Sparse table

a = [7 9 3 0 1 2 5 9 3 6 3 3 2 0 1 8]

Sparse table нужна для определения минимума на отрезке за O(1).
Для построения надо NlogN времени и памяти.

t[0] = [7 9 3 0 1 2 5 9 3 6 3 3 2 0 1 8]
t[1] = [7 3 0 0 1 2 5 3 3 3 3 2 0 0 1] - минимумы из пар (i, i+1) на предыдущем уровне
t[2] = [0 0 0 0 1 2 3 3 3 2 0 0 0] - минимумы из пар (i, i+2) на предыдущем уровне
(это то же самое, что минимумы из (i, i+1, i+2, i+3) с 0-го уровня
t[3] = [0 0 0 0 1 2 0 0 0] - минимумы из пар (i, i+4) на предыдущем уровне
(это то же самое, что минимумы из (i, i+1, …., i+7) с 0-го уровня
t[4] = [0] - минимумы из пар (i, i+8) на предыдущем уровне
(это то же самое, что минимумы из (i, i+1, …., i+15) с 0-го уровня

Теперь пусть поступил запрос: найти минимум на [L,R].

Найдем длину отрезка: len = R - L + 1.
Выберем строку с минимальным индексом, в которой хранятся отрезки <= len.
Например, если len = 11, берем строку, где хранятся минимумы для отрезков длины 8 (3-я строка)
Например, если len = 6, берем строку, где хранятся минимумы для отрезков длины 4 (2-я строка)
Например, если len = 4, берем строку, где хранятся минимумы для отрезков длины 4 (2-я строка)

L=6, R=11
len = 6
[7 9 3 0 1 **2 5 9 3 6 3** 3 2 0 1 8]
берем 2-ую строку, т.к. она хранит длины 4

[0 0 0 0 1 **2** 3 **3** 3 2 0 0 0]
t[2][6] хранит минимумы из (6, 7, 8, 9) (это 2) [7 9 3 0 1 **2 5 9 3** 6 3 3 2 0 1 8]
t[2][8] хранит минимумы из (8, 9, 10, 11) (это 3) [7 9 3 0 1 2 5 **9 3 6 3** 3 2 0 1 8]
min(t[2][6], t[2][8]) дает ответ

Fenwick tree

https://i.gyazo.com/ba2b3fa6ee3c1a48bea365d781f1692b.png

В этом разделе массивы будут начинаться с единицы.

x	binary(x)	l(x)	индексы элементов, которые хранятся в fen[i]
1	00001	1	1
2	00010	2	1 2
3	00011	1	3
4	00100	4	1 2 3 4
5	00101	1	5
6	00110	2	5 6
7	00111	1	7
8	01000	8	1 2 3 4 5 6 7 8
9	01001	1	9
10	01010	2	9 10
11	01011	1	11
12	01100	4	9 10 11 12
13	01101	1	13
14	01110	2	13 14
15	01111	1	15
16	10000	16	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

т.е. например fen[12] = a[9] + a[10] + a[11] + a[12]

Как найти сумму a[1] + … + a[28]?
28 = 11100, fen[28] = a[25]+a[26]+a[27]+a[28]
24 = 11000, fen[24] = a[17] + ... + a[24]
16 = 10000, fen[16] = a[1] + … + a[16]

Как найти сумму a[1] + … + a[27]?
27 = 11011, fen[27] = a[27]
26 = 11010, fen[26] = a[25] + a[26]
24 = 11000, fen[24] = a[17] + ... + a[24]
16 = 10000, fen[16] = a[1] + … + a[16]

Каждый раз из индекса удаляется наименьший единичный бит
http://grepcode.com/file/repository.grepcode.com/java/root/jdk/openjdk/6-b14/java/lang/Integer.java#Integer.lowestOneBit%28int%29

```
// возвращает a[1] + ... + a[r]
int sum(int r) {
      int result = 0;
      while (r > 0) {
            result += fen[r];
            r -= Integer.lowestOneBit(r); // r -= r & -r;
      }
      return result;
}
```

Посмотрим на индекс 5. В каких строчках он встречался?

x	binary(x)	l(x)	индексы элементов, которые хранятся в fen[i]
5	00101	1	5
6	00110	2	5 6
8	01000	8	1 2 3 4 5 6 7 8
16	10000	16	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

```
// увеличивает a[i] на d
void inc(int i, int d) {
      while (i <= n) { // 1-indexed, осторожно
            fen[i] += d;
            i += Integer.lowestOneBit(i); // i += i & -i;
      }
}
```

Итого, дерево фенвика умеет:

1) модифицировать один элемент (прибавить / отнять)
2) посчитать сумму на [1, r]
3) посчитать сумму на [l, r] = sum(r) - sum(l-1)

А также можно для максимума

1) УВЕЛИЧИТЬ один элемент
2) посчитать максимум на [1,r] (на [L,R] нельзя)

Дерево Фенвика работает очень быстро. С ним можно сдавать с асимптотикой N * log^2(N)

Пример: надо увеличивать элементы массива и искать наименьший индекс, где сумма <= S.
Решение: увеличивать фенвик умеет, искать сумму умеет, остается к второму запросу прикрутить бинарный поиск.

(**UPD: это на самом деле за NlogN делается, т.к. можно накапливать сумму на префиксе внутри цикла, т.е. бинпоиск не нужен, лол)**
Ищем первый индекс, где сумма >= S (не тестил, но вроде должно работать).

```
step = 1 << 20
sum = 0
i = 0
while step > 0:
      if sum + fen[i + step] < S:
            sum += fen[i + step]
            i += step
      step /= 2
return i + 1
```

Ranged Fenwick tree
http://petr-mitrichev.blogspot.com/2013/05/fenwick-tree-range-updates.html
Умеет делать прибавление на отрезке.
(я сам до конца не разобрался, но можно копипастить, код рабочий)

Здесь 0-indexed код.
Чтобы получить 1-indexed код, надо

- **at |= (at + 1)** поменять на **at += at & -at**
- **at = (at & (at + 1)) - 1** поменять на **at -= at & -at**

```
// прибавить by на отрезке [left, right]
private void update(int left, int right, int by) {
    internalUpdate(left, by, -by * (left - 1));
    internalUpdate(right, -by, by * right);
}

private void internalUpdate(int at, int mul, int add) {
    while (at < dataMul.length) {
        dataMul[at] += mul;
        dataAdd[at] += add;
        at |= (at + 1);
    }
}

// сумма на [0, at]
private int query(int at) {
    int mul = 0;
    int add = 0;
    int start = at;
    while (at >= 0) {
        mul += dataMul[at];
        add += dataAdd[at];
        at = (at & (at + 1)) - 1;
    }
    return mul * start + add;
```

```
}
```

Задачи
http://codeforces.com/problemset/problem/514/D
http://acm.timus.ru/problem.aspx?space=1&num=2109
(мой код http://pastebin.com/ubG5kYbu - не смотрите его раньше времени)

http://acm.timus.ru/problem.aspx?space=1&num=1316
http://acm.timus.ru/problem.aspx?space=1&num=1028
http://acm.timus.ru/problem.aspx?space=1&num=1470
http://acm.timus.ru/problem.aspx?space=1&num=1090
http://acm.timus.ru/problem.aspx?space=1&num=1521
http://codeforces.com/contest/652/problem/D