Математичка гимназија

Матурски рад
из информатике

Кратак увод у теорије израчунљивости и сложености

Ученик
Филип Бојковић, IVд

Ментор
проф Јелена
Хаџи-Пурић

Београд, *27.5.2024.*

Садржај

1

Увод

У овом раду ћемо се бавити теоријским основама информатике. Алгоритми представљају веома интуитиван концепт и из тог разлога нема потребе да се уводе формално ради решавања већине практичних проблема из информатике, међутим, у овом раду ћемо имати другачији приступ. Видећемо како нам формална дефиниција алгоритама допушта да доказујемо теореме о самим границама њихових могућности кроз неке интересантне проблеме.

2

Уводни појмови

Дефиниција алгоритма коју ћемо користити у целости овог рада је Тјурингова машина. По Черч–Тјуринговој тези она, заједно са још пар других модела, одговара нашој интуитивној представи алгоритма.

2.1 Детерминистичка Тјурингова машина

Дефиниција 2.1. *Детерминистичка Тјурингова машина* (ДТМ) је уређена седморка $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{пр}}, q_{\text{одб}})$:

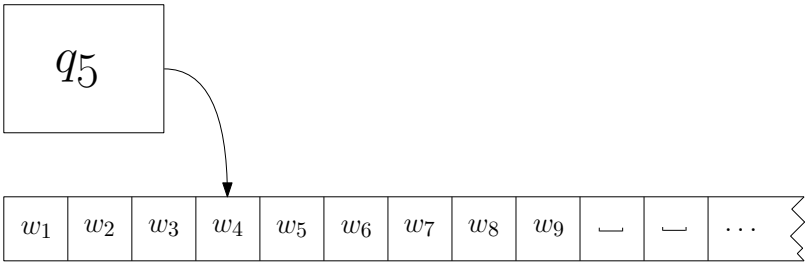
1. Q је коначан скуп стања
2. Σ је алфабет улаза и садржи празан карактер „␣”
3. Γ је алфабет траке и важи $\Sigma \subseteq \Gamma$
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, D\}$ је функција прелаза
5. $q_0 \in Q$ је почетно стање
6. $q_{\text{пр}} \in Q$ је стање прихватања
7. $q_{\text{одб}}$ је стање одбијања и $q_{\text{пр}} \neq q_{\text{одб}}$

Детерминистична Тјурингова машина се састоји из контролне јединице, која се налази у неком стању q, и главе која показује на неко место на бесконачној траци, ограниченој са леве стране.

На почетку глава показује на почетак траке, контролна јединица је у стању q_0 и улаз $w = w_1 w_2 \dots w_n \in \Sigma^*$ се налази на почетку траке, док је остатак траке попуњен празним карактерима.

Ако се у неком тренутку машина налази у стању q, глава чита карактер w_i и $\delta(q, w_i) = (q', w_i', S)$ онда машина на место w_i уписује карактер w_i', прелази у стање q' и помера главу у смеру S (ако је $S = L$ улево, у супротном удесно).

Ако машина уђе у стање $q_{\text{пр}}$ она се зауставља и кажемо да је прихватила улаз w. Ако уђе у стање $q_{\text{одб}}$ она се зауставља и кажемо да је одбила улаз w. Ако машина никада не уђе у $q_{\text{пр}}$ или у $q_{\text{одб}}$ кажемо да је одбила улаз w „вртењем".



Слика 2.1. Илустрација ДТМ чија глава показује на w_4 у стању q_5

Дефиниција 2.2. Језик детерминистичке Тјурингове машине M означавамо као $L(M)$ и он представља скуп $\{w | M \text{ прихвата } w\}$. Кажемо да M препознаје језик $L = L(M)$ и да је M *L-препознавач*.

Дефиниција 2.3. Детерминистичка Тјурингова машина M је *одлучивач* ако се заустави при сваком улазу. Ако је $L = L(M)$, M називамо *L-одлучивачем*

Дефиниција 2.4. Неки језик L је *Тјуринг-препознатљив* ако постоји L-препознавач.

Дефиниција 2.5. Неки језик L је *Тјуринг-одлучив* ако постоји L-одлучивач.

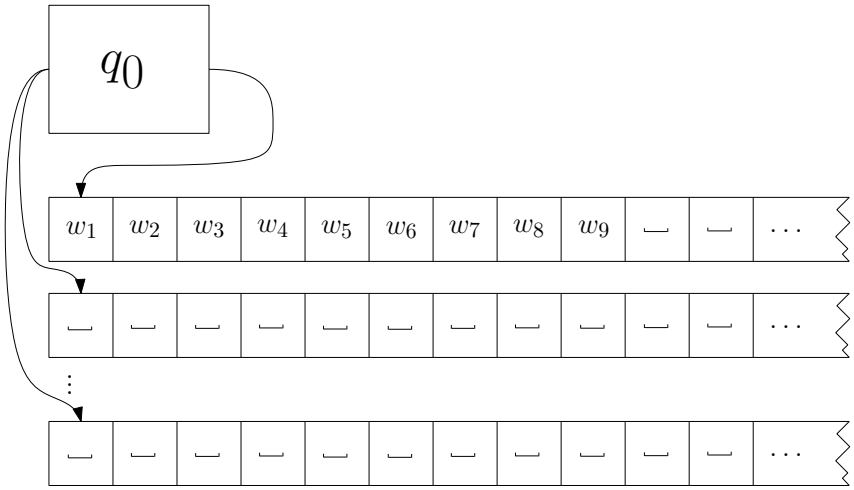
Дефиниција 2.6. $\overline{A} = \Sigma^* \setminus A$ је комплемент језика A.

