ALGORITMUSOK ÉS BONYOLULTSÁGELMÉLET

Matematika MSc hallgatók számára

1. Előadás

Előadó: Hajnal Péter
Jegyzetelő: Hajnal Péter 2010. február 1.

Az algoritmus naív fogalma

Az algoritmus egy eljárás, ami az adatok megkapása után egy jól definiált lépéssort elvégezve megadja a probléma megoldását.

Az algoritmus fogalmával együtt kialakult egy az algoritmusokkal kapcsolatos nyelvezet is. Az adatokat *inputnak*, az eredmény *outputnak* nevezzük. Ha adott inputon az algoritmus utasításait követve végezzük az előírt lépéseket, akkor az *algoritmus futásáról* beszélünk.

Már az általános iskolában tanulunk algoritmusokat. Az alapműveletek elvégezésének szokásos módja is egy-egy algoritmus. Az alapszerkesztések, a prímtényezőkre bontás megtanított módja, az Euklideszi-algoritmus mind jól ismertnek kell lenni egy érettségizett számára.

Legyen $\mathcal{I}$ az inputok, $\mathcal{O}$ az outputok halmaza. Tehát a probléma egy $f : \mathcal{I} \to \mathcal{O}$ függvény. Legtöbbször azonban nem vagyunk ennyire formálisak. A FAKTORIZÁCIÓ például egy probléma. A szóhasználat jelentheti a következők bármelyikét.

Példa (FAKTORIZÁCIÓ I). Input: egy pozitív egész szám. Output: prím osztóinak listája a megfelelő multiplicitásokkal.

Példa (FAKTORIZÁCIÓ II). Input: egy pozitív egész szám. Output: egy prím osztója.

Példa (FAKTORIZÁCIÓ III). Input: egy pozitív egész szám. Output: legkisebb prím osztója.

Példa (FAKTORIZÁCIÓ IV). Input: egy pozitív egész szám és egy t érték. Döntsük el van-e 2 és t közötti osztó.

Bármelyik megoldása átalakítható (alap programozási technikák, például rekurzió, bináris keresés ismeretével) a többi megoldásává.

Az inputok és outputok leírásában/leírtak értelmézésben is meg kell egyezniük a számítást követőknek. Ehhez $\mathcal{I}$ és $\mathcal{O}$ elemeit kódolnunk kell. A halmaz elemeit kódszavakkal helyettesítjük. Ehhez nézzünk néhány fontos alapfogalmat.

Definíció. Σ *ábécé* egy nemüres véges halmaz. Elemeire mint *betűk* vagy *karakterek* hívatkozunk.

Definíció. Σ ábécé esetén Σ^n az n hosszú karaktersorozatok, másképpen *szavak* halmaza. Σ^0 egy egyelemű halmaz, egyetlen eleme az ϵ üres (0 hosszú) szó. Σ^* a Σ ábécé-t használó véges szavak halmaza, azaz $\Sigma^* = \cup_{i=0}^{\infty}\Sigma^i$.

Az input/output kódolása az input elemeinek azonosítása kódszavakkal, azaz Σ^* elemeivel.

Példa. $\mathbb{N}^+$ (a pozitív egészek) egy kódolása lehet a következő. Legyen $\Sigma = \{0, 1\}$. Az x szám kódját úgy definiáljuk, hogy felírjuk kettes számrendszerben és a kezdő 1-est elhagyjuk. $1 \mapsto \epsilon$, $9 \mapsto 001$, hiszen $9 = 1001_2$. Ez egy bijekció $\mathbb{N}^+$ és $\{0, 1\}^*$ között. Ezt nevezzük $\mathbb{N}^+$ *standard kódolásának.*

Példa. Persze a tízes számrendszerben való felírás is egy kódolása $\mathbb{N}$-nek. Ekkor $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. A továbbiakban két lehetőségünk van. Vagy minden x természetes számnak egy kódját definiáljuk, a szokásost. Ekkor Σ^* nem minden eleme kódol inputot. 002010 például nem kódol inputot. Egy másik lehetőség, hogy minden számjegysorozatban a kezdő 0-kat elhagyva a maradékot értelmezzük. Ekkor egy számnak több kódja is van. $2010, 02010, 000002010$ kódok mindegyike ugyanazt a számot kódolja. Mindkét megoldás kérdéseket vet fel.

