Hello, World!

Међународна информатичка такмичења

Никола Милосављевић

У свету информационих технологија, а нарочито на такмичењима из програмирања, чести су проблеми у којима је потребно извршавати неке упите над неким скупом података. На пример, m упита над скупом величине n. Приступ свим подацима у сваком упиту, тј. извршавање свих упита, у сложености $O(nm)$ углавном није прихватљиво и често се користе компликоване структуре података ради убрзања. Ипак, понекад су довољни само једноставни и лепи трикови/технике за нетривијална убрзања. У овом чланку представљамо једну такву технику која је у круговима такмичарског програмирања позната као „Моов алгоритам" (*Mo's algorithm*). Она омогућава да се одређени типови упита над низом изврше у укупној сложености $O\big(m \log m + (n+m)\sqrt{n}\big)$.

Посматрајмо проблеме следећег облика: Дат је низ a (нпр. целих бројева) дужине n и m упита облика (l_i, r_i), $1 \leq l_i \leq r_i \leq n$, са значењем „израчунати дату функцију $Q(l_i, r_i)$" која зависи само од елемената низа $a_{l_i}, a_{l_i+1}, \ldots, a_{r_i}$. Потребно је одговорити на све упите. Додатно, на упите је дозвољено **одговарати у произвољном редоследу**, а не обавезно у редоследу у којем су задати. Програмерским жаргоном – дозвољено је задатак радити *offline*. Илуструјмо овај апстрактни опис са два конкретна проблема. Први се може урадити у бољој сложености од Моовог алгоритма, али је због своје једноставности згодан за разумевање поменуте технике.

Проблем 1: Дат је низ целих бројева a дужине n. Одговорити на m упита облика (l_i, r_i) са значењем „израчунати суму свих бројева од l_i-те до r_i-те позиције у низу a".

Проблем 2: Дат је низ a дужине n чији су елементи природни бројеви из сегмента $[1, n]$. Одговорити на m упита облика (l_i, r_i) са значењем „за свако $x = 1, 2, \ldots, n$, сабрати вредности $x \cdot c(x)^2$, где је $c(x)$ број појављивања вредности x међу бројевима $a_{l_i}, a_{l_i+1}, \ldots, a_{r_i}$".

Дакле, у првом проблему је $Q_1(l, r) = \sum_{i=l}^{r} a_i$, док је у другом $Q_2(l, r) = \sum_{x=1}^{n} c_{l,r}(x)^2 \cdot x$, где је $c_{l,r}(x)$ број појављивања вредности x у поднизу $a[l, r]$. Функције $Q(l, r)$ су најчешће такве да се једноставно рачунају једним проласком кроз све елементе од l-те до r-те позиције у низу тј. може се применити следећи једноставан али спор алгоритам.

Алгоритам SlowSolution
Улаз: Низ $a[1..n]$ и m упита (l_i, r_i)
Излаз: Одговори на све упите

```
for i ← 1 ... m
        sol ← 0
        for j ← l_i ... r_i
                update sol brojem a_j na osnovu funkcije Q(l, r)
        print sol
```

За проблем 1 „update sol" из алгоритма има значење $sol \leftarrow sol + a_j$. За проблем 2, потребан нам је помоћни низ $c[1..n]$ (из описа проблема) који је на почетку попуњен нулама. У „update sol" делу је потребно повећати број појављивања броја a_j ($c[a_j] \leftarrow c[a_j] + 1$), а затим повећати и тренутну суму: $sol \leftarrow sol + (2 \cdot c[a_j] - 1) \cdot a_j$ (размислити!). На крају сваког упита треба „ресетовати" одговарајуће елементе низа c на нулу поновним проласком кроз подниз (не цео низ) $a[l_i, r_i]$.

Сложеност претходног алгоритма је $\Theta(\sum_{i=1}^{m} |r_i - l_i|)$, што може да буде $\Theta(nm)$. На пример, када су упити „дугачки". Јасно, желимо боље решење.

