Korensko urejanje

↓↓↓
Damjan Strnad

→ **O tem, kako pomembna operacija je urejanje zaporedja števil, besed in drugih elementov v praksi, priča izjemno število obstoječih algoritmov urejanja. Ko govorimo o slednjih, najprej pomislimo na mehurčno urejanje, hitro urejanje ali urejanje s kopico. Skupna lastnost teh algoritmov je, da izvajajo neposredno primerjavo elementov zaporedja in izmenjujejo njihove položaje. Obstaja pa tudi drugačen način urejanja, ki namesto na primerjavi temelji na preštevanju elementov in ima teoretično celo ugodnejšo časovno zahtevnost od primerjalnih algoritmov. Enega izmed takšnih preštevalnih algoritmov urejanja bomo opisali v tem članku. Imenuje se *korensko urejanje* (angl. *radix sort*).**

Takoj na začetku naj povemo, da bomo opis omejili samo na eno izmed možnih implementacij korenskega urejanja. Njeno delovanje bomo demonstrirali na urejanju seznama celih števil v naraščajoče zaporedje. Ime korenskega urejanja izhaja iz predpostavke, da so elementi zapisani z znaki v določenem skupnem *korenu* oz. *osnovi*. Desetiška števila so, denimo, zapisana v številskem sistemu z osnovo 10 kot nizi števk med 0 in 9. Vrstni red števk je definiran in zato vemo, da število 5 v urejenem seznamu nastopi prej kot število 8.

Primerjavo večjih števil lahko prevedemo na primerjavo istoležnih števk v njihovem znakovnem zapisu, pri čemer začnemo na levi strani in se pomikamo proti desni. Kot zgled vzemimo števili 1254 in 1281. Primerjava prve in druge števke, ki sta enaki v obeh zapisih, še ne da odločitve o vrstnem redu števil. Šele ob primerjavi tretje števke ugotovimo, da pride število 1254 pred številom 1282. Če želimo na tak način primerjati števila, ki imajo različno dolge znakovne zapise, je potrebno te najprej izenačiti na največjo skupno dolžino. Tako je npr. potrebno seznam števil $\{12, 3, 156, 78, 42\}$ najprej zapisati kot $\{012, 003, 156, 078, 042\}$. Seveda to velja samo za prikaz ročnega reševanja, v računalniku so števila na tak način dejansko že predstavljena.

Predpostavimo, da imamo seznam 100 števil, ki so zapisana kot enako dolgi nizi števk. Vemo, da bodo vsa števila, ki imajo na prvem mestu ničlo (zapišimo jih kot $0*$) v urejenem seznamu pred števili, ki imajo na prvem mestu enico (zapišimo jih kot $1*$). Če torej v (neurejenem) seznamu obstaja osem števil oblike $0*$, bodo po urejanju ta števila zasedala prvih osem mest urejenega seznama, števila oblike $1*$ pa jim bodo sledila od devetega mesta dalje. Prav tako bodo vsa števila oblike $0*$ in $1*$ pred števili oblike $2*$, t.j. števili, ki se pričnejo z dvojko. Če je število oblike $1*$ v začetnem seznamu sedem, bodo po urejanju ta zasedala položaje 9 do 15, števila oblike $2*$ pa mesta od 16-tega naprej. S podobnim sklepanjem lahko zatrdimo, da bodo štiri števila oblike $9*$ po urejanju zasedala zadnja štiri mesta, t.j. položaje od 97 do 100. Vrstni red med števili, ki imajo na prvem mestu isto števko, je določen s števko na drugem mestu. Pri številih z enakima prvima števkama o vrstnem redu odloča tretja in tako naprej.

Za ljudi je opisani postopek bolj naraven, če ga izvajamo z zaporedno primerjavo števk od leve proti desni. V nadaljevanju bomo opisali različico korenskega urejanja, ki števila pregleduje od desne proti levi. V literaturi se ta algoritem imenuje *korensko urejanje z najmanj pomembno števko* (angl. *least significant digit (LSD) radix sort*). Osnovna ideja korenskega urejanja je v preštevanju elementov, ki imajo na določenem mestu v zapisu enak znak oz. števko. Ko je število elementov s posameznimi znaki znano, jih prepišemo na ustrezna mesta v pomožnem seznamu enake velikosti, kot bo opisano v nadaljevanju. Pri tem je ključnega pomena, da ohranjamo obstoječi vrstni red med elementi, ki imajo na opazova-

→

nem mestu isti znak in so bili pred tem že urejeni po števki desno od trenutno opazovane. Če opisani postopek ponavljamo s pregledovanjem števk od najbolj desne do najbolj leve, pri čemer vlogi osnovnega in pomožnega seznama izmenjujemo, bomo kot rezultat dobili urejen seznam števil.

