Četrte vaje APS2: požrešni algoritmi

(1) S polnim tankom goriva, ki zadošča za g kilometrov vožnje, se odpravimo na pot. Ob poti so na razdaljah d_1, d_2, d_3, ... razporejene črpalke (razdalja do prve črpalke je d_1, od prve do druge je d_2, od druge do tretje d_3 itd.), pri čemer za vsak i velja $d_i \leqslant g$. Na katerih črpalkah naj se ustavimo, da ne bomo nikoli ostali brez goriva in da bomo imeli karseda malo postankov? □

Precej očitno je, da se moramo vsakokrat ustaviti na zadnji možni črpalki.[1] Če smo tank ravnokar napolnili na črpalki i (ali pa če smo na začetku poti — v tem primeru lahko vzamemo $i = 0$), se moramo naslednjič ustaviti na črpalki z zaporedno številko, ki je enaka največjemu $j > i$, za katerega velja $d_{i+1} + d_{i+2} + \ldots + d_j \leqslant g$.

Dokažimo, da je tovrstni požrešni algoritem (imenujmo ga A) res optimalen. Naj bo $\mathcal{A} = \langle a_1, a_2, \ldots, a_n \rangle$ naraščajoče urejeno zaporedje zaporednih številk prvih n črpalk, na katerih se ustavimo v skladu z algoritmom A, $\mathcal{O} = \langle o_1, o_2, \ldots, o_n \rangle$ pa naj bo naraščajoče urejeno zaporedje zaporednih številk prvih n črpalk, na katerih se ustavimo v skladu z neko poljubno optimalno metodo O. Trdimo, da za vsak $k \in \{1, \ldots, n\}$ velja $a_k \geqslant o_k$.

Trditev dokažimo z indukcijo. Za $k = 1$ trditev gotovo velja, saj algoritem izbere zadnjo možno črpalko (če bi izbral črpalko $a_1 + 1$, bi na poti med črpalkama a_1 in $a_1 + 1$ ostali brez goriva). Predpostavimo, da trditev velja za $k - 1$, in pokažimo, da velja tudi za k. Algoritem A je torej izbral črpalko $a_{k-1} \geqslant o_{k-1}$, sedaj pa izbira črpalko a_k. Se lahko zgodi, da ga algoritem O »prehiti« in izbere črpalko $o_k > a_k$? Seveda ne, saj naš ljubi A vedno izbere *zadnjo možno* črpalko, zato bo ob predpostavki $a_{k-1} \geqslant o_{k-1}$ veljalo tudi $a_k \geqslant o_k$.

To je to! Če na trditev $a_n \geqslant o_n$ pogledamo z drugega zornega kota, vidimo, da nas bo algoritem A pri fiksni dolžini poti napotil na kvečjemu toliko črpalk, kot bi nas pri isti dolžini poti katerikoli optimalni algoritem. Algoritem A je zategadelj optimalen.

(2) Celo FRI-jevci se kdaj pa kdaj »friziramo«, dasiravno avtor tega besedila pošteno prizna, da to počne odločno preredko. No, pri našem frizerju ali, če hočete, FRIzerju se je na začetku delovnega dne nabralo n strank. Frizer vsako posebej vpraša, česa si želi, in odgovor pretvori v količino časa, ki mu ga bo stranka vzela (recimo, da bo za i-to stranko potreboval t_i enot časa). Kako naj stranke razporedi, da bodo v povprečju najhitreje prišle do svoje želene pričeske? Čakalni čas moramo seveda všteti v skupni čas obravnave. □

Tudi tokrat je odgovor precej očiten: stranke je treba razporediti po naraščajočih časih t_i. Mi ne verjamete? Poskusimo za $t_1 = 5$, $t_2 = 8$ in $t_3 = 3$:

Razpored	Skupna vsota časov obravnave (čakanje + »friziranje«)
1, 2, 3	$5 + (5 + 8) + (5 + 8 + 3) = 34$
1, 3, 2	$5 + (5 + 3) + (5 + 3 + 8) = 29$
2, 1, 3	$8 + (8 + 5) + (8 + 5 + 3) = 37$
2, 3, 1	$8 + (8 + 3) + (8 + 3 + 5) = 35$
3, 1, 2	$3 + (3 + 5) + (3 + 5 + 8) = 27$
3, 2, 1	$3 + (3 + 8) + (3 + 8 + 5) = 30$

[1] Tega algoritma nikar ne preizkušajte v praksi! Cena težav, povezanih z morebitnim izpadom »zadnje možne« črpalke, je namreč neprimerno višja od cene »odvečnega« postanka.

Povprečni skupni čas obravnave je vsakokrat ena tretjina izračunane vsote, a če minimiziramo vsoto, bomo tudi povprečje, saj je število strank konstanta.

Vse lepo in prav, a gotovo se bodo tudi največji lahkoverneži strinjali, da gornji »dokaz« ni vreden piškavega oreha. Koliko časa torej traja vsota skupnih časov obravnave strank 1, 2, ..., n v tem vrstnem redu? Odgovor se glasi

$$T = t_1 + (t_1 + t_2) + (t_1 + t_2 + t_3) + \ldots + (t_1 + t_2 + \ldots + t_n).$$

Trdimo, da je vsota T najmanjša natanko takrat, ko velja $t_1 \leqslant t_2 \leqslant \ldots \leqslant t_n$. Ne verjamete? OK, recimo, da to ni res in da je optimalen nek drug vrstni red strank. V takšnem vrstnem redu gotovo obstaja vsaj ena *inverzija* (par $i < j$, tako da je $t_i > t_j$); še več, gotovo obstaja vsaj ena inverzija na zaporednih indeksih (če je $t_j < t_i$, je razlika $t_k - t_{k-1}$ negativna vsaj za en k med i in j). Sedaj bomo — lahko na podoben način, kot to počne sloviti *bubblesort* — izhodiščno neurejeno zaporedje t-jev preoblikovali v naraščajoče urejeno zaporedje t-jev in bomo videli, da vsaka taka operacija kvečjemu *zmanjša* vsoto skupnih časov obravnave posameznih strank.

No, pa recimo, da za nek k velja $t_{k-1} > t_k$. Če v vsoti T števili t_k in t_{k-1} med seboj zamenjata mesti, dobimo vsoto

$$\begin{aligned} T' = \ldots + (t_1 + \ldots + t_{k-2}) + (t_1 + \ldots + t_{k-2} + t_k) + \\ (t_1 + \ldots + t_{k-2} + t_k + t_{k-1}) + (t_1 + \ldots + t_{k-2} + t_k + t_{k-1} + t_{k+1}) + \ldots \end{aligned}$$

Ta vsota je takšna kot T, le da imamo v njej komponento $(t_1 + \ldots + t_{k-2} + t_k)$ namesto komponente $(t_1 + \ldots + t_{k-2} + t_{k-1})$. Od tod sledi

$$T' = T + t_k - t_{k-1} < T,$$

ker je $t_k < t_{k-1}$. Očitno je bila naša predpostavka o optimalnosti nekega drugega zaporedja (ki ni naraščajoče urejeno) napačna. Naraščajoče urejeno zaporedje časov »friziranja« torej vodi do najmanjše vsote in s tem povprečja skupnih časov obravnave strank.

(3) Razmere so takšne kot pri prejšnji nalogi, le da ima vsaka stranka našega frizerja še določen faktor pomembnosti (p_i), frizer pa bi rad minimiziral vsoto časov obravnave (čakalni čas + čas »friziranja«), uteženo s faktorji p_i. (Če stranka i skupaj s »friziranjem« čaka T_i enot časa, bo njen prispevek k vsoti enak $T_i p_i$.) □

Pravilna rešitev tokrat ni tako očitna, a po krajšem ali daljšem eksperimentiranju se nam bo zazdelo, da je treba stranke naraščajoče urediti po količniku $t_i \,/\, p_i$. To sovpada z intuicijo: če je našemu frizerju stranka zelo pomembna, ji bo dal prednost, tudi če ji bo več ur delal »trajno«.