Приметимо да функције $Q(l, r)$ за проблеме 1 и 2 поседују следећу особину. Уколико нам је позната вредност $Q(l, r)$, тада на основу ње можемо ефикасно (у $O(1)$) израчунати и вредности „суседних" упита $Q(l, r+1), Q(l, r-1), Q(l-1, r), Q(l+1, r)$. Заиста, у проблему 1 је то само додавање/одузимање једног броја док је у проблему 2 довољно урадити два једноставна корака као у претходном решењу. Ово запажање можемо искористити на следећи начин. Уместо да на сваки упит одговарамо **независно**, одговор на наредни упит добићемо **на основу претходног**, тако што ћемо **„померати"** крајеве претходног упита/сегмента за по 1 (и мењати вредност функције) све док се не поклопе са новим. На пример, имамо два узастопна упита $Q(8, 15)$ и $Q(3, 30)$. Тада, уколико знамо вредност упита $Q(8, 15)$ можемо, редом, рачунати вредности упита $Q(8, 16)$, $Q(8, 17)$, ... , $Q(8, 29)$, $Q(8, 30)$, $Q(7, 30)$, $Q(6, 30)$, ... , $Q(3, 30)$. Ова идеја је имплементирана у наредном алгоритму.

Алгоритам MovingSegments
Улаз: Низ $a[1..n]$ и m упита (l_i, r_i)
Излаз: Одговори на све упите

```
sol ← 0, left ← 1, right ← 0,
for i ← 1 ... m
        while (right < r_i)
```

```
        right ← right + 1
        add a_right and update sol
while (right > r_i)
        remove a_right and update sol
        right ← right − 1
while (left > l_i)
        left ← left − 1
        add a_left and update sol
while (left < l_i)
        remove a_left and update sol
        left ← left + 1
print sol
```

Приметимо да смо у алгоритму кренули од вештачког (нултог) упита $(l_0, r_0) = (1, 0)$ да не бисмо посебно третирали први упит. За сваки прелаз са упита на упит (m прелаза) извршиће се највише једна од прве две while петље, као и највише једна од последње две while петље.

Да ли је овај алгоритам заиста бољи од претходног? Уколико је прелаз на „суседне" упите операција сложености $O(1)$, тада прелаз са упита (a, b) на упит (c, d) захтева $\Theta(|a - c| + |b - d|)$ операција. Тако је укупна сложеност алгоритма $\Theta(\sum_{i=1}^{m}(|l_i - l_{i-1}| + |r_i - r_{i-1}|)$. Уколико су сви упити „дугачки", нпр. сви су облика $(1, n)$, тада овај алгоритам ради у сложености $\Theta(n + m)$ за разлику од претходног чија је сложеност $\Theta(nm)$. Нажалост, испоставља се да овај алгоритам може достићи и сложеност $\Theta(nm)$. На пример, за упите $(1,1)$, (n, n), $(1, 1)$, (n, n), ... Иначе, њих претходни алгоритам решава у сложености $\Theta(n + m)$.

Иако изгледа да идеја о померању упита/сегмената није довољно добра, израз за број операција другог алгоритма $\sum_{i=1}^{m}(|l_i - l_{i-1}| + |r_i - r_{i-1}|)$ много више обећава него $\sum_{i=1}^{m}|r_i - l_i|$. Разлог је што први израз, за разлику од другог, **зависи од редоследа упита**. Како је на упите дозвољено одговарати у произвољном редоследу, то постаје интересантан оптимизациони проблем: „Како сортирати (уредити) датих m парова (l_i, r_i) тако да се минимизује израз $\sum_{i=1}^{m}(|l_i - l_{i-1}| + |r_i - r_{i-1}|)$?"

Овај проблем нећемо егзактно решавати, јер је тежак. За наше потребе биће довољно да покажемо да увек постоји поредак упита за који је вредност траженог израза $O\left((n + m)\sqrt{n}\right)$. Уколико бисмо сортирали упите растуће по левим крајевима, тада би важило $\sum_{i=1}^{m}|l_i - l_{i-1}| = O(n)$. Али тада немамо контролу над $\sum_{i=1}^{m}|r_i - r_{i-1}|$. Наћићемо компромис, тако што ћемо поделити низ a на $\sqrt{n}$ узастопних поднизова („блокова"), сваки

дужине $\sqrt{n}$. (Ради једноставности, претпостављамо да је n потпун квадрат.) Затим ћемо дефинисати релацију поретка $\preccurlyeq$ на скупу упита на следећи начин.

$$(l_i, r_i) \preccurlyeq (l_j, r_j) \Leftrightarrow \left(\left\lfloor \frac{l_i}{\lfloor\sqrt{n}\rfloor} \right\rfloor < \left\lfloor \frac{l_j}{\lfloor\sqrt{n}\rfloor} \right\rfloor\right) \vee \left(\left\lfloor \frac{l_i}{\lfloor\sqrt{n}\rfloor} \right\rfloor = \left\lfloor \frac{l_j}{\lfloor\sqrt{n}\rfloor} \right\rfloor \wedge r_i \leq r_j\right).$$