Példa. $\mathbb{N}$ kódolása az $\Sigma = \{1\}$ ábécé-vel is lehetséges: n kódja legyen n darab 1-es (1^n). Ezt $\mathbb{N}$ *unáris kódolásának* nevezzük.

Példa. $\mathbb{Q}$ szokásos kódolásában $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, /, -\}$ A racionális számot kódoló szvak egyetlen „/" jelet tartalmaznak, ami előtt egy természetes szám áll (esetleg egyetlen $-$ előjellel kezdve), míg a törtjel után egy pozitív egész kódja áll. Ebben a kódolásban a „fél" racionális számnak $1/2$ és $1005/2010$ is kódja. $1/2/3$, $12/0$, $-1/-2$ nem kódolnak racionális számot (legalábbis nem az egyszerű megállapodások alapján).

A kódolások fenti trükkös módjai kérdéseket vetnek fel. Adott Σ^* egy eleme. Lehet, hogy nem kódol inputot? Ha igen, akkor mit teszünk? Elvárjuk az algoritmust felhasználótól, hogy az általa megadott jelsorozat garantáltan inputot kódóljon és az algoritmussal bármi történhet, ha nem így tesz. Esetleg elvárjuk, hogy ebben az esetben az algoritmus jelezze, hogy „rossz kód". Ezekel a roblémákkal nem foglalkozunk. A szokásos kódolások olyanok, hogy algoritmusaink a legnagyobb követelményt is könnyen teljesítik kis többlet munkával.

A kódolás után egy számítási feladat egy $f : \Sigma^* \to \Sigma^*$ függvény.

Példa (FAKTORIZÁCIÓ V). Legyen $\Sigma - \{,\} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ;, ., \} \cup \{,\}$ Input: egy pozitív egész szám tízes számrendszerben felírva. Output: prím osztóinak növekvő listája, mindegyik osztót multiplicitása követi egy vesszővel elválasztva, majd egy pontos vessző követi, kivéve az utolsó prím osztót és multiplicitását, amit egy pont követ. Azaz:

$$24 \mapsto 2, 3; 3, 1.$$

$$121 \mapsto 11, 2.$$

$$43 \mapsto 43, 1.$$

Megjegyzés. Megjegyezzük, hogy csak megszámolható $\mathcal{I}$ és $\mathcal{O}$ halmazok kódolhatók.

Megjegyzés. A kódoláshoz elöször választani kell egy ábécé-t. Ekkor a $\Sigma = \{0, 1\}$ választáshoz is ragaszkodhatnánk. Néha azonban technikailag egyszerűbb lesz nagyobb ábécé-vel dolgozni.

A $\Sigma = \{1\}$ „minimál ábécé” is egy lehetőség. Ez azonban túlzott. Például az $\{0, 1, 2, ..., n-1\}$ elemű halmaz első kódolásában minden szám kódja legfeljebb $\log_2 n$ hosszú. Bármilyen nagyobb ábécé-t választunk ennél nagyságrendileg jobb kódolást nem találhatunk. (Az ábécé növelése csak konstansszorosára „nyomja össze” a kódokat.) Az egyelemű ábécé viszont rossz, ebben legalább $n-1$ hosszú kódszó is szükséges bármit teszünk.

Egy kódolás után beszélhetünk az *input méretéről*, az input jelsorozat karaktereinek száma, az input hossza.

Példa. $\mathbb{N}$ standard kódolásában n kódjának hossza $\lceil \log_2 n \rceil$. $\mathbb{N}$ unáris kódolásában n kódjának hossza n.

Példa. Legyen G egy egyszerű gráf a $V = \{1, 2, \ldots, v\}$ csúcshalmazon. Kódolásához legyen $\Sigma = \{0, 1, \}$. G kódjának hossza $\binom{v}{2}$ lesz (az ilyen alakú számokat nevezzük *háromszögszámoknak*). A kód pozíciói V kételemű részhalmazaival vannak azonosítva. Egy karakter/bit azt kódolja, hogy a megfelelő két csúcs össze van-e kötve. Mivel $2^{\binom{v}{2}}$ a kódolandó objektumok száma és $|\Sigma| = 2$ ezért rövidebb kódszavakkal dolgozva nem is lehetséges az összes v csücsü egyszerű gráf kódolása.