2.2 Детерминистичка Тјурингова машина са више трака

Дефиниција 2.7. *Детерминистичка Тјурингова машина са више трака* (ДТМВТ) је уређена седморка $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{пр}}, q_{\text{одб}})$ при чему је једина разлика са дефиницијом ДТМ та да је $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, D, O\}^k$, где је k број трака.

Ова машина се састоји из k трака и k глава. У сваком кораку машина чита симултано са свих k трака и за сваку главу одлучује који ће карактер написати и да ли ће се она померити лево, десно или остати ту где је.

На почетку се улаз налази на првој траци, док су остале попуњене празним карактерима.



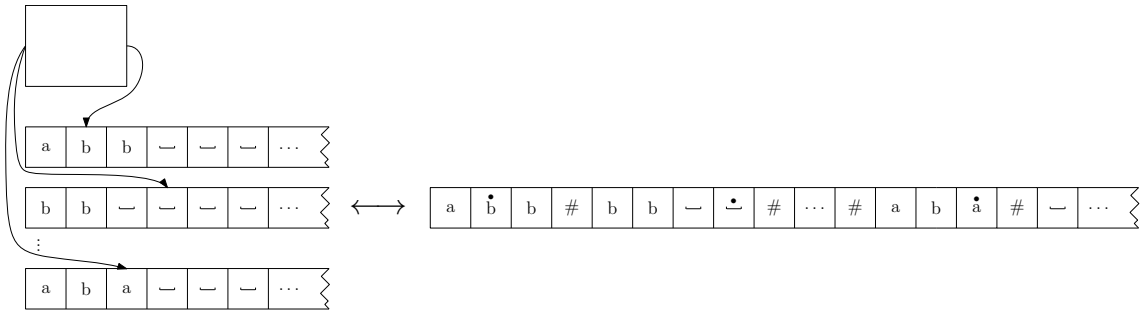
Слика 2.2. Илустрација ДТМВТ на почетку израчунавања са улазом дужине 9

Теорема 2.1. ДТМ и ДТМВТ препознају исте језике.

Доказ. Очигледно је да ДТМВТ могу препознати све језике које препознају ДТМ само некоришћењем осталих трака.

Сада ћемо конструисати ДТМ M_1 која препознаје исти језик као произвољна ДТМВТ M_k са k трака.

Садржаје трака M_k ћемо записати на једној траци M_1 једне до других раздвојене специјалним карактером „#” занемарујући бесконачно много празних карактера са десне стране сваке траке. Такође, за сваки карактер оригиналног алфабета траке (Г) ћемо додати специјалан карактер који означава да ли нека од глава M_k показује на ту позицију. (Слика 2.3)



Слика 2.3. Илустрација записивања садржаја трака M_k у M_1

Сада ћемо описати како ће машина M_1 да симулира машину M_k:

M_1 = „За улаз w:

1. Понавља за сваки корак M_k:

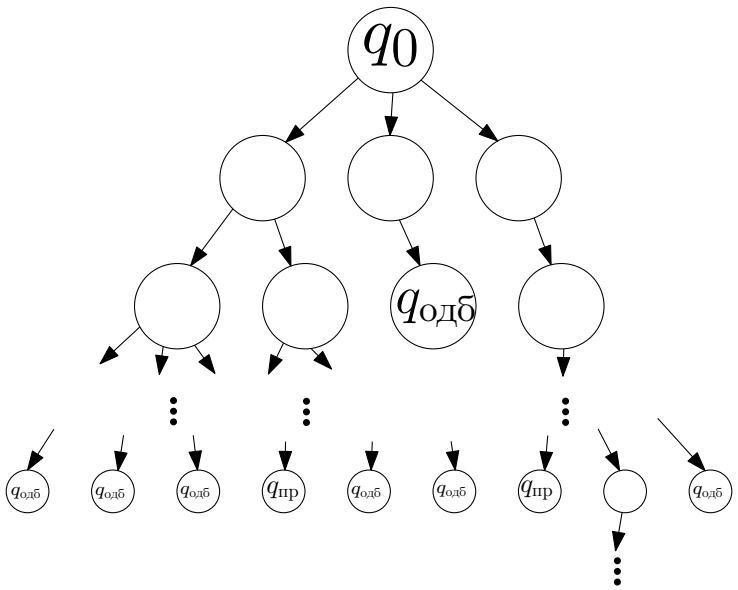
    1. Пролазом са почетка на крај траке у својој коначној меморији (стањима машине) памти које карактере читају главе M_k.
    2. Прелази у стање извршавања корака на основу очитаних карактера. Проласком са почетка на крај траке помера сваку главу и записује одговарајуће карактере. Ако је потребно да помери главу неке траке удесно на позицију на којој се налази #, прелази у помоћно стање померања свих карактера после те позиције за једно место удесно.
    3. Ако је ново стање M_k стање прихватања, прихвата улаз, ако је стање одбијања, одбија улаз."

Машина M_1 може у својој коначној меморији да памти тренутно стање M_k као и карактере које читају главе M_k, јер стања, трака и карактера у алфабету траке има коначно много. □

2.3 Недетерминистичка Тјурингова машина

Дефиниција 2.8. *Недетерминистичка Тјурингова машина* (НТМ) је уређена седморка $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{пр}}, q_{\text{одб}})$ при чему је једина разлика са дефиницијом ДТМ та да је $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, D\})$

У току рада машине се паралелно прави по једна копија машине за свако $(q', w_i', S) \in \delta(q, w_i)$ и оне настављају независно да раде.(Слика 2.4) Кажемо да је улаз w прихваћен ако га барем једна од ових *нити* рачуна прихвати, у супротном је одбијен.



