Др Драган Урошевић

ПРИМЕНА КУМУЛАТИВНИХ СУМА

1. Мотивације

Претпоставимо да треба да решите следећи проблем. Нека се на пољима нумерисаним бројевима од 1 до n ($5 \leqslant n \leqslant 10^6$) могу налазити неки предмети. На поља која су у почетном тренутку празна, могу се додавати, али и са њих уклањати (ако су непразна) предмети. Ваш програм треба да прочита и изврши низ команди следећег облика:

- D j k, где је j неки број између 1 и n, а k природан број. Овом командом се на поље са редним бројем j додаје k предмета.
- B j k, где је j неки број између 1 и n, а k природан број. Овом командом се са поља са редним бројем j уклања k предмета. Претпоставка је да на том пољу има бар k предмета.
- P j_1 j_2, где је $1 \leqslant j_1 \leqslant j_2 \leqslant n$. Овом командом се од вас захтева да одговорите колико укупно има предмета на пољима $j_1, j_1 + 1, j_1 + 2, \ldots, j_2$.
- K. Овом командом се прекида извршавање вашег програма.

На први поглед ради се о једноставном задатку. Потребно је дефинисати низ са одговарајућим бројем елемената, тј. укупан број елемената низа је једнак броју поља. Елементи низа садрже информације о броју предмета на одговарајућим пољима. Команде за додавање и уклањање предмета се реализују тако што се одговарајући елемент низа (тј. елемент низа који одговара том задатом пољу) увећа или умањи за одговарајућу вредност. Команда за одређивање укупног броја предмета на задатом скупу поља се реализује тако што се саберу вредности елемената низа који одговарају тим пољима.

Према томе, команде за додавање и избацивање предмета су реализоване врло ефикасно (једно сабирање или једно одузимање), али команда за пребрајање и није ефикасна (потребно је извршити сабирање неких елемената низа). Закључујемо да су нам команде за пребрајање критичне за извршавање програма и да ће број тих команди одредити карактеристике програма. За очекивати је да ће тих команди бити доста и да ће захтевати сабирање великог броја елемената.

Један од начина да убрзамо одређивање збира узастопних елемената је коришћење следеће чињенице:

$$a_i + a_{i+1} + \cdots + a_{j-1} + a_j = (a_1 + a_2 + \cdots + a_{j-1} + a_j) - (a_1 + a_2 + \cdots + a_{i-1}).$$

Ако са s_i означимо збир првих i елемената низа a, тј. $s_i = a_1 + a_2 + \cdots + a_i$, онда ће важити

$$a_i + a_{i+1} + \cdots + a_{j-1} + a_j = s_j - s_{i-1}.$$

Значи, збир рачунамо у константном времену једним одузимањем. Збирови s_i се обично називају парцијалне суме елемената низа a.

Међутим, проблем је само привидно решен. Није тешко приметити да имамо више израчунавања при додавању нових премета на поједина места (тј. при увећавању појединих елемената низа). Елемент a_i учествује у парцијалним сумама s_i, s_{i+1}, ... , s_n. Према томе, свака промена елемента a_i доводи до одговарајућих промена у наведеним парцијалним сумама, тако да морају бити ажуриране. Слично се дешава и при уклањању предмета са појединих поља.

2. Кумулативне суме

Ако је n природни број, означимо са $r(n)$ највећи степен броја 2 који је делилац броја n:

$$r(n) = \max\{2^k : 2^k \mid n\}.$$

Лако се закључује да је $r(n - r(n)) > r(n)$. Ако је $r(n) = 2^k$, онда је $n = q \times 2^k$, где је q непаран број. У супротном би било

$$n = q \times 2^k = 2q_1 \times 2^k = q_1 \times 2^{k+1},$$

тј. било би n дељиво са 2^{k+1}. Због тога је $n - r(n) = q \times 2^k - 2^k = (q-1) \times 2^k$, и пошто је $q - 1$ паран то је $n - r(n)$ дељиво бар са 2^{k+1}, те је

$$r(n - r(n)) \geqslant 2^{k+1}.$$

Ако формирамо низ x на следећи начин:

$$x_0 = n, \qquad x_{i+1} = x_i - r(x_i),$$

онда ће важити:

- сваки елемент низа x је ненегативан (јер је $r(x_i) \leqslant x_i$, па је $x_i - r(x_i) \geqslant 0$),
- низ x_i је монотоно опадајући,
- због тога постоји елемент x_k чија је вредност нула.