Példa. Legyen G egy e élű egyszerű gráf a $V = \{1, 2, \ldots, v\}$ csúcshalmazon. Ekkor kódja lehet, hogy V elemeit felsoroljuk és mindegyikhez kettőspont után felsoroljuk szomszédait (amelyek sorát pontosvesszővel választjuk el és ponttal zárjuk le). Azaz $\Sigma = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 0, ;, .\}$. Például egy kódolt gráf $1 : 2; 4.2 : 1.3 : 1; 5.4 : 1.5 : 3$. A gráf kódjának hosszát felülről becsülhetjük $(v+2e)(\log_2 v+1)$-gyel. Nagyságrendileg tömörebb kódolást nem is remélhetünk, hisz a kódolandó objektumok száma $\binom{\binom{v}{2}}{e}$.

Ki kell emelnünk a számítási problémák egy fontos esetét, a döntési problémákat. Ekkor egyetlen bitet számolunk ki (függvényünk két értékű). Azt is mondhatjuk (ez így szokás) hogy egy inputot vagy elfogadunk vagy elvetünk. Tehát egy eldöntési probléma azonosítható Σ^* egy részhalmazával, az elfogadandó inputok halmazával. A szavak egy elfogadandó részhalmazát nyelvnek nevezzük.

Definíció. Egy döntési probléma leírható egy $L \subset \Sigma^*$ nyelvvel. (Illetve a nyelv értelmezhető, mint a „hozzátartozik-e” döntési probléma.)

Az algoritmus fenti leírása matematikailag nem pontos (lényegésben az algoritmus idegen szót egy ismerősebb „eljárás” szóval helyettesítettük). Mégis, talán rávilágítottunk arra, hogy egy algoritmus értelmézéséhez sok megállapodás szükséges.

Azt is szeretnénk hangsúlyozni, hogy az algoritmus statikus leírása mellett (amit egy tankönyvben láthatunk) van egy dinamikus kép is (például, amikor két háromjegyű szám esetén szorzatukat kiszámoljuk). Az idő telésével a papírunkon számok jelennek meg, egész a számítás végééig szervezzünk munkaterületünket, amikor is leállunk (például az eredmény kétszeres aláhúzásával jelezve a számítás végét). A tankönyvben leírt eljárást akkor értettük meg, ha képesek vagyunk futtani azt különböző inputokon. Illetve, ha jól begyakoroltunk egy algoritmust és a matematikai és nyelvi kifejezés egy érettségi szintjét elértük, akkor képeseknek kell lennünk valamilyen tankönyvi leírását adni az eljárásunknak. A statikus és dinamikus szemlélet együtt jár. Azért életünk/matematikai tanulmányaink során a dinamikus képpel találkozunk először. Ez az az út, amit az alapműveletek elvégzésénél az általános iskola első osztályában mindenkivel követtetnek.

Az algoritmus matematikai fogalma

Az algoritmus matematikai fogalma a XX. század első harmadában alakult ki. Egyik ösztönzője a logika fejlődése, illetve Hilbert nevezetes problémái közül a tizedik volt. Ebben azt kérdezte Hilbert, hogy van-e olyan algoritmus, ami egy adott egészegyütthatós polinomról eldönti, hogy van-e egész gyöke (Diophantikus számelmélet alapproblémája). A válasz, mint később kiderült: „nem". Ennek igazolása már lehetetlen az algoritmus fogalmának matematizálása nélkül.

A probléma érdekes. Az algoritmus/eljárás szavak részei mindennapi szóhasználatunknak, de a matematikában nincsenek értelmezve. Egy definíció megadása, akkor sikeres, ha a matematikus társadalom többsége rábólint: „elfogadom korrekt definíciónak". Az a pillanat, amihez ezt kötik az Church egy előadása (amit 1935-ben tartott az Amerikai Matematikai Társulat számára). A felhívást, hogy az algoritmus korrekt definíciójának keresése véget érhet (ott vagyunk a „szent grálnál") *Church-tézisként* hivatkozzák. Érdekes megjegyezni, hogy a tézis bejelentésénél nem volt meg a matematikusok összhangja. Például Gödel nem fogadta el. Turing munkája, majd az, hogy Turing, Church, Gödel és mások próbálkozásai mind ekvivalens fogalomhoz vezetnek a tézist elfogadottá tették. Ennek ellenére, ha valami technológiai áttörés történik (amely gyökeresen átalakítja a mindennapi képünket a számítás fogalmáról, például a kvantum számítógépek megvalósításának elméleti lehetősége) a tézist újra kell értékelni.