Analiza je pravzaprav zelo podobna kot pri prejšnji nalogi, le ciljna vsota se tokrat glasi

$$T = p_1 t_1 + p_2(t_1 + t_2) + p_3(t_1 + t_2 + t_3) + \ldots + p_n(t_1 + t_2 + \ldots + t_n).$$

Trdimo, da je ta vsota najmanjša natanko takrat, ko so stranke urejene po naraščajočem količniku $t_i \,/\, p_i$. Po istem razmisleku kot prej pričnemo z zaporedjem, ki vsebuje inverzije (pare strank $i < j$ z lastnostjo $t_i/p_i > t_j/p_j$), ugotovimo, da zanj gotovo obstaja indeks k z lastnostjo $t_{k-1}/p_{k-1} > t_k/p_k$, in ga z medsebojnimi zamenjavami

zaporednih členov pretvorimo v zaporedje brez inverzij. Vsaka taka menjava kvečjemu zmanjša uteženo vsoto:

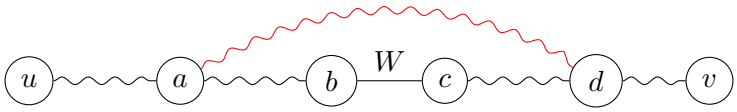
$$\begin{aligned} T' &= \ldots + p_{k-2}(t_1 + \ldots + t_{k-2}) + p_k(t_1 + \ldots + t_{k-2} + t_k) \\ &\quad + p_{k-1}(t_1 + \ldots + t_{k-2} + t_k + t_{k-1}) + p_{k+1}(t_1 + \ldots + t_{k+1}) + \ldots \\ &= T + p_{k-1}t_k - p_k t_{k-1} \\ &= T + p_{k-1}p_k \left(\frac{t_k}{p_k} - \frac{t_{k-1}}{p_{k-1}} \right) \\ &< T, \end{aligned}$$

saj je $t_k/p_k < t_{k-1}/p_{k-1}$.

(4) Podan je utežen povezan neusmerjen graf G. Naj bo $M(p)$ maksimalna utež na poti p, $m_G(u, v)$ pa naj bo minimum vrednosti $M(.)$ prek vseh poti med vozliščema u in v. Naj bo T minimalno vpeto drevo grafa G. Dokažite, da za vsak par vozlišč u in v velja $m_T(u, v) = m_G(u, v)$. □

Trdimo, da izmed vseh poti, ki med vozliščema u in v obstajajo v grafu G, v minimalnem vpetem drevesu ostane samo tista z najmanjšo vrednostjo $M(.)$.

Recimo, da to ni res. Naj bo p pot med vozliščema u in v v minimalnem vpetem drevesu, q pa naj bo pot med u in v z lastnostjo $M(q) < M(p)$. Del poti q, ki je izven drevesa, je potemtakem sestavljen iz povezav z utežmi, ki so strogo manjše od največje uteži W na poti p (na spodnji sliki je pot p sestavljena iz poti $u \rightsquigarrow b$, povezave (b, c) in poti $c \rightsquigarrow v$, pot q pa iz poti $u \rightsquigarrow a$, rdeče poti $a \rightsquigarrow d$ in poti $d \rightsquigarrow v$):



Če iz vpetega drevesa odstranimo povezavo z utežjo W, drevo razpade na dve povezani komponenti, zato moramo vanj dodati eno od povezav med vozliščema a in d na poti q (tisto, ki povezuje omenjeni komponenti). Ker ima dodana povezava manjšo utež od povezave W, smo na ta način dobili drevo z manjšo vsoto uteži povezav, kar pomeni, da izhodiščno vpeto drevo ni bilo minimalno.