \text{ Turing-gép, amelynek futási ideje minden } \omega\text{-n legfeljebb } t(|\omega|)\}.$$

$$\mathcal{SPACE}(s(n))) = \{L : \text{létezik } L\text{-et elfogadó } T \text{ Turing-gép, amelynek tár igénye minden } \omega\text{-n legfeljebb } s(|\omega|)\}.$$

1. További tartalmazások bonyolultsági osztályok között

Észrevétel 0. $\mathcal{TIME}(t(n)) \subset \mathcal{SPACE}(t(n))$.

Észrevétel 1. $\mathcal{NTIME}(t(n)) \subset \mathcal{SPACE}(t(n))$, ahol $t(n)$ szép időfüggvény.

Bizonyítás. Legyen $L \in \mathcal{NTIME}(t(n))$. Ekkor megadható ezt bizonyító T tanúszalagos Turing-gép.

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

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.

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)

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≡ELVET állapottal ér véget, vagy letelik az idő/kifutunk a $t(n)$ időből. Ez utóbbit is a tesztel tanú ELVET-éseként fogjuk fel.

Ha a szimuláció ELFOGAD álapotba jutott, akkor mi is ELFOGADjuk az inputot, $\widetilde{T}$ is leáll. Ha ELVET 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 tanúkezdetet teszteltük, akkor KIMERÜLT≡ELVET állpottal leállunk.

1. Állítás. *(i) $\widetilde{T}$ L-et számolja ki.*

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

Mindkét rész egyszerűen adódik az előzőekből. Ezzel az észrevételt igazoltuk. ■

Észrevétel 2. $\mathcal{SPACE}(s(n)) \subseteq \cup_{c\in\mathbb{N}}\mathcal{TIME}(c^{s(n)+\log(n+1)})$, ahol $s(n))$ szép tárfüggvény.

Bizonyítás. Legyen $L \in \mathcal{SPACE}(s(n))$. Ekkor megadható olyan T Turing-gép, amely eldönti L-et (speciálisan minden $\omega \in L$-en megáll), és a tárigénye legfeljebb $s(n)$.

Legyen $\kappa_0(\omega) \to \kappa_1(\omega) \to \kappa_2(\omega) \to \ldots \to \kappa_\ell(\omega)$ a futás ω-n. Azaz ez egy $\ell \geq 1$ hosszú, véges konfigurációsorozat, ahol az első konfiguráció ($\kappa_0(\omega)$) a kiinduló konfiguráció (ebben az állapot START) és az utolsó állapot ($\kappa_\ell(\omega)$) az első olyan konfigutáció a futás során, amleyben az állapot ELFOGAD/ELVET.

Könnyű látni, hogy a futás során nem ismétlődhet konfiguráció, azaz $i \neq j$ esetén $\kappa_i \neq \kappa_j$ teljesül. Valóban minden konfiguráció egyértelműen meghatározza a rákövetkezőt, így ismétlődés egy végtelen, periodikus konfigurációsorozathoz vezetne.

Hányféle konfiguráció léphet fel a fenti sorozatban rögzített ω esetén? Legyen $|\omega| = n$. Egy felső becslés a kérdésre adandó válaszra (α_T, β_T konstansok T-től függnek):

$$(n+2)\cdot|S|\cdot(s(n))+1)\cdot|\Gamma|^{s(n))} \leq \alpha_T(n+1)\beta_T^{s(n)} = \beta_T^{s(n)+\log(n+1)},$$

hiszen az input szem helyzete $n+2$-féle, a munka szem helyzete $s(n)+1$-féle, az input szalag tartalma $|\Gamma|^{s(n)}$-féle, az állapot $|S|$-féle lehet.

Összefoglalva: Ha a futási idő ${\beta_T}^{s(n))+\log n}$-nél hosszabb lenne, akkor a futás során a konfigurációk ismétlődnének, így a futás végtelen lenne.

