

ALGORITMUSOK ÉS BONYOLULTSÁGELMÉLET

Matematika MSc hallgatók számára

Nyelvosztályok, modellek

Előadó: Hajnal Péter

2014.

Emlékeztető. Az előző előadásban bevezettünk egy számítási modellt, amelyet a későbbiek során *standard modellnek* fogunk nevezni. Elevenítsük fel a modell részeit!

A standard modellben a számításokhoz három eltérő funkcióval rendelkező szalagot használunk. Ezek egy irányban (általában jobbra) végtelen mezősorozatok, a nevük: inputszalag, munkaszalag, outputszalag. A Turing-gép feje a szalagok jellegétől függően a mezők leolvasását és felülírását végzi. Az *inputszalag* a Σ ábécé karaktereiből álló véges hosszúságú *szót* tartalmazza \triangleright és \triangleleft szimbólumok között, amelyek az input kezdetét és végét jelzik. A fej karakterenként látja az inputot egy szemmel, amely az inputszalagon mindkét irányban képes mozogni. A *munkaszalag* első mezője szintén a szalag kezdetét jelölő \triangleright „bevésett” (nem felülírható) karaktert tartalmazza. A szalag mezőibe a Γ -val jelölt ún. *munkaábécé* karaktereit írhatjuk. A fej a munkaszalag mezőit egy jobbra-balra együtt mozgó szemmel és „ceruzát tartó” kézzel olvassa, ill. írja. Végül az ugyancsak \triangleright jellel kezdődő *outputszalagra* kerül a számítás eredménye, melyen a fej egy „ceruzát tartó” kézzel ír, a mezőkön egyenként jobbra haladva. A fejnek egy fontos jellemzője az állapota, amely a véges S állapothalmaz eleme.

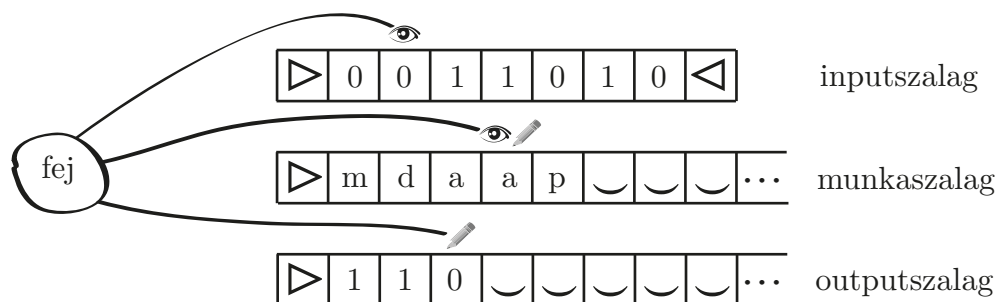
A Turing-gép működését az f átmeneti függvény szabja meg. Ennek értelmezési tartományát az ún. *látható részek* alkotják, amelyek alapján f egy „update-elést” (frissítést) hajt végre. A látható részek a fej

- (1) inputszalagon futó szeme által látott karakter,
- (2) munkaszalagon futó szeme által látott karakter,
- (3) állapota.

Az átmeneti függvény megadja, hogy a látható részek alapján

- (4) milyen irányba lépjen az inputszalagot pásztázó szem,
- (5) milyen karakter kerüljön a munkaszalag aktuális mezőjébe,
- (6) merre mozduljon a munka szem-kéz,
- (7) mi kerüljön az outputszalagra,
- (8) végül pedig meghatározza, hogy a fej milyen állapotba kerüljön.

$$f: \underbrace{(\Sigma \cup \{\triangleright, \triangleleft\})}_{(1)} \times \underbrace{(\Gamma \cup \{\triangleright, \smile\})}_{(2)} \times \underbrace{S}_{(3)} \rightarrow \underbrace{\{\leftarrow, \cdot, \rightarrow\}}_{(4)} \times \underbrace{\Gamma}_{(5)} \times \underbrace{\{\leftarrow, \cdot, \rightarrow\}}_{(6)} \times \underbrace{(\Sigma \cup \{\cdot\})}_{(7)} \times \underbrace{S}_{(8)}$$



1. ábra. A standard modell.

1. Bonyolultsági mértékek

Most definiáljuk egy algoritmus/Turing-gép időigényét és tárigényét.

Definíció. Egy T Turing-gép időigénye egy ω inputon $\ell := TIME(\omega, T)$, ha futása $\{\kappa_i\}_{i=0}^{\ell}$, vagyis az ℓ -edik konfigurációban kerül először $STOP$ állapotba, illetve ∞ , ha futása végtelen (nem éri el a $STOP$ állapotot).

Definíció. Egy T Turing-gép tárigénye egy ω inputon $s := SPACE(\omega, T)$, ha futása során a munkaszem/kéz alatti mező legnagyobb indexe s , illetve ∞ , ha a munkaszem/kéz tetszőleges messze elmozdul a munkaszalag baloldali határától.

