

## Двумерна заявка за минимум

<https://codeforces.com/blog/entry/45485>

Двумерната заявка за минимум представлява разширение на едномерната заявка за минимум и е с  $O(n \times m \times \log(n) \times \log(m))$  времева сложност за построяване на разредена (четири мерна) таблица и  $O(1)$  времева сложност за заявка. 2D версията намира максимум, минимум или друга функция, която колкото и пъти да се приложи – не променя или нарушава изходните данни или отговора.

Преди това, нека си припомним отново едномерната заявка:

### 1D RMQ $\langle O(n \times \log(n)), O(1) \rangle$

Задача: Даден е масив от  $n$  числа и  $Q$  заявки. Всяка заявка пита за минимума в даден интервал от  $x$  до  $y$ , където  $0 \leq x \leq y < n$ .

Инициализиране на разредена таблица:  $O(n \times \log(n))$

```
For (i=0; i<n; ++i)
    st[0][i]=array[i] //building base

For (j=1; j<=log(n); ++j)
    For (i=0; i+(1<<j)<=n; ++i)
        st[j][i]=min(st[j-1][i], st[j-1][i+(1<<(j-1))]) //use previously
computed
```

До тук създадохме разредена таблица с размерност  $[1 + \log(n)][n]$

$st[i][j]$  съдържа минимума в интервала от  $i$  до  $i + 2^j - 1$  включително

Всеки интервал е с дължина степен на двойката. Първо построяваме/изчисляваме базата на таблицата – всеки елемент представлява интервал. След това всеки следващ интервал степен на двойката го разделяме на две равни части, за които стойността вече е изчислена. По този начин, чрез концепцията на динамичното програмиране, създаваме таблицата за  $O(n \times \log(n))$  времева сложност.

Заявка от вида  $Query(L, R) : O(1)$

```
len=R-L+1
k=log2(len)
return min(st[k][x], st[k][y-(1<<k)+1]) // ranges may overlap
```

Заявката представлява просто разделяне на интервала на две парчета, всяко от които е степен на двойката и след това използване на преизчислените стойности за по-малките интервали от таблицата. Двете парчета интервали позволяват да се презастъпват (според вида заявка).

До тук преговорихме накратко 1D RMQ. Сега, нека разширим тази концепция, за да решим 2D RMQ.

### 2D RMQ $\langle O(n \times m \times \log(n) \times \log(m)), O(1) \rangle$

Задача: Дадена е матрица с размерност  $n \times m$  и  $Q$  заявки. Всяка заявка пита за минимума в подматрицата  $[(x_1, y_1), (x_2, y_2)]$ , където  $(x_1, y_1)$  са координатите на горния ляв ъгъл, а

[github.com/andy489](https://github.com/andy489)

$(x_2, y_2)$  на долния десен ъгъл на подматрицата. Допускаме, че индексирването става с основа 0.

За да не ни се налага да се бавим всеки път когато пресмятаме логаритмична функция, може да калкулираме всички необходими логаритми предварително и бързо, като използваме динамично програмиране по следния начин:

```
int v=max(n,m);
log2.assign(v+1, 0)
For(i=2;i<=v;++i)
    log2[i]=log2[i>>1]+1;
```

Инициализиране на таблица:  $O(n \times m \times \log(n) \times \log(m))$

- Създаваме таблица с размерност  $[1 + \log(n)][n][1 + \log(m)][m]$
- Всяка клетка от таблицата  $[1 + \log(n)][n]$  е разредена таблица с размер  $[1 + \log(m)][m]$
- Нека разгледаме какво всъщност съдържа дадена клетка  $[j_r][i_r][j_c][i_c]$ :  
Съдържа минималния елемент от колона  $i_c$  до  $i_c + 2^{j_c} - 1$  на всички редове от  $i_r$  до  $i_r + 2^{j_r} - 1$ .  
С други думи, съдържа минималния елемент в подматрицата  $[(i_r, i_c), (i_r + 2^{j_r} - 1, i_c + 2^{j_c} - 1)]$ , където  $[(x_1, y_1), (x_2, y_2)]$  задава подматрица с координати  $(x_1, y_1)$  на горен ляв връх и координати  $(x_2, y_2)$  на долен десен връх.
- Сега лесно може да заключим, че  $st[0][i_r][j_c][i_c]$  не е нищо повече от 1D RMQ таблица, ако вземем за масив реда  $i_r$ .