Tudjuk, hogy nincs így. Tehát kaptuk, hogy T időigénye automatikusan megfelel az észrevételben szereplőkkel. ■

Észrevétel 3. $\mathcal{NSPACE}(s(n)) \subseteq \bigcup_{c\in\mathbb{N}} \mathcal{TIME}(c^{s(n))+\log(n+1)})$, ahol $s(n)$ szép tárfüggvény.

Bizonyítás. Legyen $L \in \mathcal{NSPACE}(s(n))$. Ekkor megadható T (I. értelemben vett) nem-determinisztikus Turing-gép, vagyis az átmeneti függvény nem-determinisztikus, a futás „szétágazó" lehet.

Definíció. 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 takartuk le a (teljes) konfigurációból. A redukált konfigurációk halmaza legyen V. Ekkor

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

Legyen v speciális eleme A, ami a $\kappa_0(\omega)$ kezdő konfiguráció redukáltja. T-ről feltehető, hogy leálláskor az input- illetve a munkafej a szalag elejére áll, továbbá a munkaszalag első $s(n)$ karaktere üres. Így a leállás, ami két lehetséges állapotnak felel meg, az két lehetséges redukált konfigurációnak felel meg. Legyen $Z^+ \equiv$ELFOGAD, illetve $Z^- \equiv$ELVET a két leálló állapotnak megfelelő két redukált konfiguráció.

Definíció. Legyen T egy I. nemdeterminisztikus Turing-gép és ω egy inputja. Ekkor $\overrightarrow{G}_{\omega,T}$ a $(T,\omega$-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 $u(\omega)$ konfiguráció után az átmeneti függvény megengedi a $v(\omega)$ konfigurációt. Ahol egy r redukált konfiguráció esetén $r(\omega)$ az a konfiguráció, amit r-ből kapunk ω-nak az inputszalagra írásával.

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

A következő állítás a fenti definíciók megértése után nyilvánvaló.

2. Állítás. *Az $\omega \in L$ pontosan akkor teljesül, ha $\overrightarrow{G}_{\omega,T}$-ben létezik AZ^+ irányított út.*

Ezek után leírunk egy $\widetilde{T}$ determinisztikus gépet.

Ez első fázisban felírja a $\overrightarrow{G}_{\omega,T}$ gráf kódját. Majd gráfelméleti algoritmusokkal teszteli, hogy vann-e benne AZ^+ irányított út. Például szélességi vagy mélységi keresést hajt végre. A kód felírása arányos a hosszával, a keresési algoritmus polinomiális a gráf kódjában. A szükséges idő az észrevétel által megígért korlát alatt marad. ■

2. Problémak redukciói

Definíció. ELÉRHETŐSÉG=$\{(\overrightarrow{G}, s, t), s, t \in v:$ létezik $\overrightarrow{st}$ út$\} \subseteq \Sigma^*$.

Az előző bizonyítás lényege az volt, hogy $\omega \in L$ akkor és csak akkor, ha $(\overrightarrow{G}_{\omega,T}, A, Z^+) \in$ ELÉRHETŐSÉG.

Definíció. Legyen $L, \widehat{L} \subset \Sigma^*$ két nyelv és $\mathcal{C}$ egy bonyolultsági osztály. L redukálható $\widehat{L}$-ra, jelben: $L \preccurlyeq_{\mathcal{C}}^{K} \widehat{L}$, ha létezik A kiszámítható Turing-gép, hogy

(i) A egy $\mathcal{C}$ komplexitású gép/eljárás,

(ii) $\omega \in L$ pontosan akkor, ha $A(\omega) \in \widehat{L}$.

A bevezetett reláció olvasata: Az $\widehat{L}$ nyelv eldöntési feladata „legalább olyan nehéz", mint az L-é „modulo" $\mathcal{C}$.

A felső index K betűje Karp nevéből ered, aki ezt a fajta redukciót (amit Karp-redukciónak neveznek) először használta intenzíven. Ebben a kurzusban legtöbbször ilyen redukciót látunk. Legtöbbször le is hagyjuk a felső indexet. A teljessség kedvéért megemlítünk egy másik fajta redukciót. Ezt Turing nevéhez fűzik.