Сортирање упита по релацији $\preccurlyeq$ значи следеће. Прво ћемо одговорити на све упите чији је леви крај у првом блоку. Затим на све упите чији је леви крају у другом блоку итд. На крају ћемо одговорити на све упите чији је леви крај у последњем блоку. Ако има више упита са левим крајем у истом блоку, прво ћемо одговорити на онај са најмањим десним крајем итд. све до оног са највећим десним крајем. Израчунаћемо колико је било померања крајева при овом поретку упита, тако што ћемо посебно проценити изразе $\sum_{i=1}^{m} |l_i - l_{i-1}|$ и $\sum_{i=1}^{m} |r_i - r_{i-1}|$.

Уколико су узастопни леви крајеви l_{i-1} и l_i у истом блоку, тада је $|l_{i-1} - l_i| \leq \sqrt{n}$. Иначе, увек долази до померања удесно и није тешко показати да је укупна сума свих таквих разлика највише $2n$. Дакле, важи $\sum_{i=1}^{m} |l_i - l_{i-1}| \leq m\sqrt{n} + 2n$. С друге стране, за све упите чији су леви крајеви у истом блоку збир свих разлика $|r_i - r_{i-1}|$ је највише n, јер увек долази до померања удесно. Уз то, приликом преласка на наредни блок, десни крај се помера за највише n улево. Како постоји $\sqrt{n}$ блокова, следи $\sum_{i=1}^{m} |r_i - r_{i-1}| \leq 2n\sqrt{n}$.

Према томе, уколико сортирамо упите на основу релације $\preccurlyeq$, важи $\sum_{i=1}^{m} (|l_i - l_{i-1}| + |r_i - r_{i-1}|) \leq m\sqrt{n} + 2n + 2n\sqrt{n} = O\left((n+m)\sqrt{n}\right)$. Померање крајева упита је операција сложености $O(1)$ за проблеме 1 и 2, а типична сложеност соритрања m упита је $O(m \log m)$. Тако су проблеми 1 и 2 решени у сложености $O\left(m \log m + (n+m)\sqrt{n}\right)$, што је значајно побољшање у односу на тривијално $O(nm)$ решење!

Алгоритам Mo's Algorithm
Улаз: Низ $a[1..n]$ и m упита (l_i, r_i)
Излаз: Одговори на све упите
sortirati upite (l_i, r_i) na osnovu relacije $\preccurlyeq$
primeniti algoritam **MovingSegments**

Закључак. Моов алгоритам није ништа друго до „паметно" сортирање упита, а затим померање њихових крајева. Уколико проблеми поменутог облика с почетка текста задовољавају услове:

(1) упити не модификују низ;

(2) није нужно да се одговара на упите у задатом редоследу;

(3) сложеност рачунања упита

$$Q(l, r+1), Q(l, r-1), Q(l-1, r), Q(l+1, r)$$

на основу упита $Q(l, r)$ је $f(n)$ за неку функцију f,

тада се ови проблеми могу решити Моовим алгоритмом у сложености $O\left(m \log m + (n+m)\sqrt{n} \cdot f(n)\right)$. Ово је боље од тривијалног приступа када је $f(n) = o\left(\sqrt{n}\right)$. На пример, ако је $O(1)$ као у поменутим проблемима или $O(\log n)$ уколико се користе додатне структуре.

Задаци

1. Урадити проблем 1 у сложености $O(n+m)$.

2. Дат је низ a дужине n који се састоји од природних бројева из сегмента $[1, k]$ и m упита облика: „Колико има различитих бројева на сегменту $[l_i, r_i]$ у низу a?“. Решити проблем у сложености $O\left(m \log m + (n+m)\sqrt{n}\right)$ користећи $O(k)$ додатне меморије.

3. Дата је пермутација p бројева од 1 до n и m упита облика: „Колико има инверзија од позиције l_i до позиције r_i у пермутацији p?“. Одговорити на упите у сложености $O\left(m \log m + (n+m)\sqrt{n} \log n\right)$.

4. Да ли постоји вредност $k \neq \Theta\left(\sqrt{n}\right)$, таква да, када се низ подели на k узастопних блокова величине $\frac{n}{k}$, добијамо асимптотски бољу сложеност у Моовом алгоритму? (Ради једноставности претпоставити да је $m = n$.)

Статијата прв пат е објавена во списанието Тангента на ДМ на Србија во 2017/18 година