Naj n_i označuje število elementov s števko i na opazovanem mestu, kjer je $i \in \{0, \ldots, 9\}$ v primeru desetiških števil. Potem bodo v novem seznamu števila z opazovano števko j zasedala n_j mest od p_j do $p_j + n_j - 1$, kjer je:

- $$p_j = \begin{cases} 1; & j = 0 \\ 1 + \sum\limits_{k=0}^{j-1} n_k; & j \neq 0. \end{cases} \qquad (1)$$

Opazimo lahko, da za $j > 0$ velja $p_j = p_{j-1} + n_{j-1}$. Celotni algoritem lahko zapišemo v naslednjih korakih:

1. Postavi opazovani položaj na skrajno desno (t.j. položaj enic).
2. Sprehodi se skozi glavni seznam in preštej elemente, ki imajo na opazovanem položaju posamezno števko.
3. S pomočjo enačbe (1) določi nove začetne položaje elementov s posamezno števko.
4. S ponovnim prehodom skozi glavni seznam prepiši elemente na ustrezna mesta v pomožnem seznamu.
5. Zamenjaj vlogi glavnega in pomožnega seznama.
6. Če trenutni opazovani položaj ni skrajno levi, ga premakni za eno mesto v levo in pojdi na korak 2, sicer končaj.

Za zgled delovanja algoritma uporabimo seznam števil $\{87, 541, 303, 221, 34, 1231, 829, 705, 1041, 522, 92, 594\}$. V nadaljevanju bomo opazovani položaj v zapisih števil označevali s podčrtanim tiskom. Za ročno izvedbo moramo najprej števila zapisati z nizi enake dolžine, ki je v našem primeru 4. To nam da naslednji začetni seznam števil:

- $0087, 0541, 0303, 0221, 0034, 1231, 0829, 0705, 1041, 0522, 0092, 0594.$

Urejati pričnemo s preštevanjem posameznih števk na položaju enic. Ugotovimo naslednje:

- $n_0 = 0$
  $n_1 = 4 \quad (054\underline{1}, 022\underline{1}, 123\underline{1}, 104\underline{1})$
  $n_2 = 2 \quad (052\underline{2}, 009\underline{2})$
  $n_3 = 1 \quad (030\underline{3})$
  $n_4 = 2 \quad (003\underline{4}, 059\underline{4})$
  $n_5 = 1 \quad (070\underline{5})$
  $n_6 = 0$
  $n_7 = 1 \quad (008\underline{7})$
  $n_8 = 0$
  $n_9 = 1 \quad (082\underline{9})$

Z uporabo enačbe (1) izračunamo naslednje začetne položaje:

- $p_0 = 1$ $\qquad p_1 = 1 + 0 = 1$
  $p_2 = 1 + 4 = 5$ $\qquad p_3 = 5 + 2 = 7$
  $p_4 = 7 + 1 = 8$ $\qquad p_5 = 8 + 2 = 10$
  $p_6 = 10 + 1 = 11$ $\qquad p_7 = 11 + 0 = 11$
  $p_8 = 11 + 1 = 12$ $\qquad p_9 = 12 + 0 = 12$

Sedaj izvedemo ponoven prehod skozi seznam in števila sproti prepisujemo na ustrezna mesta v pomožnem seznamu. Algoritmično to najlažje naredimo tako, da število z najdeno števko j zapišemo na položaj p_j in vrednost p_j povečamo za 1. Po prvem prepisovanju je vsebina seznama naslednja:

- $0541, 0221, 1231, 1041, 0522, 0092, 0303, 0034, 0594, 0705, 0087, 0829.$

Vidimo lahko, da je relativni vrstni red števil, ki imajo na zadnjem mestu enako števko, ostal nespremenjen. Urejanju, ki ohranja takšno lastnost, pravimo *stabilno urejanje* (angl. *stable sort*).