  Лако се проверава да је

$$x_i = x_0 - r(x_0) - r(x_1) - \cdots - r(x_{i-1}),$$

а ако се стави да је $i = k$, онда

$$x_0 = n = r(x_0) + r(x_1) + \cdots + r(x_{k-1}).$$

Ако бисмо формирали низ t тако да је t_n једнако збиру $r(n)$ узастопних елемената низа a, где је последњи a_n, онда би парцијалну суму s_n могли релативно лако одредити. Ако дефинишемо низ x, као у претходном излагању, онда ће:

- елемент t_{x_0} бити збир $r(x_0)$ узастопних елемената низа a, где је индекс последњег x_0, па је индекс првог $x_0 - r(x_0) + 1 = x_1 + 1$:

$$t_{x_0} = a_{x_1+1} + a_{x_1+2} + \cdots a_{x_0}$$

- елемент t_{x_1} бити збир $r(x_1)$ узастопних елемената низа a, где је индекс последњег x_1, па је индекс првог $x_1 - r(x_1) + 1 = x_2 + 1$:

$$t_{x_1} = a_{x_2+1} + a_{x_2+2} + \cdots a_{x_1}$$

- елемент $t_{x_{k-1}}$ бити збир $r(x_{k-1})$ узастопних елемената низа a, где је индекс последњег x_{k-1}, па је индекс првог $x_{k-1} - r(x_{k-1}) + 1 = x_k + 1 = 1$:

$$t_{x_{k-1}} = a_1 + a_2 + \cdots a_{x_{k-1}}.$$

Лако се проверава да је

$$t_{x_{k-1}} + t_{x_{k-2}} + \cdots + t_{x_1} = a_1 + a_2 + \cdots + a_n = s_n.$$

Под претпоставком да су елементи низа t израчунати, функцију за одређивање парцијалне суме првих n елемената низа a можемо записати на следећи начин (напоменимо да индексирање елемената низова иде од 1):

```
ParcSuma(n, t)
1  k ← 1
2  rs ← 0;
3  while n ≠ 0 do
4    if odd(n div k) then
5      rs ← rs + t[n];
6      n ← n − k;
7    k ← 2 × k;
8  return s
```

Приметимо да број извршавања **while** петље није већи од $\lfloor \log_2 n \rfloor$. Наиме, у сваком пролазу кроз петљу број k се удвостручава, а његова вредност не може постати већа од n.

Овакви збирови носе назив кумулативне суме и очигледно омогућавају једноставније израчунавање како парцијалних сума тако и сума произвољног броја узастопних елемената низа.

Но да бисмо могли израчунати парцијалне суме, морају и елементи низа t бити увек ажурни. Шта се дешава када се промени елемент a_n за број Δ? Очигледно се за Δ мењају неки елементи низа t. Да видимо који су то елементи. За почетак, ако је $2^k \geqslant n$, онда се a_n налази међу првих 2^k елемената низа a па учествује у збиру t_{2^k} и зато тај збир треба ажурирати. Ако је $2^k < n$, онда можемо одредити количник $q = \lceil \frac{n-1}{2^k} \rceil$ и тада је $q \times 2^k + 1 \leqslant n \leqslant (q+1) \times 2^k$. Елемент $t_{(q+1)\times 2^k}$ садржи збир 2^k последњих елемената низа a, а међу њима је и a_n, али само ако је $q+1$ непаран број (јер је тада $r((q+1) \times 2^k) = 2^k$). Овакав поступак израчунавања се може применити и онда када је $2^k \geqslant n$. Наиме, тада је $n - 1 < 2^k$ па је онда $q = \lceil \frac{n-1}{2^k} \rceil = 0$ те је $q + 1 = 1$ тако да ће бити ажуриран елемент t_{2^k}.

Псеудокод за ажурирање елемената низа t (са n елемената), након промене елемента a_m за вредност Δ се може записати на следећи начин:

```
AzurirajKS(n, m, Delta, t)
1  m1 ← m − 1;
2  k ← 1
3  while k ⩽ n do
4    q ← m1 div k
5    if odd(q + 1) then
6      t[(q + 1) × k] ← t[(q + 1) × k] + Delta
7    k ← 2 × k
```

Сложеност ажурирања кумулативних сума је одређена бројем извршавања **while**-петље. Петља се извршава за вредности $k = 1, 2, 4, \ldots, k_{\max} = 2^{l_{\max}}$, где је $k_{\max}$ степен броја два који није већи од n. Одавде је $k_{\max} \leqslant n$ и $2 \times k_{\max} > n$. Будући да је $k_{\max} = 2^{l_{\max}}$, то је $l_{\max} = \lfloor \log_2 n \rfloor$, те је сложеност ажурирања $\Theta(\log_2 n)$.

Према томе сложеност свих операција је $\Theta(\log_2 n)$.

3. Дводимензионални случај

На Међународној олимпијади 2001. године био је за нијансу тежи проблем. Део равни који представља подручје које „покрива“ један оператор мобилне телефоније има облик правоугаоника и издељен је на квадрате (представљају основну јединицу у организацији). Укупно има m врста и n колона квадрата ($m, n \leqslant 1000$) и на сваком од њих има одређен број претплатника. Колоне су нумерисане бројевима од 1 до n слева надесно, а врсте бројевима од 1 до m одоздо нагоре (за нијансу другачије од нумерације врста и колона у матрицама, али то не би требало да ствара проблеме).