Mivel a Turing-gép legfeljebb annyi mezőt látogathat meg a munkaszalagon, amennyit mozog, ezért ezen jellemzők között fennáll a nyilvánvaló

$$SPACE(\omega, T) \leq TIME(\omega, T)$$

összefüggés. A futás idő- és tárigényét nyilván az input hosszától való függése alapján ítélni tudjuk meg. A következő definícióval ezen függést tudjuk kifejezni.

Definíció. Legyen $t: \mathbb{N} \rightarrow \mathbb{R}$ egy tetszőleges függvény. Azt mondjuk, hogy egy T Turing-gép eleme a $TIME(\mathfrak{t}(\mathbf{n}))$ halmaznak,

$$TIME(\omega, T) \leq t(|\omega|).$$

Definíció. Legyen $s: \mathbb{N} \rightarrow \mathbb{R}$ egy tetszőleges függvény. Azt mondjuk, hogy egy T Turing-gép eleme a $SPACE(\mathfrak{s}(\mathbf{n}))$ halmaznak,

$$SPACE(\omega, T) \leq s(|\omega|).$$

Természetesen ezek az osztályok nyelvosztályokként is megfogalmazhatók:

Definíció. Legyen $t: \mathbb{N} \rightarrow \mathbb{R}$ egy tetszőleges függvény. Azt mondjuk, hogy egy L nyelv eleme a $TIME(t(n))$ halmaznak, ha létezik olyan T Turing-gép, amely

- (i) eldönti L -et, és
- (ii) $TIME(\omega, T) \leq t(|\omega|)$.

Definíció. Legyen $s: \mathbb{N} \rightarrow \mathbb{R}$ egy tetszőleges függvény. Azt mondjuk, hogy egy L nyelv eleme a $SPACE(s(n))$ halmaznak, ha létezik olyan T Turing-gép, amely

- (i) eldönti L -et, és

(ii) $SPACE(\omega, T) \leq s(|\omega|)$.

Az imént bevezetett $TIME(t(n))/SPACE(s(n))$ jelöléseknél hasznosabbak azok az osztályok ahol az idő, illetve tár korlátozást nem egy függvénnyel, hanem egy „nagyságrenddel” írjuk elő.

Definíció. \circ *Polinomiális időben eldönthető nyelvek:*

$$\mathcal{P} := \bigcup_{a \in \mathbb{N}} TIME(an^a + a) = \bigcup_{p \in \mathbb{R}[x]} TIME(p(n)).$$

Az a szorzó az 1 hosszúságú inputokon szükséges 2 lépés végett szükséges, így írhatnánk a helyett 2-est is. A két unió, azért egyezik meg, mert bármely $p \in \mathbb{R}[x]$ polinomhoz létezik olyan $a \in \mathbb{N}$ természetes szám, melyre $p(n) \leq an^a + a$, ahol $n \in \mathbb{N}$.

\circ *Exponenciális időben eldönthető nyelvek:*

$$\mathcal{EXP} := \bigcup_{a \in \mathbb{N}} TIME(2^{an^a + a}).$$

\circ *Polinomiális tárral eldönthető nyelvek:*

$$\mathcal{PSPACE} := \bigcup_{a \in \mathbb{N}} SPACE(an^a + a).$$

\circ *Exponenciális tárral eldönthető nyelvek:*

$$\mathcal{EXPSPACE} := \bigcup_{a \in \mathbb{N}} SPACE(2^{an^a + a}).$$

\circ *Logaritmikus tárral eldönthető nyelvek:*

$$\mathcal{L} := \bigcup_{a \in \mathbb{N}} SPACE(a \log n),$$

itt megjegyezzük, hogy $\log n$ az $n = 0$ esetben nyilván nem értelmezhető, azonban ettől az elfajuló (nulla hosszúságú input) esettől eltekintünk. Továbbá \mathcal{L} miatt szükséges az inputszalag és a munkaszalag elkülönítése is, mert máskülönben nehéz lenne a logaritmikus tárat mérni/kimutatni.