Mi az alábbiakban Turing definícióját ismertetjük, ami talán a legtranszparensebben próbálja az algoritmus fogalmát megragadni. Ez az algoritmus fogalom a dinamikus szemléletét írja le.

Definíció. Turing-gép egy konfigurációját írjuk le. Ennek „fizikai" részei: input-szalag, munkaszalag, outputszalag, fej. A szalagok mezők sorozatát tartalmazzák. A t szalag mezői $\{M_i^t\}_{i=0}^{\infty}$ (azaz m_{100}^{input} az input szalag 100 indexű/101-edik mezője, m_0^{munka} a munka szalag első/0 indexű mezője). A mezőket úgy kell elképzelni mint egy négyzethálós papír egy sorát, amelyik jobb irányban végtelen. A mezők tartalma egy-egy karakter. Az input- és outputszalag mezői a feladat kódolásához használt Σ ábécé elemeit tartalmazzák. Az output szalagon egy (új) $\smile$ üres karakter is szerepelhet. A munkaszalaghoz egy Γ ábécé-t használunk. Ezenkívül (ezen ábécé karaktereitől különböző „határolójeleink" is vannak: $\triangleright$ és $\triangleleft$. Ezek a szalagok „határmezőinek" „bevésett" tartalma (lásd későbbi formális leírás).

A fej a Turing-gép szalagaival szemmel, illetve kézzel kapcsolódik. A „szem" egy mező értékét olvassa, a kéz a látott mezőt írhatja felül. A gépnek egy input-, munka-szemmel, munka-kézzel és output-kézzel rendelkezik. Ezek helyzetét az írja le, hogy melyik mező felett helyezkednek el. Azaz mindegyik szalaghoz tartozik egy pozíció, amit a fej kontrolál (a munka-szem által látott és a munka-kéz által írható mező ugyanaz).

A fejnek van egy bizonyos állapota. A lehetséges állapotok egy S véges halmazt alkotnak (S elemeit állapotoknak hívjuk, S az állapothalmaz) A konfiguráció tehát egy n hosszú input mellett egy hetes:

$$\langle \{M_i^{input}\}_{i=0}^{n+1}, \{M_i^{munka}\}_{i=0}^{\infty}, \{M_i^{outpu}\}_{i=0}^{\infty}, p^{input}, p^{munka}, p^{output}, s \rangle,$$

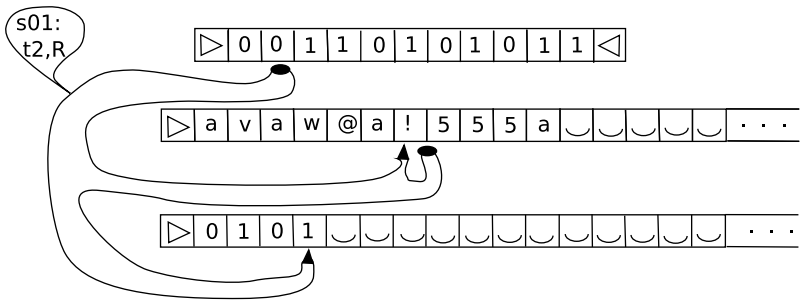
ahol $M_0^{input} = M_0^{munka} = M_0^{output} = \triangleright$, $M_{n+1}^{input} = \triangleleft$, $p^{input} \in \{0, 1, 2, \ldots, n+1\}$, $p^{munka}, p^{output} \in \mathbb{N}$, $s \in S$.