Слика 2.4. Илустрација нити НТМ

Дефиниција 2.9. Недетерминистичка Тјурингова машина M је одлучивач ако се свака нит рачунања заустави на сваком улазу.

Теорема 2.2. ДТМ и НТМ препознају исте језике.

Доказ. Очигледно је да НТМ могу препознати све језике које препознају ДТМ само неправљењем нових нити.

Сада ћемо конструисати ДТМВТ M' која препознаје исти језик као произвољна НТМ M. Машина M' ће имати 3 траке, на првој ће се налази улаз, на другој ће се налазити тренутно стање траке једне нити машине M, а на трећој ће се налазити опис тренутне нити. Опис нити је низ карактера из скупа $\{1, 2, \ldots, |Q \times \Gamma \times \{L, D\}|\}$, где нам сваки карактер описује коју од потенцијално $|Q \times \Gamma \times \{L, D\}|$ много опција пратимо при сваком гранању. Машина M' ће радити следеће.

M' = „За улаз w на првој траци:

1. Понавља следеће кораке:
    1. На трећу траку постави следећи по реду опис нити. Прво поставља краће описе, ако су описи исте дужине, пре је онај који је лексикографски мањи читано уназад.
    2. Прекопира улаз са прве на другу траку, и симулира нит машине M са треће траке на другој траци.
        - Ако у неком тренутку функција прелаза не предвиђа опцију која стоји у опису нити, прелази на следећу нит.
        - Ако у неком тренутку понестане корака на трећој траци, прелази на следећу нит.
        - Ако у неком тренутку нит одбије улаз, M' прелази на следећи опис нити.
        - Ако у неком тренутку нит прихвати улаз, M' прихвата улаз.”

Приметимо да симулација сваког описа нити траје коначно много, јер је сваки опис коначне дужине. Машина M' једино прихвата улаз ако и нека нит M прихвата улаз. Такође, ако нека нит M' прихвата улаз у k корака, онда ће машина M проћи кроз највише $(|Q \times \Gamma \times \{L, D\}| + 1)^k$ симулација нити, док не дође до оне прихватајуће.

Овиме смо показали да НТМ и ДТМВТ препознају исте језике, а како по теореми 2.1 ДТМ и ДТМВТ препознају исте језике, тако и НТМ и ДТМ препознају исте језике. □

3

Теорија израчунљивости

Дефиниција 3.1. Ознака $\langle A_1, A_2, \ldots, A_n \rangle$ означава репрезентацију података $A_1, \ldots, A_n$ у алфабету улаза.

Дефинишимо језик који се често користи за доказивање неодлучивости других језика.

Дефиниција 3.2. $A_{TM} = \{\langle M, w \rangle | \text{ДТМ } M \text{ прихвата } w\}$

3.1 Проблем прихватања улаза

Проблем 1. Да ли постоји ДТМ која одлучује језик A_{TM}?

Теорема 3.1. Језик A_{TM} је Тјуринг-препознатљив.

Скица доказа. Конструишемо ДТМ M_1 која препознаје A_{TM}.

$M_1 =$ „За улаз $\langle M, w \rangle$:

1. Симулира M на улазу w:
   - Ако је M прихватио w, прихвата улаз.
   - Ако је M одбио w, одбија улаз.”

Нећемо показивати да Тјурингова машина може да симулира произвољну Тјурингову машину дату на улазу. Таква Тјурингова машина се зове универзална Тјурингова машина.

Ако M прихвата w, онда и M_1 прихвата $\langle M, w \rangle$.

Ако M одбија w улажењем у стање одбијања, онда и M_1 одбија $\langle M, w \rangle$.

Ако M одбија w „вртењем”, онда ће се и M_1 „вртети” и одбити $\langle M, w \rangle$. □

Теорема 3.2. Језик A_{TM} није Тјуринг-одлучив.

Доказ. Претпоставимо супротно, да постоји ДТМ M_1 која одлучује језик A_{TM}. Изградимо сада машину M_2 на следећи начин.

M_2 = „За улаз $\langle M \rangle$:

1. Симулира M_1 на улазу $\langle M, M \rangle$:
   - Ако је M_1 прихватио $\langle M, M \rangle$, одбија улаз.
   - Ако је M_1 одбио $\langle M, M \rangle$, прихвата улаз."

Машина M_2 ће увек извршити инструкцију 1, јер је M_1 одлучивач.

Ако машина M_2 прихвата улаз $\langle M_2 \rangle$, онда M_1 прихвата улаз $\langle M_2, M_2 \rangle$, али тада M_2 одбија улаз $\langle M_2 \rangle$. Контрадикција.

Слично, ако машина M_2 одбија улаз $\langle M_2 \rangle$, онда M_1 одбија улаз $\langle M_2, M_2 \rangle$, али тада M_2 прихвата улаз $\langle M_2 \rangle$. Контрадикција.

Овиме је полазна претпоставка нетачна, стога не постоји ДТМ која одлучује језик A_{TM}. □

3.2 Проблем неприхватања улаза

Теорема 3.3. Ако је језик A одлучив, онда је и језик $\overline{A}$.

Доказ. Ако ДТМ M одлучује језик A. Заменом $q_{\text{пр}}$ и $q_{\text{одб}}$ машине M добијамо ДТМ M' која одлучује $\overline{A}$. □

У овом одељку ћемо посматрати језик $\overline{A_{TM}}$. Приметимо да не можемо изградити ДТМ која препознаје $\overline{A_{TM}}$ као у теореми 3.3 због могућности одбијања „вртењем".