Током времена се појављују нови претплатници и одјављују неки од постојећих. Оператора интересује да с времена на време одреди број претплатника у некој подобласти (која има облик правоугаоника и одређена је координатам квадрата који се налазе код наспрамних темена (рецимо (v_1, k_1) и (v_2, k_2), $1 \leqslant v_1 \leqslant v_2 \leqslant m$ и $1 \leqslant k_1 \leqslant k_2 \leqslant n$). Такмичарски програм треба да „испроцесира“ низ команди облика: додај одређен број претплатника на одређено поље, одјави одређен број претплатника, преброј укупан број претплатника на правоугаоној подобласти.

По аналогији са једнодимензионим случајем, збир на пољима у оквиру правоугаоне области чија наспрамна темена су (v_1, k_1) и (v_2, k_2) се може одредити комбиновањем збирова у правоугаоним областима чије је једно теме $(1, 1)$ (то би били аналогони парцијалних сума). Ако је $s[v, k]$ сума садржаја поља у правоугаоној области чије је једно теме $(1, 1)$, а друго (v, k), онда је збир поља правоугаоне облати чија су темена (v_1, k_1) и (v_2, k_2) једнака

$$s[v_2, k_2] - s[v_1 - 1, k_2] - s[v_2, k_1 - 1] + s[v_1 - 1, k_1 - 1].$$

Ове парцијалне суме се могу израчунати применом кумулативних сума, аналогних кумулативним сумама за једнодимензиону област. Тако ће $t[v, k]$ бити сума поља правоугаоне области чије је горње лево теме (v, k), чија је висина $r(v)$ (тј. садржи

$r(v)$ врста), а ширина $r(k)$. Ако су те суме ажурне, онда ћемо $s[v,k]$ одредити следећим поступком

ParcSuma2D(v,k,t)
```
1  s ← 0
2  kv ← 1
3  while v > 0 do
4     if odd(v div kv) then
5       kk ← 1
6       k1 ← k
7       while k1 > 0 do
8         if odd(k1 div kk) then
9           s ← s + t[v, k1]
10            k1 ← k1 − kk
11        kk ← 2 × kk
12    kv ← 2 × kv
13 return s
```

Ажурирање изводимо као и ажурирање у једнодимензионом случају. То значи да ћемо за сваки пар вредности kv и kk експонената одређивати правоугаону област чије горње десно теме има координате (v,k) где је v умножак од 2^{kv}, k умножак од 2^{kk} и ако су притом $v/2^{kv}$ и $k/2^{kk}$ непарни, онда кумулативну суму за то поље треба ажурирати.

AzurirajKS2D$(m,n,v,k,Delta,t)$
```
1  v1 ← v − 1;
2  vk ← 1
3  while vk ⩽ m do
4    vq ← v1 div vk
5    if not odd(vq) then
6      k1 ← k − 1
7      kk ← 1
8      while kk ⩽ n do
9        kq ← k1 div kk
10       if not odd(kq) then
11         t[(vq + 1) × vk][(kq + 1) × kk] ← t[(vq + 1) × vk][(kq + 1) × kk] + Delta
12       kk ← 2 × kk
13   vk ← 2 × vk
```

4. Примена на израчунавање минимума/максимума

Други проблем који се може решити применом описане стратегије је и проблем одређивања минимума (или максимума) за подниз кога чине сви елементи низа од првог (почетног) до неког конкретног. По аналогији са кумулативним сумама у низу кумулативних минимума (или максимума) чувамо минимуме (максимуме) за део низа: `tmin[k]` ће бити минимум $r(k)$ узастопних елемената низа

где је последњи елемент на k-том месту. Применом таквих кумулативних минимума лако рачунамо минимум подниза од првог до произвољног елемента (ако смо код парцијалних сума рачунали то као збир кумулативних сума, у овом случају рачунамо минимум кумулативних минимума).

Међутим, ако бисмо желели да израчунамо минимум од узастопних елемената где почетни не би био први елемент низа, него неки каснији, то не бисмо могли да изведемо као за збир (рачунајући разлику парцијалних сума). Познавање минимума за првих n елемената и минимума за првих $m-1$ елемената ($n \geqslant m$) не омогућава директно израчунавање минималног елемента у поднизу почев од m-тог до n-тог.

О структури података која би обезбедила ефикасно одређивање таквих минимума и максимума у неком будућем тексту.

ЛИТЕРАТУРА

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein, *Introduction to Algorithms*, MIT Press and McGraw-Hill, 2001.
2. Robert Sedgewick, *Algorithms in C++*, Addison-Wesley Professional.

Математички институт САНУ, Београд, Кнеза Михаила 36

E-mail: draganu@mi.sanu.ac.yu