Сега нека видим как може да го преобразуваме в код:

Първо, за всеки ред  $i_r$ , трябва да калкулираме таблицата  $st[0][i_r][j_c][i_c]$ , по същия начин както правим и при 1D версията. Взимаме всеки ред като масив и изчисляваме разредената таблица за него по аналогичен начин както при 1D RMQ.

```
For(ir=0;ir<n;++ir){
    For(ic=0;ic<m;++ic)
        st[0][ir][0][ic]=Matrix[ir][ic];

    For(jc=1;jc<=log2[m];++jc)
        For(ic=0;ic+(1<<(jc-1))<m;++ic)
            st[0][ir][jc][ic]=min(st[0][ir][jc-1][ic],st[0][ir][jc-1]
[ic+(1<<(jc-1))])
}
```

Горната стъпка не е нищо повече от пресмятане на разредена таблица а всеки ред. Сложността за един ред е  $O(m \times \log(m))$  и следователно за всички редове е  $O(n \times m \times \log(m))$ .

Ще използваме тази база, за да построим цялата таблица.

```
For(jr=1;jr<=log2[n];++jr)
    For(ir=0;ir<n;++ir)
        For(jc=0;jc<=log2[m];++jc)
            For(ic=0;ic<m;++ic)
                st[jr][ir][jc][ic]=min(st[jr-1][ir][jc][ic],st[jr-1]
[ir+(1<<(jr-1))][jc][ic])
```

Не е толкова трудно да се разбере това, просто минете през всеки ред на горното парче код и се опитайте да си го визуализирате.

Очевидно, горната стъпка е с времева сложност  $O(n \times m \times \log(n) \times \log(m))$ .

Заявка от вида  $Query(x_1, y_1, x_2, y_2): O(1)$

Трябва да намерим минимума който подматрицата  $[(x_1, y_1), (x_2, y_2)]$  съдържа.

```
lenx=x2-x1+1
kx=log2[lenx]
leny=y2-y1+1
ky=log2[leny]
```

Първо ще направим заявка върху  $n$ . Трябва да направим заявката от  $x_1$  до  $x_2$ . Но ние отново сме пресметнали само интервалите, които са степени на двойката и за това ще разделим на два подинтервала, всеки от които е степен на двойката.

$x_1$  до  $x_1 + 2^{k_x} - 1$  (интервал  $R_1$ )  
 $x_2 - 2^{k_x} + 1$  до  $x_2$  (интервал  $R_2$ )

Разделянето може да се припокрива, но това не е проблем, тъй като търсим функция, която колкото и пъти да приложим, не променя изходните данни и резултата.

Сега за всеки от горните интервали  $R_1$  и  $R_2$  имаме заявка в колона от  $y_1$  до  $y_2$ . За тази цел ще направим аналогично разбиване на интервала както направихме по-горе, т.е.

$y_1$  до  $y_1 + 2^{k_y} - 1$  (интервал  $C_1$ )  
 $y_2 - 2^{k_y} + 1$  до  $y_2$  (интервал  $C_2$ )

следователно заявката е представлява просто три реда код,

```
min_R1=min(st[kx][x1][ky][y1],st[kx][x1][ky][y2-(1<<ky)+1])
min_R2=min(st[kx][x2-(1<<kx)+1][ky][y1],st[kx][x2-(1<<kx)+1][ky][y2-(1<<ky)+1])
return min(min_R1,min_R2)
```