Turingov stroj in Postov problem

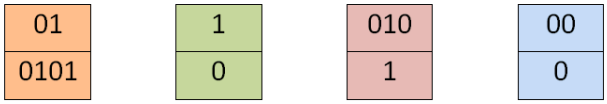
Anja Petković, *FMF, Ljubljana*
Vesna Iršič, *FMF, Ljubljana*
Rok Gregorič, *FMF, Ljubljana*
David Gajser (mentor), *FMF, Duplek*

Povzetek

So problemi v matematiki, tako kot v življenju, ki v okviru dane teorije niso rešljivi. So pa problemi, ki so lahko rešljivi ali ne, vendar nam noben algoritem tega ne more povedati. Kaj pomeni, da z algoritmom rešimo problem in kakšen bi bil primer problema, kjer to ni mogoče?

Uvod

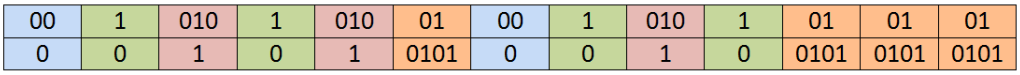
Lotimo se naslednjega problema: na voljo imamo domine



,

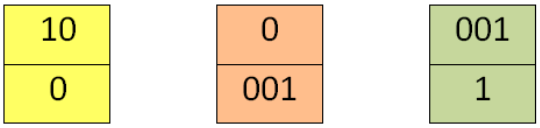
ki jih želimo zložiti v vrsto eno za drugo tako, da bomo zgoraj in spodaj dobili enak niz znakov. Pri tem lahko vsako domino uporabimo poljubno mnogokrat, vendar smemo uporabiti le končno mnogo domin.

Ta problem ni preveč težek in ga rešimo tako, da dane domine zložimo v vrsto



.

Enak problem pri danih dominah



je mnogo težje rešiti na roke, saj njegovo (najkrajšo) rešitev sestavlja 75 domin, pri čemer obstajata dve različni rešitvi te dolžine.

Problem lahko zastavimo tudi v splošnem za n domin s poljubnimi znaki, vendar se moramo prej spoznati s formalizacijo pojmov *abecede* in *besed*.

Definicija 1. *Abeceda* je neprazna končna množica. Ponavadi jo označujemo s Σ. *Besede dolžine* n *nad* Σ so zaporedja $a_1a_2\cdots a_n$, za $a_1, \ldots, a_n \in \Sigma$. Množico vseh besed dolžine n nad Σ označimo s Σ^n in definiramo množico *vseh besed nad abecedo* Σ kot

$$\Sigma^* := \{\varepsilon\} \cup \Sigma^1 \cup \Sigma^2 \cup \cdots,$$

kjer je ε poseben element, ki mu rečemo *prazna beseda*.

Pogosto rabljen primer abecede je $\{0, 1\}$. Besede nad njo so vsa zaporedja ničel in enic, na primer $0, 1, 00, 01, 10, 11, 000, 001, \ldots, \in \{0, 1\}^*$. Besede lahko na naraven način kombiniramo v nove besede: iz besed $w, w' \in \Sigma^*$ dobimo besedo ww' *s stikanjem*. Na primer, če staknemo besedi

1001 in 10, dobimo besedo 100110. Stik prazne besede ε s poljubno besedo w definiramo kot w, torej velja $\varepsilon w = w\varepsilon = w$.

Prej omenjen problem domin je zdaj mogoče natančno matematično formulirati kot:

Postov problem. *Naj bo Σ abeceda in naj bosta $w_1, \ldots, w_n$ in $w'_1, \ldots, w'_n$ zaporedji besed nad Σ. Ali je mogoče najti takšno naravno število k, da bo veljalo*

$$w_{i_1} w_{i_2} \cdots w_{i_k} = w'_{i_1} w'_{i_2} \cdots w'_{i_k}$$

za nek nabor indeksov $i_1, \ldots, i_k$?

Prejšnjo obliko problema z dominami prevedemo v to obliko tako, da j-ti podani domini v prejšnji formulaciji problema zdaj sovpada par besed (w_{i_j}, w'_{i_j}). Ta problem, ki ga je leta 1946 prvi zastavil ameriški matematik Emil Post, bo gonilo preostanka članka. Dokazati želimo naslednji rezultat:

Glavni izrek. *Postov problem je neodločljiv.*

Intuitivni pomen glavnega izreka je, da ni mogoče sestaviti algoritma, ki bi preveril, ali je Postov problem rešljiv ali ne. Da pa bomo lahko formulacijo glavnega izreka sploh razumeli, bomo spoznali nekaj pojmov teorije izračunljivosti ter Turingov stroj.