Könnyen belátható, hogy a nyelvosztályok között az alábbi tartalmazások teljesülnek:

$$\mathcal{L} \subseteq \underbrace{\mathcal{PSPACE}}_{\cup \mathcal{P}} \subseteq \underbrace{\mathcal{EXPSPACE}}_{\cup \mathcal{EXP}} \subseteq \mathcal{D} \subseteq \mathcal{S} \subseteq \mathcal{P}(\Sigma^*)$$

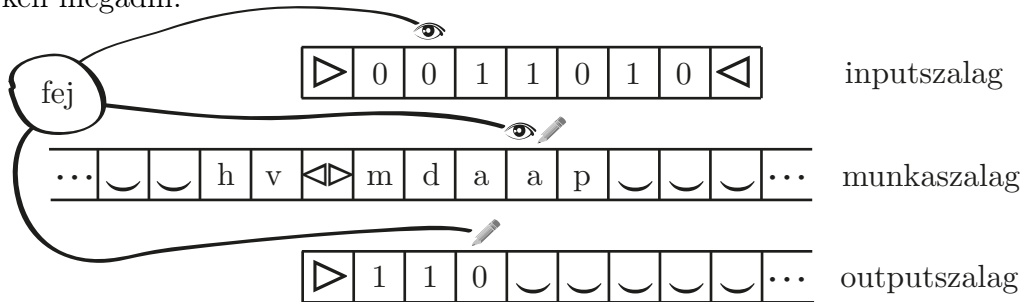
Természetesen további kapcsolatok is vannak, ezekről a későbbiek során esik szó. Nyelvosztályokból is jelentősen több létezik, mint amit itt megadtunk, a részletekért lásd a Qwiki oldalát: http://qwiki.stanford.edu/index.php/Complexity_Zoo.

A bonyolultságelmélet „feladata” a \mathcal{D} nyelvosztály finomítása, az ide tartozó nyelvek vizsgálata. \mathcal{D} -n kívüli nyelvek összehasonlítása és vizsgálata inkább matematikai logika jellegű kérdésekhez vezet.

2. A standard modell változatai

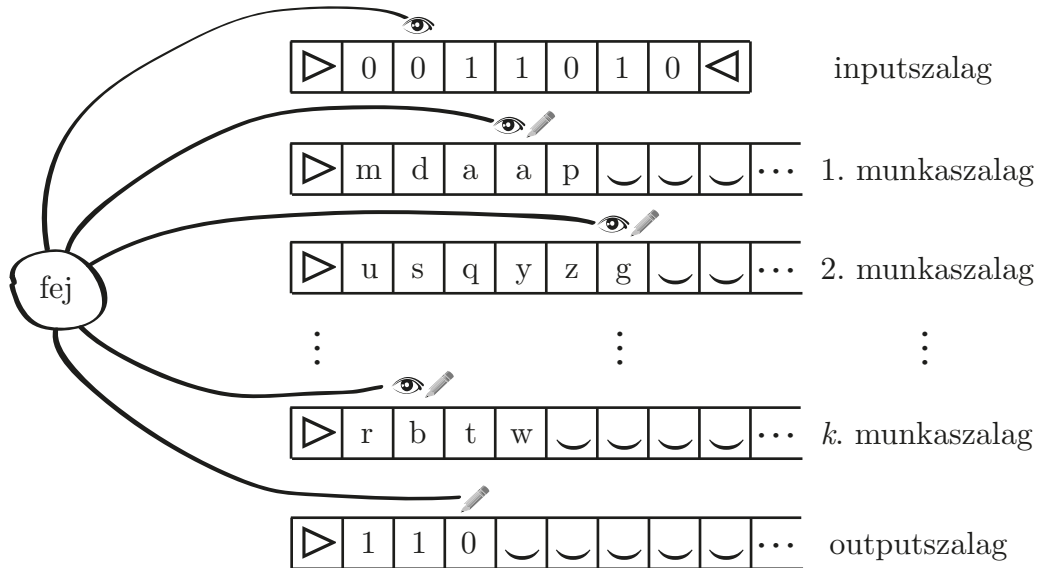
A fent leírt standard modell esetleges, hiszen például az, hogy egy irányban végtelen szalagokat használunk továbbá, hogy éppen három szalagon dolgozunk önkényes választások. Így természetesen léteznek más változatok is, íme néhány példa:

- **Két irányban végtelen munkaszalag:** A munkaszalag bal és jobb irányban is végtelen és nem tartalmaz jidezszalag eleje jelet, csupán egy kezdeti mezőt kell megadni.



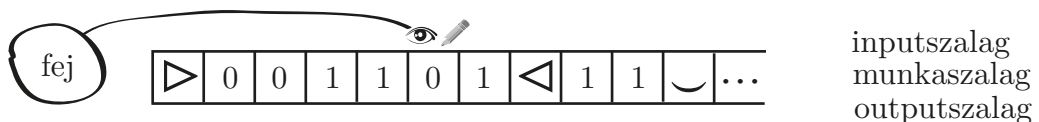
2. ábra. A két irányban végtelen munkaszalag.

- **k -szalagos modell:** Rögzített $k \in \mathbb{N}$ számú munkaszalagot használunk, és minden egyes munkaszalaghoz tartozik egy szem-kéz páros. Fontos megjegyezni, hogy k nem függ az input méretétől.



3. ábra. A k -szalagos modell.

