Požrešni algoritmi

1 Razvrščanje datotek na trak

1.1 Povprečni čas dostopa

Na i-tem mestu je zapisana datoteka s_i. Čas dostopa (glede na rešitev s) do datoteke zapisane na i-tem mestu je

$$t_i(s) = c \cdot \sum_{j=1}^{i} l(s_j).$$

Povprečni čas dostopa pa lahko iz dvojne vsote poenostavimo na enojno vsoto

$$\begin{aligned} \bar{t}(s) &= \frac{1}{n}\sum_{i=1}^{n} t_i(s) = \frac{c}{n}\sum_{i=1}^{n}\sum_{j=1}^{i} l(s_j) \\ &= \frac{c}{n}\sum_{j=1}^{n}\sum_{i=j}^{n} l(s_j) = \frac{c}{n}\sum_{j=1}^{n}(n-j+1)l(s_j). \end{aligned}$$

Slednje se lepo vidi iz grafične predstavitve, npr. za $n = 4$:

$$\begin{array}{cccc} l_1 & & & \\ l_1 & l_2 & & \\ l_1 & l_2 & l_3 & \\ l_1 & l_2 & l_3 & l_4 \end{array}$$

V prvi dvojni vsoti seštevamo po vrsticah, v drugi pa po stolpcih. Ker so v drugi vsoti v stolpcih enaki elementi, lahko notranjo vsoto lepo poenostavimo. Še enkrat poglejmo enojno vsoto. Opazimo, da je $n-j+1$ enako številu datotek, ki še sledijo (vključno s samo seboj) datoteki na mestu j. Slednja ugotovitev pride prav pri posplošenem problemu z m trakovi.

1.2 Dokaz optimalnosti požrešnega algoritma

Če v rešitvi požrešnega algoritma zamenjamo poljubni dve komponenti, potem se vrednost rešitve kvečjemu poveča (nikoli zmanjša). Posledično je požrešna rešitev najmanjša. Pokažimo to malce bolj formalno.

Brez izgube na splošnosti predpostavimo, da so dolžine zapisov urejene, torej $l_1 \leq l_2 \leq \ldots l_n$. Rešitev, ki jo vrne požrešni algoritem je torej $s = (1, 2, \ldots, n)$ in ustrezno zaporedje dolžin zapisov

$$l_1, l_2, \ldots, l_h, \ldots, l_k, \ldots, l_n,$$

pri čemer sta l_h in l_k dolžini zapisov h in k, kjer $h < k$, ki ju bomo zamenjali. Seveda velja tudi $l_h \leq l_k$. Po menjavi dobimo novo rešitev $s' = (1, 2, \ldots, h-1, k, h+1, \ldots, k-1, h, k+1, \ldots, n$ in ustrezno zaporedje dolžin

$$l_1, l_2, \ldots, l_k, \ldots, l_h, \ldots, l_n.$$

Kolikšna je razlika med vrednostjo s in s'? Torej zanima nas $\bar{t}(s') - \bar{t}(s)$. Pokazali bomo, da

$$\bar{t}(s') - \bar{t}(s) = \frac{c}{n}(k-h)(l_k - l_h) \geq 0,$$

pri pogojih $h < k$ in $l_h \leq l_k$. Posledično je $\bar{t}(s') - \bar{t}(s) \geq 0$ oz. $\bar{t}(s') \geq \bar{t}(s)$, kar pomeni, da je rešitev s' kvečjemu slabša od s. Potemtakem mora biti s optimalna.