Проблем 2. Да ли постоји ДТМ која препознаје језик $\overline{A_{TM}}$?

Теорема 3.4. Ако су језици A и $\overline{A}$ препознатљиви, онда је A одлучив.

Доказ. Нека су M_1 и M_2 редом A и $\overline{A}$ препознавачи. Конструишемо ДТМВТ M са 2 траке која одлучује A.

M = „За улаз w:

1. Препише w на другу траку.
2. Паралелно симулира M_1 на горњој траци и M_2 на доњој траци:
   - Ако M_1 прихвати w, онда M прихвата улаз и излази.
   - Ако M_2 прихвати w, онда M одбија улаз и излази."

Ако је $w \in A$, M_1 ће прихвата улаз w у коначно много корака, па ће и M прихватити w у коначно много корака.

Ако $w \notin A$, M_2 ће прихватити улаз w у коначно много корака, па ће M одбити w у коначно много корака.

Овиме смо показали да се M зауставља на сваком улазу, стога је одлучивач. □

Последица 3.1. $\overline{A_{TM}}$ је непрепознатљив.

Доказ. Како знамо да је A_{TM} неодлучив и препознатљив по теоремама 3.2 и 3.1, по теореми 3.4 следи да је $\overline{A_{TM}}$ непрепознатљив. □

Слика 3.1. Илустрација скупова одлучивих и препознатљивих језика по теоремама 3.3 и 3.4

3.3 Постов проблем кореспонденције

Претходна два проблема су била везана за саму „природу" Тјурингове машине, па је и било разумно очекивати да нас доведу до саме границе њихових могућности. Сада ћемо посматрати један проблем који не поставља питање о самим Тјуринговим машинама, доказати да је неодлучив и демонстрирати моћну технику за доказивање неодлучивости, чију ћемо идеју и касније користити за доказивање теорема у поглављу 4.

Проблем 3 (**Модификовани Постов проблем кореспонденције**)**.** Дат је коначан скуп уређених парова речи (домина) $S = \{(p_1, q_1), \ldots, (p_k, q_k)\} \subset \{(p, q) | p, q \in \Sigma^*\}$. Рецимо да је упаривање скупа S коначан низ $(i_1, \ldots, i_n)$, такав да је $p_{i_1} p_{i_2} \ldots p_{i_n} = q_{i_1} q_{i_2} \ldots q_{i_n}$ и $i_1 = 1$. Да ли постоји ДТМ која одлучује језик $\{\langle S \rangle | \text{постоји упаривање } S\}$?

Једина разлика са правим Постовим проблемом кореспонденције је та да важи $i_1 = 1$.

$$S = \{\binom{\text{ab}}{\text{a}}, \binom{\text{aa}}{\text{ba}}, \binom{\text{aa}}{\text{a}}, \binom{\text{a}}{\text{aaab}}\}$$

$$\left|\begin{matrix}\text{a} & \text{b} & \text{a} & \text{a} & \text{a} & \text{a} & \text{b} & \text{a} & \text{a}\\ \text{a} & \text{b} & \text{a} & \text{a} & \text{a} & \text{a} & \text{b} & \text{a} & \text{a}\end{matrix}\right|$$

Слика 3.2. Илустрација скупа домина и могућег упаривања

Како би се описала нека ДТМ у произвољном тренутку, довољно је имати позицију главе, садржај траке и стање машине. Ове информације ћемо компактно записивати тако што ћемо додати карактер који описује стање машине одмах испред позиције на траци на коју показује глава. (Слика 3.3)

$$w_1w_2w_3q_5w_4w_5w_6w_7w_8w_9$$

Слика 3.3. Пример описа машине са слике 2.1

Дефиниција 3.3. Историја рачунања ДТМ је конкатенација описа машине у сваком тренутку рачунања раздвојених специјалним карактерима, које ћемо означавати са #. Прихватајућа историја је она при којој је машина прихватила улаз (Слика 3.4).

$$\#q_000\#1q_30\#10q_3\_\#101q_{\text{пр}}\_\#$$

Слика 3.4. Пример прихватајуће историје неке машине

Теорема 3.5. Модификовани Постов проблем кореспонденције је неодлучив.

Доказ. Показаћемо да пажљивом конструкцијом скупа S можемо одговорити на питање да ли постоји прихватајућа историја рачунања произвољне ДТМ решавањем проблема 3. Када би проблем 3 био одлучив, то би значило да је и A_{TM} одлучив, праћењем посткупка за конструкцију одговарајућег скупа S и покретањем решења проблема 3 на том S. Пошто знамо, по теореми 3.2, да је A_{TM} неодлучив, мора бити да је и 3. Остаје само да опишемо поступак конструисања таквог скупа S.

За ДТМ $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{пр}}, q_{\text{одб}})$ скуп S конструишемо као унију следећих скупова:

1. $\{(\#, \#q_0w_1 \dots w_n\#)\}$ је почетни пар. За случај да је w празан уместо ове додаје се домина $(\#, \#q_0\_\#)$.

2. $\{(qa, bq')|\delta(q, a) = (q', b, D) \wedge q \neq q_{\text{одб}}\}$ одговара свим случајевима када се глава помера удесно.

3. $\{(cqa, q'cb)|\delta(q, a) = (q', b, L) \wedge q \neq q_{\text{одб}}\}$ одговара свим случајевима када се глава помера улево.

4. $\{(a, a)|a \in \Gamma\} \cup \{(\#, \#)\}$ служи да би се преписао део траке који није директно уз главу и који се не мења.