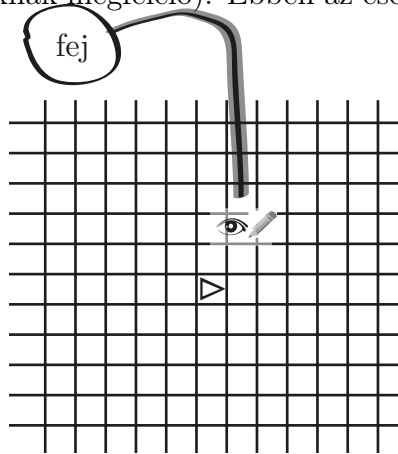
- **Egyszalagos modell:** Minden művelet egyetlen szalagon történik, amely így egyszerre input-, munka-, és outputszalag. A szalag jobbra végtelen és a kezdeti mezőket az input foglalja el $\triangleright, \triangleleft$ jelek között. Ebben a modellben az input is felülírható és az output a szalag tartalma a $STOP \in S$ állapot elérésekor.



4. ábra. Az egyszalagos modell.

- **Kétdimenziós munkaszalag:** Érdemes a munkaszalagot úgy elképzelni, mint egy végtelen négyzethálós független háló (ez lehet a teljes síkot lefedő háló vagy

csupán egy negyedsíknak megfelelő). Ebben az esetben is ki kell jelölnünk egy kezdeti mezőt.



5. ábra. Munkaszalag a síkon.

Nyilvánvaló, hogy a különböző modellekhez más látható részek tartoznak, ebből adódóan pedig eltérők a modellek átmeneti függvényei. Emellett egy természetesen felmerülő kérdés, hogy függ-e a modell megválasztásától egy adott számítási feladat megoldásának „nehézsége”? Ugyanazt a problémát megoldó, de eltérő modellek esetleg különbözhetnek futásidő, tárigény tekintetében? A válasz: „Elmosott” idő/tár-korlátánál a modell nem számít. Az $\mathcal{L}, \mathcal{P}, \mathcal{PSPACE}, \mathcal{EXPTIME}, \mathcal{D}$ nyelvosztályok változatlanok maradnak a különböző modelleknél. (Ha \mathcal{D} változna a modellel, akkor a Church-tézist „megcáfolnánk”).

Ha azonban finomabb osztályokat nézünk, akkor a modell választása lényegesen kihat a nyelvosztályunkra. Például „lineáris idő”:

$$\cup_{\alpha, \beta \in \mathbb{N}} TIME(\alpha n + \beta)$$

más lesz, ha a standard modellre alapítjuk, illetve ha az egyszalagos modellen nyugszik. Ha szükségünk van ilyen finomabb nyelvosztályra, akkor a definíció lényegét érintő „létezik T Turing-gép” kifejezés alatt egy k -szalagos modell létezését értjük. Azaz a létezik kvantor kibontva: $\exists k \in \mathbb{N} \quad \exists \Gamma \quad \exists f$.

Egy érdekesség, hogy ha léteznek forradalmian új számítási modellek. A fizikusok a (klasszikus mechanika, fizikán alapuló) Turing-géppel szemben bevezetették (az egyelőre meg nem épített) kvantum számítógép fogalmát. Az erre alapított kiszámítható nyelvek/ \mathcal{D} osztály ugyanaz maradt. Azt gyanítjuk, hogy ez a forradalmi modellváltás \mathcal{P} fogalmát megváltoztatja.

3. A PALINDROM nyelv példája

Definíció. Legyen

$$PALINDROM = \{ \omega = \omega_1, \dots, \omega_n : \text{ahol } \omega_i = \omega_{n+1-i} \\ \text{minden } i \in \{1, 2, \dots, n\} \text{ esetén} \}$$

a palindrom szavak nyelve.

Tehát döntési problémával állunk szemben. Adott egy szó, el kell döntenünk, hogy előlről és hátulról olvasva ugyanazt olvasuk-e. Így nem kell outputszalag, ezt az S állapothalmaznak ELVET és az ELFOGAD eleme helyettesíti.

Két Turing-gépet/algorithmust vázolunk. Ennek során elmondjuk, hogyan néznek ki a Turing-gépről készített pillanatfelvételek és szemezgetünk az állapothalmazból. Az egyszerűség kedvéért feltesszük, hogy $\Sigma = \{0, 1\}$.

1. algoritmus/Turing-gép

Ez egy egyszalagos Turing-gép lesz a $\Gamma = \{0, 1, 0^\vee, 1^\vee\}$ munkaábécével.