V naslednjem koraku preštevamo števke na položaju desetic in dobimo:

- $n_0 = 2 \quad (030\underline{3}, 070\underline{5})$
  $n_1 = 0$
  $n_2 = 3 \quad (022\underline{1}, 052\underline{2}, 082\underline{9})$
  $n_3 = 2 \quad (123\underline{1}, 003\underline{4})$
  $n_4 = 2 \quad (054\underline{1}, 104\underline{1})$

$n_5 = 0$
$n_6 = 0$
$n_7 = 0$
$n_8 = 1$ (00$\underline{8}$7)
$n_9 = 2$ (00$\underline{9}$2, 05$\underline{9}$4)

Novi začetni položaji so:

- $p_0 = 1$ $p_1 = 1 + 2 = 3$
  $p_2 = 3 + 0 = 3$ $p_3 = 3 + 3 = 6$
  $p_4 = 6 + 2 = 8$ $p_5 = 8 + 2 = 10$
  $p_6 = 10 + 0 = 10$ $p_7 = 10 + 0 = 10$
  $p_8 = 10 + 0 = 10$ $p_9 = 10 + 1 = 11$

Rezultat prepisovanja je seznam:

- 0303, 0705, 0221, 0522, 0829, 1231, 0034, 0541, 1041, 0087, 0092, 0594.

V tretjem koraku s preštevanjem stotic dobimo:

- $n_0 = 4$ (0$\underline{0}$34, 1$\underline{0}$41, 0$\underline{0}$87, 0$\underline{0}$92)
  $n_1 = 0$
  $n_2 = 2$ (0$\underline{2}$21, 1$\underline{2}$31)
  $n_3 = 1$ (0$\underline{3}$03)
  $n_4 = 0$
  $n_5 = 3$ (0$\underline{5}$22, 0$\underline{5}$41, 0$\underline{5}$94)
  $n_6 = 0$
  $n_7 = 1$ (0$\underline{7}$05)
  $n_8 = 1$ (0$\underline{8}$29)
  $n_9 = 0$

Izračunani začetni položaji so tokrat:

- $p_0 = 1$ $p_1 = 1 + 4 = 5$
  $p_2 = 5 + 0 = 5$ $p_3 = 5 + 2 = 7$
  $p_4 = 7 + 1 = 8$ $p_5 = 8 + 0 = 8$
  $p_6 = 8 + 3 = 11$ $p_7 = 11 + 0 = 11$
  $p_8 = 11 + 1 = 12$ $p_9 = 12 + 1 = 13$

S prepisovanjem pred zadnjim korakom dobimo seznam:

- 0034, 1041, 0087, 0092, 0221, 1231, 0303, 0522, 0541, 0594, 0705, 0829.

V zadnjem koraku s preštevanjem tisočic dobimo:

- $n_0 = 10$ (0$\underline{0}$34, 0$\underline{0}$87, 0$\underline{0}$92, 0$\underline{2}$21, 0$\underline{3}$03, 0$\underline{5}$22, 0$\underline{5}$41, 0$\underline{5}$94, 0$\underline{7}$05, 0$\underline{8}$29)
  $n_1 = 2$ (1$\underline{0}$41, 1$\underline{2}$31)
  $n_2 = 0$
  $n_3 = 0$
  $n_4 = 0$
  $n_5 = 0$
  $n_6 = 0$
  $n_7 = 0$
  $n_8 = 0$
  $n_9 = 0$

Ugotovimo še, da bomo števila s tisočico 0 zapisovali od položaja $p_0 = 1$, števili s tisočico 1 pa od položaja $p_1 = 11$ naprej, kar nam da končni urejen seznam:

- 0034, 0087, 0092, 0221, 0303, 0522, 0541, 0594, 0705, 0829, 1041, 1231.

Predstavljeni postopek je mogoče optimizirati tako, da elemente najprej grupiramo po dolžinah zapisa in ločeno uredimo elemente z eno, dvema, tremi ali štirimi števkami. Urejene podsezname na koncu združimo. Časovna zahtevnost takšnega korenskega urejanja je $\mathcal{O}(kN)$, kjer je k povprečno število števk, N pa število elementov seznama. Teoretično to pomeni, da je korensko urejanje učinkovitejše od najboljših algoritmov urejanja na podlagi primerjav elementov, katerih časovna zahtevnost je $\mathcal{O}(N \log N)$. V praksi se korensko urejanje izkaže pri velikem številu elementov, medtem ko se pri urejanju krajših seznamov števil bolje obnesejo drugi algoritmi. Ena od omejitev korenskega urejanja je tudi ta, da ga ne moremo uporabiti v kombinaciji s poljubnim kriterijem urejanja, ki ga lahko pri klasičnih algoritmih definiramo s primerjalno funkcijo. Je pa mogoče vse korake v tem članku opisanega postopka urejanja učinkovito paralelizirati, kar je v zadnjem času zelo zaželena lastnost algoritmov.

Literatura

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest in C. Stein, *Introduction to Algorithms*, 3. izdaja, The MIT Press, 2009.

×××