5. $\{(\#, \textvisiblespace\#)\}$ служи како би се додао празан карактер са леве стране за случај да глава сада њега чита.

6. $\{(aq_{\text{пр}}, q_{\text{пр}}), (q_{\text{пр}}a, q_{\text{пр}})\}$ служи како би након што се машина зауставила могло да се настави спајање све док у доњем делу не остане само „$q_{\text{пр}}\#$” као вишак.

7. $\{(q_{\text{пр}}\#\#, \#)\}$ се додаје на крај како би се употпунило упаривање.

Јасно је да ако M прихвата улаз w, постојаће и упаривање на овом скупу S. Такође, ако постоји упаривање на S, оно одговара прихватајућој историји M.

Овом конструкцијом смо показали да одлучивост проблема 3 повлачи и одлучивост A_{TM}, стога је, по теореми 3.2, проблем 3 неодлучив. □

4

Теорија сложености

4.1 Временска сложеност

Дефиниција 4.1. Нека је M ДТМ или ДТМВТ која је одлучивач. *Временска сложеност најгорег случаја* машине M је функција $f : \mathbb{N} \to \mathbb{N}$, где је $f(n)$ **највећи** број корака који машина M изврши на неком улазу дужине n.

Није увек јасно колика је дужина улаза. На пример, када је улаз $\langle M \rangle$ за неку ДТМ M. Из тог разлога, треба имати на уму да временска сложеност битно зависи од начина на који је улаз енкодиран. На пример, није исто ако бројеве на траци записујемо у некој основи већој од 1 или као низ нула. Из овог разлога ће у даљем тексту бити увек наглашено како је улаз представљен. Ако није другачије наглашено n ће увек означавати број карактера на улазу у датом енкодирању.

Дефиниција 4.2. Нека је M НТМ која је одлучивач. *Временска сложеност најгорег случаја* M је функција $f : \mathbb{N} \to \mathbb{N}$, где је $f(n)$ **највећи** број корака који нека нит машине M изврши на неком улазу дужине n.

Пошто ова функција може бити јако компликована, бавићемо се само њеном *асимптотском величином*. За то ћемо користити нотацију *велико–$\mathcal{O}$* и *велико-*Ω.

Дефиниција 4.3. Нека су $f, g : \mathbb{N} \to \mathbb{R}^+$ функције. Кажемо да је $f(x) = \mathcal{O}(g(x))$ ако $(\exists x_0 \in \mathbb{N})(\exists c \in \mathbb{R}^+)(\forall x \in \mathbb{N})(x > x_0 \implies f(x) \leq cg(x))$

Дефиниција 4.4. Нека су $f, g : \mathbb{N} \to \mathbb{R}^+$ функције. $f(x) = \Omega(g(x)) \iff g(x) = \mathcal{O}(f(x))$

Дефиниција 4.5. Временску сложеност која је облика $\mathcal{O}(n^k)$ за неко k називамо *полиномијалном*.

Пример 4.1. Посматрајмо језик $\{0^k1^k2^k|k \in \mathbb{N}\}$. Показаћемо да је он одлучив у $\mathcal{O}(n^2)$ времена детерминистичком Тјуринговом машином, где је n број карактера на улазу.

$M =$ „За улаз w:

1. Проверава да ли је w облика $0^k1^l2^m$ за неке k, l и m у $\mathcal{O}(n)$ времена

2. Понавља све док сви карактери нису прецртани:

    1. У једном пролазу са почетка до краја улаза прецрта једну 0, једну 1 и једну 2. Ако у неком тренутку не може да прецрта одговарајући карактер, одбија улаз.

3. Прихвата улаз”

Корак 1 траје $\mathcal{O}(n)$ времена. Корак 2 се понавља $\mathcal{O}(n)$ пута и сваки пут у себи има $\mathcal{O}(n)$ корака. Одавде следи да је сложеност овог алгоритма $\mathcal{O}(n^2)$.

Пример 4.2. Сада ћемо показати да је језик $\{0^k1^k2^k|k \in \mathbb{N}\}$ одлучив у $\mathcal{O}(n \log n)$ времена детерминистичком Тјуринговом машином, где је n број карактера на улазу.

$M =$ „За улаз w:

1. Проверава да ли је w облика $0^k1^l2^m$ за неке k, l и m у $\mathcal{O}(n)$ времена

2. Понавља све док сви карактери нису прецртани:

    1. У једном пролазу са почетка до краја улаза прецрта сваку другу непрецртану нулу, сваку другу јединицу и сваку другу двојку, док прати парност бројева нула, јединица и двојки. Ако се у неком тренутку не подударају парности нула, јединица и двојки, одбија улаз.

3. Прихвата улаз.”

За овај пример је мање очигледно зашто M препознаје овај језик. Наиме, ако посматрамо бинарну репрезентацију броја нула, јединица и двојки, машина M заправо у свакој интерацији 2. корака проверава да ли се текуће цифре подударају. Ако се све цифре подударају, бројеви нула јединица и двојки су на почетку били једнаки.

Што се временске сложености тиче, у свакој итерацији 2. корака се број непрецртаних карактера преполови. Ово значи да ће се 2. корак понављати $\mathcal{O}(\log n)$ пута, а сваки пут траје $\mathcal{O}(n)$ времена. Овако смо показали да је $\mathcal{O}(n \log n)$ укупна временска сложеност машине M која одлучује тражени језик.

4.2 Класе П и НП

Теорема 4.1. Сваки проблем решив у полиномијалном времену на Тјуринговој машини са више трака је решив у полиномијалном времену и на Тјуринговој машини са једном траком.

