

```
int arr[] = {1, 2, -1, -2, 4, 1, -3, 3};
```

arr[i]	1	2	-1	-2	4	1	-3	3
i	0	1	2	3	4	5	6	7

```
int mxN = sizeof arr / sizeof(int);
```

```
int K = 3; /// K =  $\lceil \log_2 8 \rceil$ 
```

```
vector<int> log(mxN + 1);
```

```
vector<vector<int>> st( mxN + 1, vector<int>(K + 1));
```

```
for (int i = 2; i <= mxN; ++i)
```

```
    log[i] = log[i / 2] + 1;
```

log[i]	0	0	1	1	2	2	2	2	3
i	0	1	2	3	4	5	6	7	8

```
for (int i = 0; i < mxN; ++i)
```

```
    st[i][0] = arr[i];
```

Sparse Table

st:	0	1	2	3
0	1	$\min(1, 2)=1$	$\min(1, -2)=-2$	$\min(-2, -3)=-3$
1	2	$\min(2, -1)=-1$	-2	
2	-1	-2	-2	
3	-2	-2	-3	
4	4	1	-3	
5	1	-3		
6	-3	-3		
7	4			
8				

```
for (int j = 1; j <= K; ++j)
```

```
    for (int i = 0; i + (1 << j) <= mxN; ++i) {
```

```
        st[i][j] = min(st[i][j - 1], st[i + (1 << (j - 1))][j - 1]);
```

$st[i][j]$ съхранява отговора за интервала $[i, i + 2^j - 1]$ с дължина 2^j

Нека сега имаме заявка от вида $[L, R]$, където L е индекса на началото на интервала, а R е индекса на края на интервала, в който искаме да знаем минималния елемент. Дължината на този интервал е $R - L + 1$. Ще го разбием на два интервала (възможно е тези два интервала да са разбиване в пълния смисъл на думата или да се припокриват и да имат общ подинтервал – във втория случай това няма да влияе на верността на отговора, както споменахме в разглеждането на алгоритъма: [link sparse table](#)).

За целта дефинираме разделителя $s = \log[R - L + 1]$, т.е. качано по друг начин, s ще показва, най-голямата степен на двойката, която се съдържа в дължината на интервала. Следователно търсения минимум за заявката ще се изчислява по-следния начин:

```
int L, R;  
cin >> L >> R;  
--L, --R;  
int minimum = min(st[L][s], st[R - (1 << s) + 1][s]);
```

За което ще е необходима $O(1)$ сложност.