Turingov stroj

Leta 1936 je angleški matematik Alan Turing zasnoval *Turingov stroj*, ki je preprost model računalnika. Sestavljata ga v eno smer neskončen trak in glava, ki se premika po traku levo ali desno. Glava prebere znak zapisan pod njo na traku, nato ga lahko pobriše, na njegovo mesto napiše drug znak, pri tem pa prehaja preko različnih stanj, dokler ne pride do sprejemnega ali zavrnitvenega stanja - takrat se stroj ustavi. Lahko se tudi zgodi, da stroj nikoli ne pride v eno od teh dveh stanj in se nikdar ne ustavi.

Povejmo še formalno definicijo Turingovega stroja.

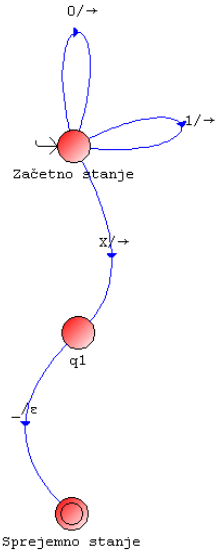
Definicija 2. Turingov stroj je osmerica $(Q, \Sigma, \Gamma, \textvisiblespace, \delta, q_0, q_{accept}, q_{reject})$, kjer so Q, Σ, Γ neprazne končne množice in

1. Q je *množica stanj*,
2. Σ je *vhodna abeceda* (ki ne vsebuje praznega znaka $\textvisiblespace$),
3. Γ je *tračna abeceda*, kjer je $\textvisiblespace \in \Gamma$ in $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, D\}$ je *prehodna funkcija*,
5. $q_0 \in Q$ je *začetno stanje*,
6. $q_{accept} \in Q$ je *sprejemno stanje* in
7. $q_{reject} \in Q$ je *zavrnitveno stanje*, kjer $q_{accept} \neq q_{reject}$.

Razložimo, kako deluje prehodna funkcija δ, ki je ključen del Turingovega stroja. Denimo, da je stroj v stanju q in je njegova glava na mestu na traku, kjer je znak a. Če je $\delta(q, a) = (r, b, L)$, bo stroj izbrisal znak a in na njegovo mesto zapisal b, pri tem bo prešel v stanje r ter se premaknil v levo (L). Če bi bila zadnja komponenta D, bi se stroj premaknil v desno. Vhodne podatke oziroma vhod Turingovemu stroju podamo kot nbesedo iz Σ^*, ki se nahaja na začetku traku, preostanek traku pa je prazen, t.j. zapolnjen s praznimi simboli. Glava se na začetku nahaja na najbolj levem delu traku in je v začetnem stanju. Od tu naprej stroj deluje, kot mu predpisuje prehodna funkcija.

Za boljšo predstavo bomo opisali zelo preprost Turingov stroj M, ki preveri, če je vhod sestavljen le iz 0 in 1, ter se zaključi z znakom X. Prehodna funkcija deluje na sledeč način:

- če glava prebere 0 ali 1, se premakne v desno in stroj ostane v istem stanju,
- če prebere X, se premakne v desno in stroj preide v stanje q_1,
- če prebere kateri koli drug znak, stroj zavrne vhod,
- če v stanju q_1 glava prebere prazen znak, stroj sprejme vhod.



Slika 1: Grafični prikaz Turingovega stroja M

Kot zanimivost omenimo, da Turingov stroj z lahko rešuje enake probleme kot stroj z v obe strani neskončim trakom, stroj z večimi vzporednimi trakovi ter sodobni in kvantni idealizirani računalnik. Prav tako lahko brez škode za splošnost namesto poljubne abecede vzamemo za vhodno abecedo $\Sigma = \{0, 1\}$ in za tračno abecedo $\Gamma = \{0, 1, \_\}$.

Church - Turingova teza

Neformalno algoritem razumemo kot končno zaporedje preprostih ukazov, ki izvedejo nek postopek. S pomočjo Turingovega stroja pa lahko algoritem tudi korektno pojmujemo.

Church-Turingova teza. *Problem lahko rešimo z algoritmom natanko tedaj, ko obstaja Turingov stroj, ki reši ta problem.*

Church-Turingova teza je v rabi od leta 1936 in je v teoretičnem računalništvu splošno sprejeta. Njej zahvaljujoč se ni potrebno spuščati v podrobnosti, ki jih zahteva delo s Turingovim strojem, ampak lahko delamo na višjem nivoju abstrakcije. To omogoča lažje razumevanje konceptov, saj je po Church-Turingovi tezi reševanje problemov z algoritmi enakovredno reševanju s Turingovim strojem. Tako Turingovega stroja ni potrebno uporabljati neposredno, ampak predstavlja matematični koncept, ki daje osnovo za delo z algoritmi.