Definíció. Az $\omega \in \Sigma^n$ inputhoz tartozó kezdőkonfiguráció definíciójához szükséges néhány előzetes megállapodás. Legyen $START \in S$ egy speciális állapot (a számítás kezdetét jeloz" állapota gépünknek) és legyen $\smile$ egy speciális eleme Γ-nak (az üres karakter). Legyen

$$\langle \{M_i^{input}\}_{i=0}^{n+1}, \{M_i^{munka}\}_{i=0}^{\infty}, \{M_i^{output}\}_{i=0}^{\infty}, p^{input}, p^{munka}, p^{output}, s\rangle,$$

ahol $M_0^{input} = M_0^{munka} = M_0^{output} = \triangleright$, $M_{n+1}^{input} = \triangleleft$, $M_i^{input} = \omega_i$ $(i = 1, 2, \ldots, n$, $M_i^{munka} = M_i^{output} = \smile$ $(i \in \mathbb{N}^+)$, $p^{input} = p^{munka} = p^{output} = 0$, $s = START$.

A mellékelt rajzon egy vizualizációját adjuk a Turing-gép egy állapotának.



1. ábra. Egy képzeletbeli gép képzeletbeli állapota

Megjegyzés. Néhány fontos megállapítást, megjegyzést teszünk:

1) A fej a konfigurációnak csak egy szűk részét „látja". Ez a két szem által látott mező tartalma és az állapota (azaz $(\Sigma \cup \{\triangleright, \triangleleft\}) \times (\Gamma \cup \{\triangleright, \smile\}) \times S$ egy eleme. Az algoritmus a konfigurációt csak lokálisan írja felül. A lokalitás miben létét az alábbiakban fogaljuk össze:

   - A szemek (és velük a kezek) mozgása „folytonos". Ez alatt azt értjük, hogy mindegyik szalagon az új pozíció legfeljebb 1-gyel tér el az előzőtől.
   - Az inputszalag csak olvasható, a munkaszalag olvasható és írható, az outputszalag csak írható.
   - Az output szalag funkciója csak az output leírása. Ezt a következő megállapodással garantáljuk: Az input szalagra akkor írunk majd, ha a kéz egyet jobbra mozdul. Más mozgást nem is engedélyezünk.

2) A határoló jelek ($\triangleright$ és $\triangleleft$) szerepe a szemek/kezek szalag fölött tartása.

Definíció. A változtatás lokalitása miatt ezt egy egyszerű, úgy nevezett *átmeneti-függvény* írja le:

$$\delta : (\Sigma \cup \{\triangleright, \triangleleft\}) \times (\Gamma \cup \{\triangleright, \smile\}) \times S \to \{B, ., J\} \times \Gamma \times \{B, ., J\} \times (\{.\} \cup \Sigma) \times S.$$

Megjegyzés. Az átmenetifüggvény értelmezési tartománya az 1) megjegyzés alatt leírtakat formalizálja.

Az átmeneti függvény értékkészletében minden értéknek öt komponense van. Ezek rendere a következőt írják le:

1) input-szem mozgása ('B'= bal szomszédra ugrás, '.' = maradás, 'J' = jobb szomszédra ugrás),

2) munka-kéz által leírt karakter (a nem írás a már ott lévő karakter újraírása),

3) munka-szem mozgása (ami egyben a kéz mozgása is lásd az input-szem mozgására vonatkozó magyarázatot),

4) az output-kéz mozgása és írása egybeolvasztva (a kéz vagy marad és nem ír (.), vagy jobbra mozog és ír egy karaktert (Σ egy eleme)),

5) az új konfigurációban a fej állpota.

Az értelmezés után egyszerű gyakorlat a felvett érték alapján egy κ *konfiguráció* κ^+ *rákövetkezőjének* formális leírása. Ezt az olvasóra bízzuk.

A 2) megjegyzést külön feltétel fogalmazza meg. Például $\delta(\triangleright, karakter, STATE)$-nek olyan értéknek kell lenni, hogy első koordinátája nem B. $\delta(karakter, \triangleright, STATE)$-nek olyan értéknek kell lenni, hogy harmadik koordinátája nem B, második koordinátája lényegtelen mert a határoló jel nem írható felül, a rákövetkező konfigurációban a munkaszalag tartalma ugyanaz lesz mint korábban. A teljes feltételrendszer felírását nem végeztük el. Mindenki megteheti, aki úgy érzi, hogy a formalizálás segíti megértését.

Ezekután már természetes az algoritmus futásának dinamikus képét leírni.