Definíció. Legyen $L, \widehat{L} \subseteq \Sigma^*$ két nyelv és $\mathcal{C}$ egy bonyolultsági osztály.

$L \preccurlyeq_{\mathcal{C}}^{T} \widehat{L}$ pontosan akkor, ha megadható A eldöntő Turing-gép, amelynek van egy extra kérdésszalagja. Erre csak írhat a gép (nincs szem a szalag felett, a kéz csak jobbra mozogva írhat). Az írott karakterek $\widehat{\Sigma}$ elemei, azaz a $\widehat{L}$ nyelv ábécéjének elemei és egy speciális '?' jel. A kérdésszalagra a ? jel feírása egyben a ? speciális állapotba való kerülést is jelenti. Ekkor a gép meg tudja, hogy a kérdés szalag eleje, illetve előző kérdésnél leírt '?' jele utáni karaktersorozat $\widehat{L}$ eleme-e. Ezen információ megszerzése a következőkonfigurációban megtörténik, azaz egy lépésnyi az „ára". A gép az L nyelvet dönti el, bonyolultsága $\mathcal{C}$-beli.

A Karp redukció nagyon speciális Turing-redukció. Szokásos számolás után egyetlen kérdés hangozhat el az $\widehat{L}$ nyelvhez tartozásról. A kérdésre adott válasz egyben a kiszámított bit is. A Turing-redukció nyilván sokkal erősebb fogalom. Az $\widehat{L}$-ra úgy gondolhatunk, mint egy szubrutin. A redukció lényege, hogy a $\widehat{L}$ szubrutin felhasználásával L hatékonyan eldönthető.

Példa. Legyen L egy tetszőleges $\mathcal{NSPACE}(s(n))$-beli nyelv, ahol $s(n)$ szép tárfüggvény. Ekkor

$$L \preccurlyeq_{\mathcal{C}}^{K} \text{ELÉRHETŐSÉG}$$

ahol $\mathcal{C} = \mathcal{SPACE}(\mathcal{O}(s(n) + \log(n+1))) = \cup_{\alpha \in \mathbb{N}} \mathcal{SPACE}(\alpha \cdot s(n) + \log(n+1))$.

Valóban: A korábbiakban láttuk, hogy ω inputhoz $(\overrightarrow{G}_{\omega,T}, A, Z^+)$ hármas rendelhető úgy, hogy $\omega \in L$ akkor és csak akkor teljesüljön, ha $(\overrightarrow{G}_{\omega,T}, A, Z^+) \in$ ELÉRHETŐSÉG.

Csupán azt kell ellenőrizni, hogy $(\overrightarrow{G}_{\omega,T}, A, Z^+)$ „legyártása" az adott tár korlát mellett megtehető. Ehhez egy gépet kell megadnunk. Ez kiszámol egy gráf és két csúcsának a kódját. Ehhez kell egy outputszalagra. A feltétel, hogy a munkaszalagon jóval kisebb terület áll rendelkezésünkre mint a kiszámítandó gráf kódja. Az alábbiakban ennek egy lehetséges megoldását vázoljuk:

A munkaszalag tartalmát kettéosztjuk a lehetséges csúcsok, illetve a lehetséges szomszédok helyére. A redukált konfigurációk gráfjának egy csúcsának kódolásához

$\log_2 |V|$ hely kell. Ez $\mathcal{O}(s(n) + \log(n+1))$ darab mező. (Azaz az előírt tárméret konstans sok csúcs tárolására elegendő.)