Odločljivi in neodločljivi problemi

Problemom, na katere lahko odgovorimo samo z *da* ali *ne*, pravimo *odločitveni problemi*. Ker nas zanima predvsem Postov problem, ki je odločitven, se bomo od tu naprej ukvarjali le s takimi problemi.

Definicija 3. Naj bo Σ abeceda. *Jezik* je podmnožica Σ^*.

Odločitvne probleme želimo reševati s Turingovimi stroji, zato jih zakodiramo v abecedi Σ. Dovolj je predpisati, za katere besede je odgovor na odločitveni problem *da*, torej je za odločitveni problem dovolj podati množico besed nad Σ, to je jezik. Vidimo, da odločitveni problemi sovpadajo z jeziki, zato lahko vse pojme, povezane z odločitvenimi problemi, formuliramo z jeziki.

Do sedaj smo govorili o reševanju problemov, vendar pa glavni izrek govori o tem, ali je reševanje z algoritmom sploh mogoče. Zato potrebujemo naslednjo definicijo.

Definicija 4. Jezik $\mathcal{J} \subseteq \Sigma^*$ je *odločljiv*, če obstaja Turingov stroj, ki se vedno ustavi in sprejme natanko besede iz $\mathcal{J}$.

Ekvivalentno, odločitven problem je *odločljiv*, če sovpada z odločljivim jezikom. Primer odločitvenega problema, ki smo ga navedli pri opisu Turingovega stroja, smo rešili z algoritmom, zato vemo, da je odločljiv.

Glede na to, da smo definirali, kateri problemi so odločljivi, bi pričakovali, da bodo obstajali tudi problemi, ki niso. Takšnim pravimo *neodločljivi problemi*. Eden najpomembnejših in najbolj znanih neodločljivih problemov je

Zaustavitveni problem. *Ali se Turingov stroj M nad abecedo Σ ustavi na besedi $w \in \Sigma^*$?*

Dokaza, da je zaustavitveni problem neodločljiv, za voljo dolžine in poljudnosti članka ne bomo navedli. Za interesiranega bralca bomo omenili le, da se uporabi variacija Cantorjevega diagonalizacijskega argumenta.

Neodločljivost zaustavitvenega problema se pogosto uporablja za dokaze, da nekateri drugi problemi niso odločljivi. Tako je tudi s Postovim problemom.

Skica dokaza glavnega izreka. Recimo, da je Postov problem odločljiv. Pokazali bomo, da iz tega sledi, da je tudi zaustavitveni problem odločljiv. To bomo dosegli v dveh delih.

1. Najprej pokažemo, da je Postov problem odločljiv natanko tedaj, ko je odločljiv *modificiran Postov problem*, to je problem, ki je enak Postovemu, le da predpišemo s katero domino se mora zaporedje domin v rešitvi začeti.[1]

2. Nato pokažemo, da iz odločljivosti modificiranega Postovega problema sledi odločljivost zaustavitvenega problema. To naredimo tako, da na podlagi danega Turingovega stroja M in besede w konstruiramo takšne domine, da bo modificiran Postov problem rešljiv natanko tedaj, ko se M ustavi na w.

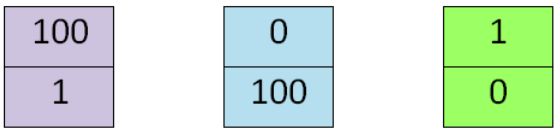
Izpeljali smo, da je zaustavitveni problem odločljiv, kar nas privede v protislovje. S tem je glavni izrek dokazan. □

Zaključek

Videli smo, da Postov problem ni odločljiv in tako ni smiselno iskati algoritma, ki bi ga preverjal. To pa ne pomeni, da se s Postovim problemom ni več vredno ukvarjati.

Lahko se vprašamo na primer, ali je Postov problem pri konkretnih naborih domin rešljiv ali ne. Izmed problemov s tremi dominami, kjer je največja dolžina besede tri, so razrešeni že skoraj vsi problemi. To pomeni, da zanje obstaja rešitev, ali pa je dokazano, da se jih ne da rešiti. Zadnji odprt problem je podan z dominami

[1] Formalno trditev modificiranega Postovega problema dobimo tako, da v formalni trditvi Postovega problema dodamo še zahtevo , da je $w_{i_1} = w_1$ in $w'_{i_1} = w'_1$.



Ko boste imeli trenutek prostega časa, se lahko z njim pozabavate tudi vi.

Viri

[1] M. Sipser, *Introduction to the Theory of Computation, Second Edition.* Course Tehnology, 2006.

[2] L. Zhao, *PCP: a Nice Problem*, `http://webdocs.cs.ualberta.ca/~games/PCP/`, citirano dne 21. 8. 2013.