Definíció. Legyen $\{\kappa_i\}_{i=0}^{\infty}$ konfigurációk azon sorozata, amelyre $\kappa_0 = \kappa_0(\omega)$ (az ω-hoz tartozó kezdőkonfiguráció) és $\kappa_{i+1} = \kappa_i^+$. Ezt a konfiguráció-sorozatot az ω *inputhoz tartozó futásnak* nevezzük.

Definíció. Legyen $STOP$ egy speciális állapot, azaz $STOP \in S$. Ennek szerepe a számítás végének jelölése. A futás végtelen sorozatát Legyen $\{\kappa_i\}_{i=0}^{\ell}$ konfigurációk azon sorozata, amelyre $\kappa_0 = \kappa_0(\omega)$ (az ω-hoz tartozó kezdőkonfiguráció) és $\kappa_{i+1} = \kappa_i^+$, továbbá $\ell = \min\{i : \kappa_i$ állapota $STOP\}$. Amennyiben a minimum mögött álló halmaz üres $\ell = \infty$. Ha $\ell < \infty$, akkor azt mondjuk *a futás leáll* és κ_ℓ-ben az outputszalag tartalma a $\triangleright$ jel után az output-kézig tartalmazza a kiszámított outputot.

Definíció. Egy f függvényt kiszámít egy T Turing-gép, ha minden $\omega \in \Sigma^*$ esetén leáll a Turing-gép és a kiszámított érték $f(\omega)$.

Definíció. Eldöntési feladat megoldására szolgáló Turing-gép esetén az outputszalag egyetlen bit leírására szolgál. A definíció egyszerűsíthető: A STOP álapotot helyettesítsük ELFOGAD és ELVET állapotokkal. Ebben az esetben az outputszalagra nincs szükségünk. Döntési feladatoknál ezzel a modellal dolgozunk, ezt *eldöntő Turing-gépnek* nevezzük. Ekkor a futás akkor és csak akkor áll le, ha ELVET vagy ELFOGAD állapotba kerül.

Definíció. Egy eldöntő T Turing-gép eldönti az L nyelvet, ha minden $\omega \in L$ esetén ELFOGAD állapottal áll le a gép és minden $\omega \notin L$ esetén ELVET állapottal áll le a gép.

Megemlítünk egy további lehetőséget az elfogadás/elvetés „kódolására".

Definíció. Egy eldöntő T Turing-gép felsorolja az L nyelvet, ha minden $\omega \in L$ esetén ELFOGAD állapottal áll le a gép és minden $\omega \notin L$ esetén nem áll le.

⋆

A fenti definíciókhoz nagyon sok megállapodást rögzítettünk. Ezek a megállapodások esetlegesek. Más utat követve más Turing-gép definícióhoz jutottunk volna el. A kiszámíthatóság, eldönthetőség, felsorolhatóság végső definíciója viszont már nem függ ezektől az apróságoktól. Ezek a fogalmak érzéketlenek a részletekben rejlő finomságoktól.

Általában a problémák nem pontos leírása sem zavaró. Könnyű látni, hogy a FAKTORIZÁCIÓ mindegyik változata kiszámítható függvényhez vezet.

Az esetlegesség aláhúzására mi az alapmodel két változatát ismertetjük.

Definíció. k-szalagos Turing-gép. Ebben a munkaszalagot k darab munkaszalaggal (és mindegyikhez egy-egy munkaszem/munkakéz) helyettesítjük. Ekkor a fej által látható részét a konfigurációnak, illetve az átmenetifüggvény módosítását minden érdeklődő olvasó elvégezheti.

A 2010 szalagos Turing-gép által kiszámolt függvény kiszámítása egy munkaszalag segítségével (azaz a standard modelben) nem egyszerű, de csak technikai problémákat rejt.

Definíció. Az egyetlen szalagos Turing-gépnek egyetlen (amely szerepe input, munka, output egyben) szalagja van, ami egy irányban végtelen. Erre írjuk az inputot $\triangleright$ és $\triangleleft$ jelek közé. Az első egy bevésett határolójel. A második csak az input végének jelölésére szolgál. Ez felülírható, akár az input karakterei. Leálláskor a szalagot mint outputszalag interpretáljuk.