1. način Uporabimo prvo dvojno vsoto.

$$\begin{aligned} \frac{n}{c}(\bar{t}(s') - \bar{t}(s)) &= \sum_{i=1}^{n}(t_i(s') - t_i(s)) \\ &= (t_h(s') - t_h(s)) + \sum_{i=h+1}^{k-1}(t_i(s') - t_i(s)) + (t_k(s') - t_k(s)) \\ &= (l_k - l_h) + (k - 1 - (h+1) + 1)(l_k - l_h) + (l_k - l_h + l_h - l_k) \\ &= (k-h)(l_k - l_h). \end{aligned}$$

2. način Uporabimo poenostavljeno enojno vsoto. Upoštevajmo še, da $l(s_k) = l_k = l(s'_h)$ in $l(s_h) = l_h = l(s'_k)$.

$$\begin{aligned}\frac{n}{c}(\bar{t}(s') - \bar{t}(s)) &= (n-k+1)l(s'_k) + (n-h+1)l(s'_h) - (n-h+1)l(s_h) - (n-k+1)l(s_k)\\ &= (n-k+1)l_h + (n-h+1)l_k - (n-h+1)l_h - (n-k+1)l_k\\ &= (h-k)l_h + (k-h)l_k\\ &= (k-h)(l_k - l_h).\end{aligned}$$

1.3 Posplošeni problem

Če imamo m trakov. Požrešno izberemo trak, ki je najmanj zaseden. Čas dostopa je

$$\bar{t}(s) = \frac{1}{n}\sum_{i=1}^{n} r_i l_i,$$

kjer je r_i število datotek, ki sledijo datoteki i na traku na katerem je zapisana.

2 Preprosti nahrbtnik

2.1 Dokaz optimalnosti algoritma

2.1.1 Intuitivno

Brez izgube na splošnosti predpostavimo, da so predmeti urejeni po c_i/v_i. Naj bo x rešitev (sestavljena iz k predmetov), ki jo vrne požrešni algoritem

$$x = (1, \ldots, 1, x_k, 0, \ldots, 0).$$

Naj bo y neka optimalna rešitev

$$y = (1, \ldots, 1, y_j, y_{j+1}, \ldots, y_n).$$

Pri tem je j najmanjši indeks, kjer je $x_j \neq y_j$. Da se pokazati, da $j \leq k$ in $y_j < x_j$.

Iz rešitve y sestavimo novo rešitev z, tako da y_j spremenimo v x_j, ostale y_i, kjer $i > j$ pa ustrezno zmanjšamo.

$$z = (1, \ldots, 1, x_j, z_{j+1}, \ldots, z_n).$$

Ker so predmeti urejeni po c_i/v_i, je vrednost rešitve $z \geq$ od vrednosti rešitve y. Torej je tudi z optimalna rešitev. Postopek ponavljamo, dokler iz y postopoma ne pridemo v x.

2.1.2 Zdaj pa zares

Brez izgube na splošnosti predpostavimo, da so predmeti urejeni po c_i/v_i.

Označimo z $x = (x_i)$ rešitev, ki jo vrne požrešni algoritem. Naj bo k, kjer $1 \leq k \leq n$, število predmetov v x. Spomnimo se, da algoritem prvih $k-1$ predmetov izbere v celoti, k-tega lahko izbere le delno (rezanje), ostalih pa ne izbere. Rešitev algoritma je torej naslednje oblike

$$x = (1, 1, \ldots, 1, x_k, 0, 0, \ldots, 0).$$

Torej za $1 \leq i < k$, velja $x_i = 1$, za k-ti element velja $0 \leq x_k \leq 1$ in za $k < i \leq n$, velja $x_i = 0$.

Naj bo $y = (y_i)$ nek optimalen izbor predmetov. Primerjajmo rešitvi x in y. Če sta enaki, potem ni kaj dokazovati – algoritem vrne optimalno rešitev. Torej predpostavimo, da $x \neq y$. Tedaj naj bo j najmanjši indeks, da $x_j \neq y_j$. Za vse $1 \leq i < j$ seveda velja $x_i = y_i$.

Najprej pokažimo, da velja $y_j < x_j$ in $j \leq k$. Ločimo tri primere.