Доказ. Подсетимо се доказа теореме 2.1 где смо конструисали ДТМ M_1 како бисмо симулирали ДТМВТ M_k. Анализирајмо сада која је временска сложеност машине M_1.

Машина M_k има временску сложеност $\mathcal{O}(n^l)$ за неко l. При симулирању сваког корака машине M_k, машина M_1 прође највише константан број пута кроз корисни део своје траке. Машина M_k је могла да повећа корисни део својих трака за највише $\mathcal{O}(n^l)$ током свог извршавања (за k у сваком кораку). Како машина M_1 симулира $\mathcal{O}(n^l)$ корака, тако да је њена укупна временска сложеност $\mathcal{O}(n^{2l})$.

Овиме смо показали да произвољну полиномијалну ДТМВТ можемо симулирати ДТМ у полиномијалном времену. □

Надахнуте теоремом 4.1, интересује нас да ли постоји некаква „полиномијална еквивалентност" између „разумних" модела израчунавања. Испоставља се да постоји, и да сви „разумни" модели решавају исте проблеме у полиномијалном времену, као и да све недетерминистичке верзије „разумних" модела решавају исте проблеме у полиномијалном времену. У овом раду нећемо доказивати ове еквивалентности (поред теореме 4.1), али ћемо се надаље више бавити питањем да ли се неки корак може урадити у полиномијалном времену, а не и која је његова конкретна сложеност.

Дефиниција 4.6. Нека је $t : \mathbb{N} \to \mathbb{R}^+$ функција. Класа временске сложености $VREME(t(n))$ је скуп свих језика који су одлучиви у времену $\mathcal{O}(t(n))$ неком ДТМ.

Дефиниција 4.7. Нека је $t : \mathbb{N} \to \mathbb{R}^+$ функција. Класа временске сложености $NVREME(t(n))$ је скуп свих језика који су одлучиви у времену $\mathcal{O}(t(n))$ неком НТМ.

Дефиниција 4.8. Класу временске сложености П дефинишемо као

$$\text{П} = \bigcup_{k=1}^{\infty} VREME(n^k)$$

Дефиниција 4.9. Класу временске сложености НП дефнишемо као

$$\text{НП} = \bigcup_{k=1}^{\infty} NVREME(n^k)$$

Проблем 4 (Ограничени модификовани Постов проблем кореспонденције). Посматрајмо поново проблем 3, али сада са додатним ограничењем, а то је да се питамо да ли постоји упаривање са највише k домина.

Број k је на улазу дат у основи 1. Домине су записане редом и одвојене специјалним карактерима. Укупан број карактера дат на улазу означавамо са n.

Пример 4.3. Конструишемо НТМ која решава проблем 4:

$M =$ „За улаз $\langle S, k \rangle$:

1. Понавља док низ домина није дужи од k:
    1. Проверава да ли је тренутни низ домина упаривање, ако јесте, прихвата улаз.
    2. Прави $\mathcal{O}(n)$ нових нити, при чему свака додаје другачију домину на тренутни низ домина. ”

Свака нит ће k пута додати нову домину и k пута извршити проверу, при чему се ове радње могу имплементирати у $\mathcal{O}(n^2)$ времена. Стога је укупна временска сложеност овог приступа $\mathcal{O}(n^3)$.

Пример 4.4. Конструишемо ДТМ која решава проблем 4.

Детерминистичка Тјурингова машина може слично као и недетерминистичка машина да генерише све конфигурације домина и да провери да ли је нека од њих упаривање. Међутим, за разлику од недетерминистичке машине, не може све конфигурације паралелно да проверава. Како њих има у најгорем случају $\Omega(2^k)$, тако ни овај алгоритам не може бити полиномијалан по k, односно n.

Последица 4.1. Проблем 4 се налази у класи НП, али не знамо да ли је у класи П.

Приметимо да иако нисмо успели да решимо проблем 4 у полиномијалном времену детерминистичком Тјуринговом машином, када би нам неко дао одговарајући низ домина (који је полиномијалне дужине у односу на оригиналан улаз) могли бисмо у полиномијалном времену детерминистичком машином да проверимо да ли је то решење проблема.

Теорема 4.2. За сваки проблем у НП и улаз који он прихвата, постоји *сертификат* полиномијалне дужине којим детерминистичка машина може у полиномијалном времену да потврди да се тај улаз заиста прихвата.

Доказ. За сертификат ћемо користити историју рачунања прихватајуће нити НТМ. Ако ова НТМ ради у времену $\mathcal{O}(n^k)$ за неко k, онда ће ова историја бити дужине $\mathcal{O}(n^{2k})$. Једино што детерминистичка машина треба да уради јесте да провери да ли је овај сертификат валидна историја рачунања, што се може урадити у полиномијалном времену. □

Теорема 4.3. Ако за неки проблем и сваки улаз који прихвата, постоји сертификат полиномијалне дужине којим детерминистичка машина може у полиномијалном времену да потврди да се тај улаз заиста прихвата, онда се тај проблем налази у НП.

Доказ. Како је сертификат дужине $\mathcal{O}(n^k)$, постоји $\mathcal{O}(n^k)$ могућих сертификата за све улазе дужине n. Можемо направити НТМ чије нити генеришу све могуће сертификате и за сваки проверавају у полиномијалном времену да ли доказује прихватање улаза. Генерисање сертификата се врши слично као у примеру 4.3. Ако је неки сертификат прихваћен, улаз се прихвата, ако није, улаз се одбија. □

Последица 4.2. Проблеми у НП су они чија решења можемо „брзо" проверити.