A START állapotból egyet jobbra lép az input szem/kéz (a szalaghatároló jel utáni első mező, az input első karaktere felett lesz). Az új állapot ELŐL-PIPÁL lesz: A karaktert „megjegyzi”, felülírja pipált változatával és megkeresi az utolsó inputkaraktert. Ehhez HÁTUL-TESZT-0, HÁTUL-TESZT-1 állapotokat használjuk. Ekkor az input szem/kéz folyamatosan jobbra mozog. Ez akkor áll le, amikor az inputszalagon az \triangleleft jel nem olvasható, majd egyet visszalép (ezt a későbbiekben egy kissé felülírjuk). Ekkor megtalálta a karaesett karaktert és teszteli, hogy megegyezik-e az első karakterrel: HÁTUL-TESZT-0-MOST, HÁTUL-TESZT-1-MOST állapotba kerül. Az állapot bitje az előlről hozott „emlékezet”. Ha a látott bit nem egyezik a hozott emlékezettel, akkor a gép ELVET állapotba kerül, leáll. Ha egyezik, akkor HÁTUL-PIPÁL állapotba kerül: felülírja a karaktert a pipált változatával és egyet balra lép. Az olvasott karaktertől függően ELŐL-TESZT-0, ELŐL-TESZT-1 állapotba kerül: balra megy, amíg el nem ér egy pipált karaktert és ennek elérésekor visszalép. Ezzel ELŐL-TESZT-0-MOST és ELŐL-TESZT-1-MOST állapotok egyikébe kerül, ahol a bit a „hátról hozott emlékezet”. Vagy leállunk, vagy pipálunk és egyet jobbra lépünk. Az itt látott karakter alapján HÁTUL-TESZT-0, HÁTUL-TESZT-1 állapotok egyikébe kerülünk. (Az állapothalmaz egy kis memóriát szimulál.) Korábban azt mondtuk, hogy ekkor az utolsó input karaktert keressük meg. Most pontosítunk: folyamatosan jobbra mozog az utolsó pipálatlan karakterig. Azaz a mozgás akkor áll le, amikor az inputszalagon az \triangleleft jel vagy pipált jel található, majd egyet visszalép.

A Turing-gép eddigi munkájához következő állapothalmazt használtuk

$$S = \{\text{START, HÁTUL-PIPÁL, HÁTUL-TESZT-0, HÁTUL-TESZT-1, HÁTUL-TESZT-0-MOST, HÁTUL-TESZT-1-MOST, ELŐL-PIPÁL, ELŐL-TESZT-0, ELŐL-TESZT-1, ELŐL-TESZT-0-MOST, ELŐL-TESZT-1-MOST, ELFOGAD, ELVET}\}.$$

Az elfogadó leálláshoz ismét módosítunk a korábbi (az általános teendőket mutató) leíráson: Ha az HÁTUL-TESZT-0, HÁTUL-TESZT-1, ELŐL-TESZT-0, ELŐL-TESZT-1 állapotba jutáskor egy pipált karaktert látunk, akkor a fenti „generikus” utasítás leírás helyett ELFOGAD állapotba kerülünk: Ez a helyzet/konfiguráció azt jelenti, hogy a bal és jobb pipák „összeértek”, inputunk palindrom szó. Az átmeneti függvény formalizálását a fenti „mesélő leírás” alapján az érdeklődő hallgatóra bízunk.

Könnyű látni, hogy minden ω inputon a fenti gép futása $\mathcal{O}(|\omega|^2)$. Ha ω egy palindrom szó, akkor a fenti T' gép futásának hosszát könnyen kiszámíthatjuk és ennek nagyságrendje $|\omega|^2$ lesz.

A konstansokat nem számoltuk ki. Lényegtelen is, az állapotok számának növelésével a futási idő csökkenthető. Például használhatnánk a HÁTUL-TESZT-000, HÁTUL-TESZT-001, HÁTUL-TESZT-010, HÁTUL-TESZT-011, HÁTUL-TESZT-100, HÁTUL-TESZT-101, HÁTUL-TESZT-110, HÁTUL-TESZT-111 állapotokat

input bitek hármassainak együttes tesztelésére és kevesebbet kellene „ingázni” a gépnek.

A fenti algoritmus nem hatékony. Az inputnak csak egy példánya áll rendelkezésre egy szemnek. Ez kényszeríti ki az ingázást. Ha van két szalagunk és két szemünk, akkor hatékonyabbak lehetünk.

2. algoritmus/Turing-gép

A standard modellt használjuk, $\Gamma = \Sigma = \{0, 1\}$, az állapothalmaz legyen

$$S = \{\text{START, MÁSOLÓK, MÁSOLÁS-KÉSZ, INPUTFEJ-ELŐRE, ELŐL-VAGYOK, TESZT, ELFOGAD, ELVET}\}.$$

Az inputot átmásoljuk a munkaszalagra. Ennek befejezésével MÁSOLÁS-KÉSZ állapotba jutunk.