- Če $j < k$, potem $x_j = 1 > y_j$. Slednje velja, ker $x_j \neq y_j$ in $y_j \leq 1$.
- Če $j = k$, potem
$$\sum_{i=1}^{k-1} y_i v_i + y_k v_k = \sum_{i=1}^{k-1} x_i v_i + y_k v_k + x_k v_k - x_k v_k = V + y_k v_k - x_k v_k \leq V.$$
Iz zadnje neenačbe sledi $y_k \leq v_k$.
- Če $j > k$, potem
$$\sum_{i=1}^{k+1} y_i v_i = \sum_{i=1}^{k} x_i v_i + y_{k+1} v_{k+1} = V + y_{k+1} v_{k+1} > V.$$
Ker $x_{k+1} = 0 \neq y_{k+1}$, zadnja neenačbna ne more nikoli veljati.

Mogoča sta le prva dva primera, torej $j \leq k$. V obeh pa smo pokazali, da $y_j \leq x_j$. To pomeni, da je na mestu razlikovanja med optimalno rešitvijo in rešitvijo, ki jo vrne algoritem delež v optimalni rešitvi manjši od deleža algoritmove rešitve.

Iz optimalne rešitve y skonstruirajmo novo rešitev z, za katero bomo pokazali, da je tudi optimalna. Konstrukcijo naredimo tako, da komponento y_j povečamo, da je enaka x_j, torej $z_j = x_j$. Porabljeno prostornino smo s tem povečali za $(z_j - y_j)v_j$, zato jo moramo pri ostalih komponentah y_i, kjer $i > j$, ustrezno zmanjšati, da se prostornina ne spremeni. Velja torej

$$(z_j - y_j)v_j = \sum_{i=j+1}^{n} (y_i - z_i)v_i.$$

Povečanje komponente y_j lahko naredimo, ker

$$\sum_{i=1}^{j} y_i v_i \leq \sum_{i=1}^{j} x_i v_i \leq \sum_{i=1}^{k} x_i v_i \leq V.$$

Nova optimala rešitev je

$$z = (x_1, \ldots, x_j, z_{j+1}, \ldots, z_n).$$

torej dopustna. Rešitev y se je torej z x ujemala na prvih $j-1$ mestih, rešitev z pa se je ujema na prvih j mestih.

Izračunajmo še vrednost rešitve z

$$\begin{aligned}
\sum_{i=1}^{n} z_i c_i &= \sum_{i=1}^{n} y_i c_i + (z_j - y_j)c_j - \sum_{i=j+1}^{n} (y_i - z_i)c_i = \sum_{i=1}^{n} y_i c_i + \frac{v_j}{v_j}(z_j - y_j)c_j - \frac{c_j}{v_j}\frac{v_j}{c_j}\sum_{i=j+1}^{n} (y_i - z_i)c_i \\
&= \sum_{i=1}^{n} y_i c_i + \frac{c_j}{v_j}\left((z_j - y_j)v_j - \sum_{i=j+1}^{n} (y_i - z_i)c_i\frac{v_j}{c_j}\right) \geq \sum_{i=1}^{n} y_i c_i + \frac{c_j}{v_j}\left((z_j - y_j)v_j - \sum_{i=j+1}^{n} (y_i - z_i)v_i\right) = \sum_{i=1}^{n} y_i c_i
\end{aligned}$$

Neenakost velja, ker za $j+1 \leq i \leq n$, velja

$$\frac{c_i}{v_i} \leq \frac{c_j}{v_j} \quad \text{oz.} \quad c_i\frac{v_j}{c_j} \leq v_i.$$

Vrednost v velikih oklepajih je enaka nič, ker sta oba člena enaka in se odštejeta.

Ugotovili smo, da je vrednost rešitve z enaka vrednosti rešitve y. Torej je z tudi optimalna rešitev. Z rešitvijo x pa se je ujema v eni komponenti več kot y. Postopek tolikokrat ponovimo, dokler se ne dobimo rešitve x, pri čemer so vse vmesne rešitve optimalne in torej je optimalna tudi x.