A lehetséges csúcsok számára fenntartott helyen felsoroljuk az összes ott elférő kódot. Mindegyikről megnézzük, hogy valójában csúcsot kódol-e. Ha igen, akkor felírjuk az outputszalagra, majd egy ':' rakunk. Ide kell írnunk a ki-szomszédok sorozatát. Ehhez a lehetséges szomszédok helyén soroljuk fel a lehetséges kódokat. Minden olyan kódnál ami csúcsot kódol meg kell néznünk, hogy az átmeneti függvény megengedi-e, hogy a lehetséges csúcsok helyére írt redukált konfigurációból az ω-val kiegészítve kapott konfiguráció olyan-e, hogy átmehetünk-e a lehetséges szomszédok helyén leírt redukált konfigurációnak ω-val való kiterjesztésével kapott konfigurációba. ω az inputszalagon, a két redukált konfiguráció elfér a munkaszalagon. A kért információ nagyon egyszerű könnyen kiolvasható külön tárigény nélkül. Ha a lehetséges szomszéd valóban gráf-szomszéd, akkor az outputszalagra átmásoljuk.

A és Z^+ kódjának felírása az outputszalagra szintén könnyen megoldható.

Végül megemlítünk egy fontos tulajdonságát a redukcióknak.

3. Lemma.

(i) $L \preccurlyeq_{\mathcal{P}} \widehat{L}$ *és* $\widehat{L} \in \mathcal{P}$, *akkor* $L \in \mathcal{P}$.

(ii) $L \preccurlyeq^K_{\mathcal{L}} \widehat{L}$ *és* $\widehat{L} \in \mathcal{L}$, *akkor* $L \in \mathcal{L}$.

Bizonyítás. (i) Tekintsünk egy A Turing-gépet, amely az L-ről $\widehat{L}$-ra történő redukciót végzi, valamint $\widehat{A}$-ot, amely a $\widehat{L}$ nyelvehz tartozási feladatot dönti el $\mathcal{P}$-ben. Legyen adott az ω input.

Ekkor végezzük el a

$$\omega \to A(\omega) \in \Sigma^{p(n)} \to \widehat{A}(A(\omega)) \in \Sigma^{q(p(n))}$$

számolást. Az elsőidő igényét az n inputméretben egy p polinom korlátozza. A leghosszabb input, amit kiszámolhatunk $\Sigma^{p(n)}$-ba esik. A második lépés időigényét az inputméretben egy q polinom korlátozza. Az összidő $(p + q \circ p)(n)$, ami n egy polinomiális függvénye.

A két algoritmus együttese a Karp-redukció fogalma alapján éppen az L nyelvet dönti el, vagyis L is eldönthető polinom időben.

(ii) Tekintsünk egy A Turing-gépet, amely az L-ről $\widehat{L}$-ra történő redukciót végzi és egy $\widehat{A}$-ot, amely $\widehat{L}$ nyelvhez tartozás problémáját dönti el $\mathcal{L}$-ben.

Ismét a két gépet szeretnénk kombinálni. Azonban A outputszalagja (egyben $\widehat{L}$ inputszalagja) nem fér meg a logaritmikus tár korláttal. Nincs is rá szükségünk. A két gép közül kezdjük el $\widehat{A}$ szimulációját. Egy munkaszalagon tároljuk az input szem pozícióját. Amikor $\widehat{A}$-nak szüksége van egy karakterre az inputszalagról, akkor $\widehat{A}$ szimulációja leáll, „PAUSE" állapotba kerül (a szükséges információk eltárolásával) és kezdjük meg az A gép futását. Addig futtatjuk míg ki nem írja azt a karaktert, amelyre szükségünk van. Ez a futtatás outputszalag nélkül történik. Azaz a kiszámolt karakterek elvesznek. Csak az éppen olvasott örződik meg $\widehat{A}$ további futása érdekében (ezért tároljuk $\widehat{A}$ futása közben hol is áll az input szem). A szimulációjához használt tárat letöröljük. Hiszen újabb $A(\omega)$-beli karakter olvasasához szükségünk lesz erre a helyre (hogy kiszámoljuk a kért karaktert, de nem arra, hogy a korábbi karaktereket megőrizzük). A PAUSE állpotból visszatérünk és folytatjuk $\widehat{A}$ futtatását, amíg új inputkarakterre nincs szükségünk. ■

A fenti bizonyítás gondolatmenete elvezet a következő lemmához is:

4. Lemma. *$\preccurlyeq_{\mathcal{L}}$ és $\preccurlyeq_{\mathcal{P}}$ tranzitív.*