⋆

Rögzítsünk egy Σ alapábécé-t. Az összes döntési probléma halmaza legyen $\mathcal{U}$. Legyen $\mathcal{S}$ a felsorolható és $\mathcal{D}$ az eldönthető nyelvek halmaza. Nyilván $\mathcal{D} \subset \mathcal{S} \subset \mathcal{U}$.

$\mathcal{D}$ és $\mathcal{S}$ halmazok megszámlálhatóak. Ehhez csak azt kell észrevennünk, hogy a munkaszalag ábécé-jénél feltehető, hogy a választható karakterek $\mathbb{N}$ egy véges kezdőszeletét alkotják. Hasonlóan az S állapothalmazról is feltehető ugyanez. Ekkor azonban csak megszámlálhatóan végtelen Turing-gép van, így eldönthető és felsorolható nyelv is. $\mathcal{U}$ kontinuum számosságú.

Az egész gyökökkel rendelkező egész együtthatós polinomok kódjai által alkotott DIOPHANTOS nyelv nincs $\mathcal{D}$-ben (Hilbert tizedik problémájának megoldása alapján tudjuk ezt). könnyen tervezhető egy Turing-gép, ami viszont felsorolja ezt a nyelvet.

Bonyolultsági osztályok

Definíció. Egy T Turing-gép időigénye egy ω inputon $\ell := TIME(\omega; T)$, ha futása $\{\kappa_i\}_{i=0}^{\ell}$, azaz az ℓ-edik konfigurációban kerül először STOP állapotba.

Azt is mondhatnánk, hogy a konfigurációk sorozatában az index egy óra ütéseit számolja. Minden óraütésre a rákövetkező konfigurációba kerül a gép.

Definíció. Egy T Turing-gép tárigénye egy ω inputon $s := SPACE(\omega; T)$, ha futása során a munkaszem/kéz alatti mező legnagyobb indexe s.

A szemek/kezek folytonos mozgása miatt nyilvánvalóan

$$SPACE(\omega; T) \leq TIME(\omega; T).$$

A futás ideje és tárigénye nyilván függ az input hosszától. Ez a függés igen fontos számunkra. Az alábbi definíció az algoritmus legrosszabb eseet analíziseként hivatkozzuk. Az itt leírt függvény egy garancia, hogy bármilyen n hosszú inputtal is dolgozunk az idő/tár igény nem haladja meg ezt (a legrosszabb esetben sem).

Definíció.

$$TIME(n; T) = \max\{TIME(\omega; T) : \omega \in \Sigma^n\},$$
$$SPACE(n; T) = \max\{SPACE(\omega; T) : \omega \in \Sigma^n\}.$$

Egy Turing-gép esetén a $TIME(n; T)$ vagy $SPACE(n; T)$ függvények meghatározása nehéz, technikai lehet. Legtöbbször a függvények jó becslése, angyságrendjének meghatározása a reális, fontos cél. A nagyságrendek jelölésére szokásos jelöléseket az alábbi definíció foglalja össze.

Definíció. $g(n) = \mathcal{O}(f(n))$ akkor és csak akkor, ha alkalmas $\alpha > 0$ konstansra $g(n) \leq \alpha \cdot f(n)$.

$g(n) = \Omega(f(n))$ akkor és csak akkor, ha alkalmas $\beta > 0$ konstansra $g(n) \geq \beta \cdot f(n)$.

$g(n) = \Theta(f(n))$ akkor és csak akkor, ha alkalmas $\alpha, \beta > 0$ konstansra $\beta \cdot f(n) \leq g(n) \leq \alpha \cdot f(n)$.

Az alábbiakban bevezetjük a legfontosabb fogalaminkat, a bonyolultsági osztályok prototípusait. Ezeket csak eldöntési problémákra írjuk fel. Számítási problémák esetén is bevezethetők a megfelelő függvényosztályok. A bonyolultságelmélet alapjait azonban csak nyelvosztályokkal dolgozva is nagyon jól megvilágíthatjuk. Mi is — ahogy szokás — ezt az utat követjük.