Ebből mindkét szem balra egyet lép (a munka szem/kéz az átmásolt sorozat utolsó karaktere felett lesz), továbbá az INPUTFEJ-ELŐRE állapotba jut a gép. Ekkor a munkaszalag felett a szem/kéz mozdulatlan, az inputszalag feletti szem folyamatosan jobbra mozog. Egész addig, amíg a \triangleright jelet nem látja (ELŐL-VAGYOK állapot). Innen TESZT állapotba jut a gép az input szem eggyel jobbra mozgatása után (ekkor az input szem az input első karaktere fölé kerül, közben az output szem az átmásolt input utolsó karaktere felett maradt végig). A TESZT állapotban mindig ellenőrzi a gép, hogy a két szem ugyanazt látja-e. Ha valamikor ez nem teljesül, akkor ELVET állapotba jut, különben az input szem egyet jobbra, a munka szem egyet balra mozdul. Ez addig történik, amíg az input szem az input végét jelző karaktert nem látja (ekkor szükségszerű, hogy a munka szem a munkaszalag kezdetét jelző karakter felett legyen. Ha ez megtörténik, akkor a gép ELFOGAD állapotba kerül.

A fenti „szöveg” egyszerűen megfogalmazható az átmeneti függvény alkalmas definíciójával.

Minden ω inputra a futás hossza legfeljebb $3(n+1) = 3|\omega| + 3 = \mathcal{O}(|\omega|)$, ahol $n = |\omega|$, és az \mathcal{O} (olvasd „nagy ordó”) egy felső becslést jelöl rejtett szorzó és additív konstanssal.

Definíció. T Turing-gép időigénye egy $\omega \in \Sigma^*$ inputon $TIME(\omega; T) = \ell$, mely egy „csonkított” konfigurációsorozat hossza (azaz a konfigurációk végtelen sorozatában megállunk az első olyanánál, amelyben az állapot a számítás végét jelzi). Ekkor a futás $\{\kappa_i\}_{i=0}^{\ell}$, azaz az ℓ -edik konfigurációban kerül először STOP vagy döntési feladatnál ELFOGAD/ELVET állapotok.

Korábbi megállapítáunk az új jelöléssel kimondva

Minden $\omega \in \Sigma^*$ esetén $TIME(\omega; T) = \mathcal{O}(|\omega|)$, ahol T a fenti ismertetett, a PALINDROM nyelvet elfogadó gép.

Ez az eredmény a nagyságrend szempontjából éles. Pontosabban minden PALINDROM-ot kiszámító T Turing-gépre, van olyan ω input, amin T futása legalább $|\omega|$ hosszú. Feltéve, hogy $0 \in \Sigma$ a 0^n (n darab 0 karakter) esetén a gépnek el kell ezt fogadni, de ezt nem teheti meg az utolsó karakter elovasása nélkül. Ehhez viszont legalább n darab jobbra lépést kell tennie.

A második algoritmus fontos eleme volt, hogy kettő szemet (input és munka) használjunk, így ez az egyszalagos modellben **nem** valósítható meg.

Az algoritmusok időigényét is meghatároztjuk:

- Észrevétel.** 1. algoritmus (T_1 Turing-gép) négyzetes rendű: $TIME(\omega, T_1) \leq |\omega|^2$,
 2. algoritmus (T_2 Turing-gép) lineáris futásidejű: $TIME(\omega, T_2) \leq 3|\omega| + 3$.

Látjuk, hogy a második algoritmus, amely kisebb futásidejű, nem valósítható meg az egyszalagos modellen. A következő tétel szerint, ha egyszalagos modellt használó algoritmussal akarjuk eldönteni a *PALINDROM* nyelvet, akkor a négyzetes időigény lényegében nem javítható.

1. Tétel. *Ha T egy olyan Turing-gép, amely az egyszalagos modellben eldönti a PALINDROM nyelvet, akkor $\forall n, \exists \omega \in \Sigma^n$:*

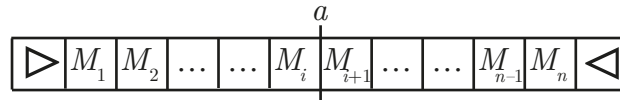
$$TIME(\omega, T) \geq \alpha_T |\omega|^2,$$

valamilyen α_T pozitív konstansra.

A tétel bizonyításához szükségünk lesz néhány új fogalomra.

Definíció.

Az inputszalag két szomszédos mezőjének közös határát *ajtónak* nevezzük. Ha az inputszalag mezőit úgy képzeljük el, mint végtelen egymásba nyíló szobasorozatot, akkor azt mondhatjuk, hogy a fej csak az ajtókon át tud közlekedni.



6. ábra. Az M_i és M_{i+1} mezőket elválasztó a ajtó.