Још један начин за посматрање проблема из НП је да замислимо да их можемо решити у полиномијалном времену ако при сваком гранању недетерминистичког рачунања бацамо новчић за бирање нити коју ћемо наставити да пратимо и да нам новчић увек даје онај одговор који води у стање прихватања, ако оно постоји.

Природно се постављају питања:

1. Да ли су сви проблеми из НП решиви и у П?
2. Да ли је проблем 4 решив у П?
3. Које још проблеме можемо много лакше решити у НП него у П?

Нажалост, на питање 1 нико нема одговор. У наставку ћемо питање 1 свести на питање 2 и видети још један проблем који има својство да његова решивост у П повлачи П=НП.

4.3 НП–комплетност

Дефиниција 4.10. Проблем $X \in$ НП је НП–комплетан ако његово решавање у полиномијалном времену повлачи да је П=НП.

Дефиниција 4.11. Нека су L_1 и L_2 језици две ДТМ. Ако постоји функција $f : \Sigma^* \to \Sigma^*$ која је израчунљива у полиномијалном времену неком ДТМ, и ако важи $(\forall w \in \Sigma^*)(w \in L_1 \iff f(w) \in L_2)$ кажемо да је језик L_1 полиномијално сводљив на језик L_2 и пишемо $L_1 \leq_p L_2$.

Лема 4.1. Ако су L_1 и L_2 језици две ДТМ и ако важи $L_1 \leq_p L_2$, онда ако постоји полиномијална ДТМ за решавање L_2, постоји и за L_1.

Доказ. Конструишемо ДТМ M_1 која препознаје L_1 користећи ДТМ M_2 која препознаје L_2. Нека је n дужина улаза w машине M_1.

$M_1 =$ „За улаз w:

1. Израчуна $f(w)$.

2. Симулира машину M_2 на улазу $f(w)$."

Нека је функција $f(w)$ израчунљива у времену $\mathcal{O}(n^a)$ за неко a и нека је M_2 израчунљива у времену $\mathcal{O}(n^b)$ за неко b. Тада је $f(w)$ дужине $\mathcal{O}(n^a)$ и M_1 симулира $\mathcal{O}(n^{ab})$ корака M_2. Ова симулација се може урадити у полиномијалном времену, па је и временска сложеност M_1 полиномијална. □

Последица 4.3. Ако за $Y \in$ НП важи $(\forall X \in \text{НП})(X \leq_p Y)$, онда је проблем Y НП–комплетан.

Теорема 4.4. Проблем 4 је НП–комплетан.

Доказ. Нека је X произвољан проблем у НП који НТМ M_X решава у полиномијалном времену. Направимо функцију $f : \Sigma^* \to \{\langle S, k\rangle | S$ је скуп домина, $k \in \mathbb{N}\}$ која слика улазе X у улазе проблема 4 тако да важе услови из дефиниције 4.11.

За неки улаз w, скуп S из $f(w)$ ће бити слично дефинисан као у доказу теореме 3.5, једина је разлика та да је $\delta(q, a)$ за M_X сада скуп, па ћемо додати по домину за сваку вредност из скупа. Приметимо да је величина скупа S константна јер за сваку M_X правимо другачију функцију, па она не зависи од улаза w.

Број k из $f(w)$ за неки улаз w ће бити највећи број корака који машина M_X изврши на неком улазу дужине $|w|$. Ова вредност је лако израчунљива анализом сложености машине M_X и полиномијалне је величине јер се X налази у НП.

Сада, свако упаривање проблема 4 са улазом $f(w)$ одговара историји неке прихватајуће нити M_X. Такође, свака историја прихватајуће нити одговара неком упаривању.

Сви услови су испуњени да применимо лему 4.1 и приметимо да је, по последици 4.3, проблем 4 НП–комплетан. □

4.4 Кук–Левинова теорема

Проблем 5 (**CAT**)**.** Дат је логички израз који се састоји из променљивих $a_1, \ldots, a_n$, оператора негације ($\neg$), коњункције ($\wedge$), дисјункције ($\vee$) и заграда.

Да ли постоји додела вредности променљивама израза ($f : \{a_1, \ldots, a_n\} \to \{\top, \bot\}$) тако да је израз тачан.

Улаз ћемо енкодирати простим записивањем формуле, док ће променљиве бити нумерисане редом у основи 2. Ако је n број променљивих у изразу (не нужно различитих) онда ће дужина улаза бити $\mathcal{O}(n \log n)$. Како је овим записивањем дужина улаза полиномијална по n, даље ћемо за дужину улаза узимати само број n.