Definíció.

$$TIME(f(n)) = \{L : \text{alkalmas } T \text{ Turing-gép eldönti } L\text{-et, és } TIME(n; T) \leq f(n)\},$$

$$SPACE(f(n)) = \{L : \text{alkalmas } T \text{ Turing-gép eldönti } L\text{-et, és } SPACE(n; T) \leq f(n)\}.$$

Megjegyzés. Néha a definícióban rejlő T Turing-gépre is mondják, hogy $TIME(f(n))$-beli. Hogy a lehető legkevesebb eltérés legyen a definíciók között és technikailag is könnyebben kezelhető osztályt kapjunk az általános megállapodás szerint a definícióbeli T Turing-gép k-szalagos Turing-gép.

Most pedig a legalapvetőbb bonyolultsági osztályokat írjuk le.

Definíció.

$$\mathcal{L} = \cup_{\alpha \in \mathbb{N}} SPACE(\alpha \log n) \quad \text{(logaritmikus tár)}, \tag{1}$$
$$\mathcal{P} = \cup_{\alpha \in \mathbb{N}} TIME(\alpha n^{\alpha}) \quad \text{(polinomiális idő)}, \tag{2}$$
$$\mathcal{PSPACE} = \cup_{\alpha \in \mathbb{N}} SPACE(n^{\alpha}) \quad \text{(polinomiális tár)}, \tag{3}$$
$$\mathcal{EXP} = \cup_{\alpha \in \mathbb{N}} TIME(e^{n^{\alpha}}) \quad \text{(exponenciális idő)}, \tag{4}$$
$$\mathcal{EXPSPACE} = \cup_{\alpha \in \mathbb{N}} SPACE(e^{n^{\alpha}}) \quad \text{(exponencialis tár)}. \tag{5}$$
$$\tag{6}$$

Ezek az osztályok már függetlenek a model leírásában rejlő apróságoktól. Lineáris idő azonban már olyan osztályhoz vezetne, ahol az apró döntéseink befolyásolják a kialakuló fogalmakat.

Arra is figyelnünk kell, hogy hogyan kódoltuk a feladatot. Ha a FAKTORIZÁCIÓ problémában az adott szám unárisan van kódolva, akkor az általános iskolában megtanult eljárás azt bizonyítja, hogy a probléma $\mathcal{P}$-beli. Ugyanekkor természetes kódolásoknál (például a standardnál) a $\mathcal{P}$ osztályhoz tartozás a matematika egy nagy problémája. Ha egy probléma megnevezésénél a kódolás nincs rögzítve, akkor valamelyik „természetes kódolásra" kell gondolni. Ezek általában olyan formalizálásokhoz vezetnek, amiknél egy bonyolultsági osztályhoz tartozás ekvivalens.

A $TIME(f(n))$ és $SPACE(f(n))$ osztalyok leírásában az $f(n)$ függvényről semmit sem mondtunk. Így elképzelhető, hogy nem is kiszámítható függvényekre alapozunk kiszámíthatósági osztályokat. Ez többször technikai problémákhoz vezethet. A következő definíciók olyan függvényeket írnak le, amiknél ezek a technikai nehézségek könnyen megoldhatók. Ha $TIME(f(n))$ és $SPACE(f(n))$ osztályokkal dolgozunk mindig feltesszük, hogy a használt $f(n)$ függvények szépek.

Definíció. $t(n)$ szép idő-függvény, ha van olyan T Turing-gép a $\Sigma = \{1\}$ ábécé-vel, ami az 1^n inputon pontosan $t(n)$ lépés után áll le.

Definíció. $s(n)$ szép tár-függvény, ha van olyan T Turing-gép a $\Sigma = \{1\}$ ábécé-vel, ami az 1^n inputon pontosan $s(n)$ tár felhasznalásával áll le.

A természetes függvények, a logaritmus, exponenciális, polinom függvényekből felépített függvények mind szépek (idő esetében persze növekedése gyorsabb legyen mint az input elolvasásához szükséges idő).

1. Lemma. *Legyen $s(n)$ egy szép tár-függvény. Tegyük fel, hogy az L nyelvet eldönti egy $s(n)$ tárigényű T Turing-gép. Ekkor van olyan $\widehat{T}$ Turing-gép is, amely ugyanazt a nyelkvet dönti el és bármely n hosszú input esetén a leálló konfigurációjában a munkaszalag tartalma standard: a határolójel után $s(n)$ darab $\sqcup$ „radírjel" következik, amik után már csak (érintetlen) $\smile$ karakterek jönnek.*