Tekintsük egy T Turing-gép ω inputon való futását. Ekkor egy

$$\kappa_0(\omega) \rightarrow \kappa_1 \rightarrow \kappa_2 \rightarrow \dots \rightarrow \kappa_\ell$$

konfigurációsorozatot kapunk, ahol

$$\ell := \min\{n \mid T \text{ állapota } \kappa_n \text{-ben ELFOGAD vagy ELVET}\}$$

az időpont, amelyben eldöntjük, hogy ω eleme-e a *PALINDROM* nyelvnek. Ez az időpont biztosan véges, hiszen a *PALINDROM* nyelv az eldönthető nyelvek osztályába tartozik (ezt igazolják a korábban említett T_1, T_2 algoritmusok).

Most vegyük azokat a κ_j, κ_{j+1} konfigurációkat, amelyekben az inputszem egyszer az M_i , másszor az M_{i+1} mezőt nézi. Jelölje s_j a mezőket elválasztó a ajtón történő átlépéskor a Turing-gép állapotát. Ezen s_j állapotok sorozatát $\sigma(a, \omega)$ jelöli. Szemléletesen úgy képzelhetjük a $\sigma(a, \omega)$ sorozatot, hogy az a ajtóban egy őr áll, amely a fej minden áthaladásakor feljegyezi annak állapotát.

Nyilvánvaló, hogy a $\sigma(a, \omega)$ sorozatból kiolvasható az ajtón való áthaladás iránya, hiszen a fej balról érkezik, ezért a páratlan sorszámú állapotok a balról-jobbra (\rightarrow) történő átlépéskor, míg a páros sorszámú állapotok a jobbról-balra (\leftarrow) történő átlépéskor kerülnek feljegyzésre.

Jelölje $\omega \overset{a}{|}$ az ω input a ajtó előtti (tőle balra található) részét és $\overset{a}{|} \omega$ az ω input a ajtó utáni (tőle jobbra található) részét. Ha például az a ajtó a 6. ábrának megfelelően helyezkedik el, akkor $\omega \overset{a}{|} = \omega_1 \dots \omega_i$ és $\overset{a}{|} \omega = \omega_{i+1} \dots \omega_n$. Természetesen a két rész kiadja a teljes ω inputot: $\omega = \left(\omega \overset{a}{|}\right) \left(\overset{a}{|} \omega\right)$.

Észrevétel. Ha ismerjük az $\omega \overset{a}{|}$ inputrészletet és a $\sigma(a, \omega)$ állapotsorozatot, akkor meg tudjuk mondani, hogy a Turing-gép „hogyan működik”, amikor a szem az a ajtó előtti mezőket pásztázza. Azt nem tudhatjuk, hogy a szem meddig volt a jobb oldalán, de amint átlépi az ajtót (és amíg a bal oldalon marad) képesek vagyunk T futását leírni. Természetesen ugyanez elmondható az $\overset{a}{|}\omega$ darabra is; ekkor az ajtó jobb oldalán ismerjük T lépéseit. Az alábbi animáció illusztrálja az inputszem mozgását és az ajtó elválasztó szerepét. A megfigyelt mezők pirosak, ha $\omega \overset{a}{|}$ elemei és kékek, ha az $\overset{a}{|}\omega$ részei. A $\sigma(a, \omega)$ állapotsorozat a színváltások előtti állapotokból tevődik össze.

2. Következmény. Legyenek $\omega, \omega' \in \Sigma^n$ tetszőleges inputok és a egy ajtó. Tegyük fel, hogy $\sigma(a, \omega) = \sigma(a, \omega')$ és a T Turing-gép futásának eredménye megegyezik a két inputon. Ekkor az $\tilde{\omega} = \left(\omega \overset{a}{|}\right) \left(\overset{a}{|}\omega'\right)$ inputon is ugyanazt számolja ki T . Valójában ennél többet is mondhatunk, hiszen az ajtón történő áthaladások számától függ, hogy melyik inputon (ω -n vagy ω' -n) fejeződik be a futás.

Definíció. Tegyük fel, hogy $3 \mid n$. Legyen

$$I_0 := \{\alpha 0^{\frac{n}{3}} \overleftarrow{\alpha} : \alpha \in \Sigma^{\frac{n}{3}}\} \subseteq \text{PALINDROM} \cap \Sigma^n.$$

Tehát I_0 azon n hosszú palindrom szavakból áll, amelyekben egy $n/3$ hosszú szó és fordítottja $n/3$ nullát fog közre.

Megjegyzés. Mivel egy $\alpha 0^{\frac{n}{3}} \overleftarrow{\alpha} \in I_0$ elemet egyértelműen meghatároz az α szó, ezért $|I_0| = |\Sigma|^{\frac{n}{3}} = |\{0, 1\}|^{\frac{n}{3}} = 2^{\frac{n}{3}}$.

3. Következmény. Legyenek $\omega, \omega' \in I_0$ különböző szavak és a egy középső ajtó (azaz valamelyik a középső $n/3$ nullát elválasztó ajtó közül). Ekkor $\sigma(a, \omega) \neq \sigma(a, \omega')$.