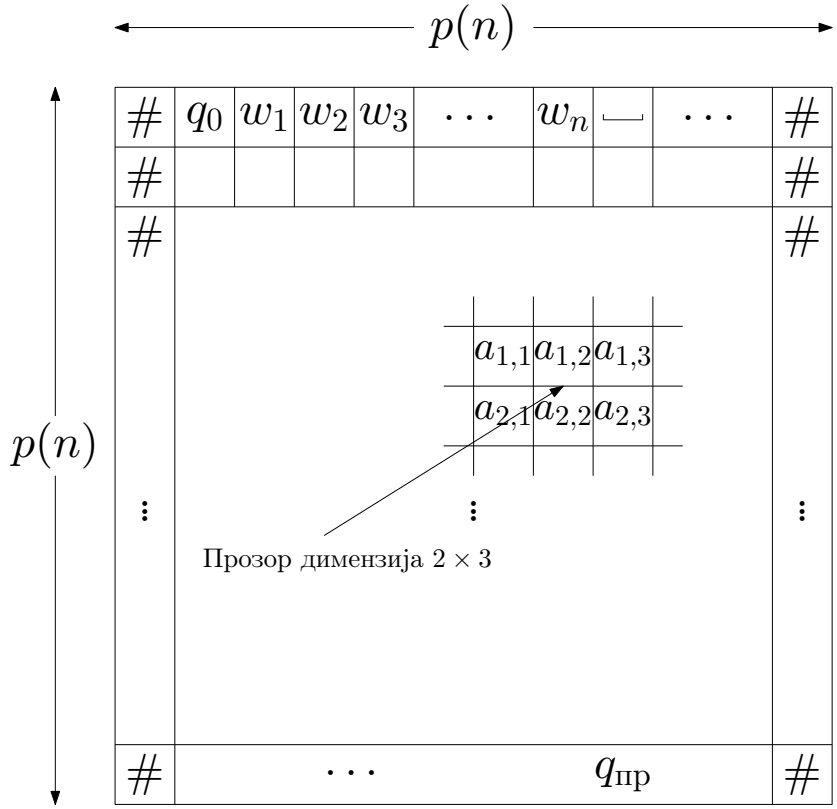
Теорема 4.5 (**Кук–Ленивона теорема**)**.** Проблем 5 је НП–комплетан.

Доказ. За почетак треба показати да је проблем 5 у НП. Како у полиномијалном времену може да се провери да ли једна додела вредности променљивима (која је такође полиномијалне дужине у односу на величину израза) задовољава логички израз, тако се проблем 5 налази у НП.

Нека је X произвољан проблем у НП који НТМ M_X решава у полиномијалном времену.

Направимо функцију $f(w) : \Sigma^* \to \{\phi | \phi \text{ је логички израз}\}$ која слика улазе X у улазе проблема 5 тако да важе услови дефиниције 4.11.

Како је X у НП, постоји полином $p(n)$ такав да M_X на улазу дужине n изврши највише $p(n)$ корака. Запишимо прихватајућу историју рачунања једне нити M_X у матрицу димензија $p(n)^2$, тако да се сваки опис машине налази у засебном реду. Сваки опис је највише дугачак $p(n)$ јер у сваком кораку машина може искористити само једно додатно поље на траци.



Слика 4.1. Илустрација матрице историје рачунања једне нити

За улаз w проблема X, градимо $f(w) = \phi$ тако да постоји додела вредности променљивима ϕ тако да је израз тачан акко постоји прихватајућа историја рачунања M_X.

За свако i и j између 1 и $p(n)$ и свако $s \in S = Q \cup \Gamma \cup \{\#\}$ ћемо направити по променљиву $c_{i,j,s}$ која ће бити $\top$ акко се на позицији (i, j) у матрици налази карактер s.

Формула ϕ ће бити коњункција следећих делова:

1. Треба у сваком пољу да се налази само један карактер.

$$\phi_{\text{поље}} = \bigwedge_{1 \leq i,j \leq p(n)} \left(\left(\bigvee_{s \in S} c_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S \wedge s \neq t} (\neg c_{i,j,s} \vee \neg c_{i,j,t}) \right) \right)$$

Свако поље има барем један карактер

Ни једно поље нема више од једног карактера

2. Треба почетни ред да представља почетну конфигурацију машине.

$$\begin{aligned}\phi_{\text{почетак}} =& c_{1,1,\#} \wedge c_{1,2,q_0} \wedge c_{1,3,w_1} \wedge c_{1,4,w_2} \wedge \cdots \wedge c_{1,n+2,w_n} \\ & \wedge c_{1,n+3,\_} \wedge \cdots \wedge c_{1,p(n)-1,\_} \wedge c_{1,p(n),\#}\end{aligned}$$

3. Крајњи ред мора бити у стању прихватања.

$$\phi_{\text{пр}} = \bigvee_{1 \le i \le p(n)} c_{p(n),i,q_{\text{пр}}}$$

4. Једино је преостало да некако енкодирамо информацију да је прелаз из сваког реда у следећи валидан. Посматраћемо прозоре димензије 2×3, како се садржај траке мења само у непосредној околини главе, потребно је и довољно да су сви прозори ове димензије валидни. Како прозора ове димензије има $|S|^6$ тако је и број валидних прозора константан. Означимо сваки прозор шесторком $(a_{1,1}, a_{1,2}, a_{1,3}, a_{2,1}, a_{2,2}, a_{2,3})$ као на слици 4.1. Дозволићемо и да се прошли ред прекопира, како бисмо успели да попунимо матрицу и када машина прихвати улаз пре корака $p(n)$.

$$\phi_{\text{корак}} = \bigwedge_{1 \le i \le p(n)-1 \wedge 1 \le j \le p(n)-2} \left(\bigvee_{(a_{1,1},a_{1,2},a_{1,3},a_{2,1},a_{2,2},a_{2,3}) \text{ је валидан прозор}} \left(\bigwedge_{1 \le k \le 2 \wedge 1 \le l \le 3} c_{i+k-1,j+l-1,a_{k,l}} \right) \right)$$

(i, j) је почетак сваког прозора — Прозор у (i, j) мора да одговара неком валидном прозору

Како су границе свих $\bigwedge$ и $\bigvee$ полиномијалне величине, тако је могуће изградити ϕ у полиномијалном времену. Функција f задовољава све услове из леме 4.1, па је, због последице 4.3, проблем 5 НП–комплетан. □

Литература

[1] Michael Sipser, *Introduction to the Theory of Computation. 3rd ed.* Cengage Learning, 2012.

[2] Michael R. Garey and David S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* Freeman, New York, 1985.