Bizonyítás. Indirekt módon tegyük fel, hogy $\sigma(a, \omega) = \sigma(a, \omega')$. Ekkor az előző következmény alapján, ha mindkét inputon *ELFOGAD* állapottal áll le a Turing-gép, akkor az $\tilde{\omega} = \left(\omega \overset{a}{|}\right) \left(\overset{a}{|}\omega'\right) = \alpha 0^{\frac{n}{3}} \overleftarrow{\alpha'}$ inputot is elfogadja, vagyis $\tilde{\omega} \in I_0$. Azonban $\omega \neq \omega'$, így $\alpha \neq \alpha'$, amiből $\tilde{\omega} \notin I_0$ következik. Ez az ellentmondás igazolja az állítást. ■

Észrevétel. Ha feltételezzük, hogy $|\Sigma| \geq 2$ és $|S| \geq 3$, akkor a t -nél rövidebb állapotsorozatok számára az

$$1 + |S| + \dots + |S|^{t-1} = \frac{|S|^t - 1}{|S| - 1} < |S|^t - 1 < |S|^t$$

felső becslés adható.

4. Következmény. Ha $|I_0| = |\Sigma|^{\frac{n}{3}} \geq |S|^t$, akkor $\exists \omega \in I_0$, hogy $\sigma(a, \omega)$ hossza legalább t , ahol a egy középső ajtó.

Bizonyítás. Indirekt módon tegyük fel, hogy nincs ilyen ω . Ekkor bármely $\omega \in I_0$ és a középső ajtó esetén $\sigma(a, \omega)$ hossza kisebb, mint t . Ez $|I_0| > |S|^t$ állapotsorozatot jelent. Azonban a t -nél rövidebb állapotsorozatok számára adott becslés alapján, mint $|I_0| < |S|^t$. Ez ellentmondás, tehát az állításban szereplő ω input létezik. ■

Megjegyzés. Az előbbi következményből az is leolvasható, hogy $|I_0| = |\Sigma|^{\frac{n}{3}} \geq |S|^t$ esetén $t \sim \beta_T \cdot n$.

5. Következmény. Ha $|I_0| \geq 2|S|^t$, akkor létezik legalább $|I_0|/2$ olyan szó I_0 -ban, amelyre $|\sigma(a, \omega)| \geq t$ és mindegyik különböző állapotsorozatot ad, amikor kiszámítjuk. Ezek halmazát jelölje I_1 . Tehát

$$I_1 = \{\omega \in I_0 : |\sigma(a, \omega)| \geq t\} \subseteq I_0.$$

Az előbbi megjegyzés alapján $t \sim \gamma_T \cdot n$, alkalmas γ_T konstansra.

Ezek után térjünk rá az 1. Tétel bizonyítására

Bizonyítás. Azok az időpontok nyilván nem relevánsak, amikor a fej nem mozdul, ezért az alábbi alsó becslés adható

$$\sum_{\omega \in I_0} \text{TIME}(\omega, T) \geq \sum_{\omega \in I_0} \sum_{\substack{a \text{ középső} \\ \text{ajtó}}} |\{t : t \text{ időpillanatban a fej átlép } a\text{-n}\}|$$

az összegben megjelenő halmaz számossága $\sigma(a, \omega)$ definíciója alapján $|\sigma(a, \omega)|$,

$$= \sum_{\omega \in I_0} \sum_{\substack{a \text{ középső} \\ \text{ajtó}}} |\sigma(a, \omega)| = \sum_{\substack{a \text{ középső} \\ \text{ajtó}}} \sum_{\omega \in I_0} |\sigma(a, \omega)|$$

ahol kettős összegben a szokásos sorrendcsere történt. Az 5. Következmény alapján

$$\geq \sum_{\substack{a \text{ középső} \\ \text{ajtó}}} \sum_{\omega \in I_1} t = \sum_{\substack{a \text{ középső} \\ \text{ajtó}}} |I_1| \cdot t \geq \sum_{\substack{a \text{ középső} \\ \text{ajtó}}} \frac{|I_0|}{2} \cdot t$$

mivel a középső ajtók száma $n/3$, ezért ez

$$= \frac{n}{3} \cdot \frac{|I_0|}{2} \cdot t = \frac{\gamma_T}{6} \cdot n^2 \cdot |I_0|$$

ahonnan $|I_0|$ -val való osztás után

$$\frac{1}{|I_0|} \sum_{\omega \in I_0} \text{TIME}(\omega, T) \geq \frac{\gamma_T}{6} \cdot n^2$$

adódik. Tehát azt kaptuk, hogy az átlagos futásidő négyzetesen függ az input hosszától. A γ_T konstans függ a Turing-géptől, ezáltal értéke csökkenthető, de a négyzetes jelleg megmarad. ■