Минимум и максимум

Даден е масив от N елемента, 2 < N < 50 000. Извършват се Q (1 ≤ Q ≤ 200 000) запитвания за минималният и максималният елемент измежду елементите с индекси между X и Y.

Входа съдържа няколко тестови примера. На първия ред на всеки от тях ще бъдат зададени две цели положителни числа N и Q, разделени с един интервал. На следващия ред са дадени N-те цели числа в интервала [-1 000 000, 1 000 000]. Следват Q реда съдържащи по две числа – X и Y. Край на входа е -1.

Изхода трябва да съдържа Q реда, с по две числа. Първото е минималния, а второто е и максималният елемент измежду елементите в интервала $[X_i, Y_i]$, $1 \le X_i \le Y_i \le N$.

Пример:

Вход	Изход
10 5 0 7 3 8 2 4 9 1 6 3 1 7 1 10 2 5 4 8 3 6 -1	0 9 0 9 2 8 1 9 2 8

Задачата се решава лесно с известен алгоритъм **RMQ** (**R**ange **M**inimum/**M**aximum **Q**uery).

Когато е възможна промяна на стойността на елемент става дума за динамично RMQ, ако такава операция липсва – за статично RMQ. Ще разгледаме статичното RMQ, каквото се изисква за решението на конкретната задача. Динамичния вариант го има добре обяснен тук.

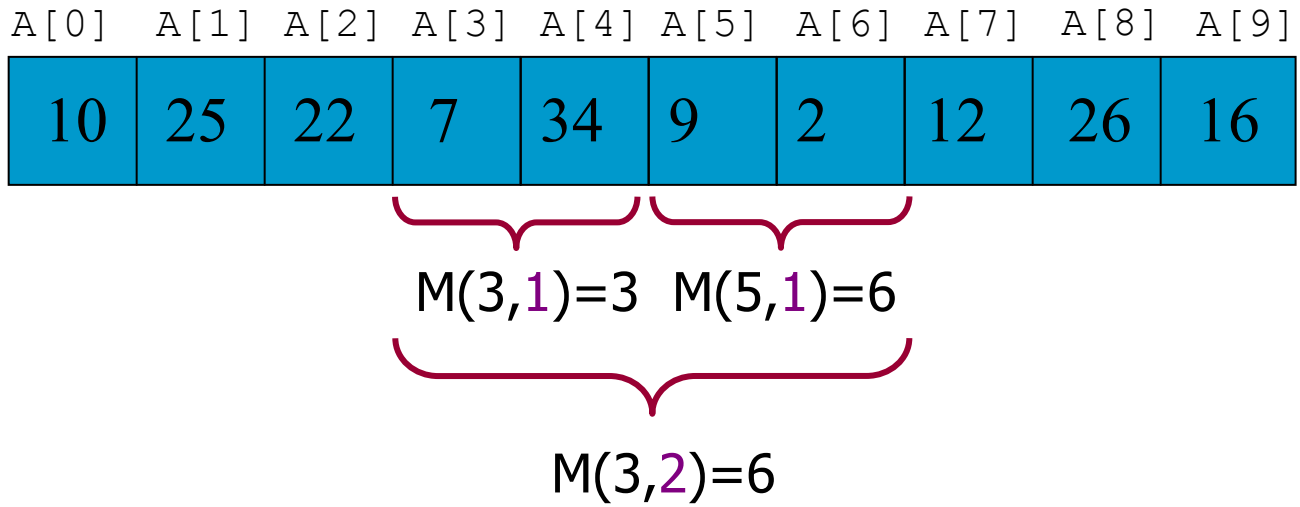
Съществуват няколко подхода за реализиране на статично и динамично RMQ. Сложността на RMQ се разделя на две части. Тъй като в различните алгоритми предварително се извършват някакви изчисления с цел по-бързото отговаряне на заявки се определя сложност на предварителните изчисления. Важна е и сложността, с която се отговаря на заявките. Ще разгледаме алгоритъм, със сложност **O(N log N)** за т.нар. препроцесинг и **O(1)** за отговор на заявка. Реализацията му е сравнително лесна и е подходящ за състезание.

Решението се базира на подхода динамично оптимиране. Идеята е да се пресметне предварително минималният елемент във всички интервали с дължина степен на двойката. Поддържаме двумерен масив с размерности M[0, logN][0, N-1], където в M[i][j] съхраняваме индекса на минималният елемент в интервал с начало i и дължина 2^j.

За да се пресметне стойността за един елемент M[i][j] въпросният интервал се разделя на две равни части, всеки с дължина 2^{j-1}. Тези два подинтервала са [i, i+2^{j-1}-1] и [i+2^{j-1}, i+2^j-1].

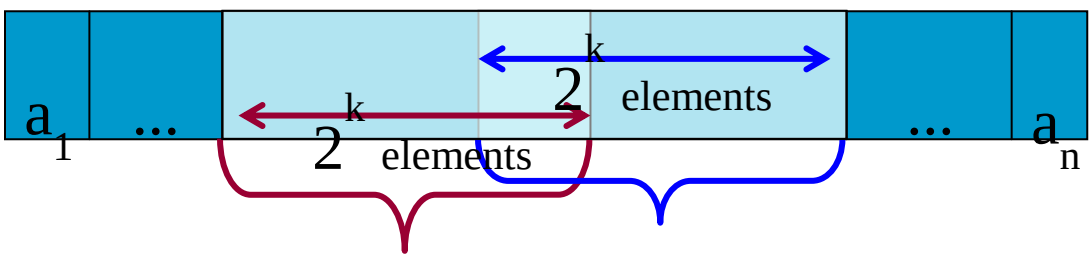
Минималните елементи в тези подинтервали са на позиции M[i][j-1] и M[i+2^j][j-1]. Ето защо M[i][j] = min(A[M[i][j-1]], A[M[i+2^j][j-1]]), като A е масивът с елементи. Стойностите на M се пресмятат в нарастващ ред на параметъра j. Така се гарантира, че когато се пресмята M[i][j] стойносттите за подинтервалите, на които се разделя големият интервал са вече пресметнати. Рекурентната формула е:

$$M[i,j] = \begin{cases} A[M[i,j-1]] \le A[M[i+2^{j-1}-1, j-1]] & M[i,j-1] \\ \text{Otherwise} & M[i+2^{j-1}-1, j-1] \end{cases}$$



При отговаряне на заявка се подава някакъв интервал [i, j], в който се търси индекса на минималният елемент. С така направените предварителни изчисления отговарянето на заявки става за константно време. Намира се максималното число, което е степен на двойката и не е по-голямо от дължината на зададеният интервал. Нека това число е 2^k. Интервалите [i, i+2^k-1] и [j-2^k+1, j] се застъпват. От тук следва, че ако се намерят минималните елементи в тези два интервала и се вземе по-малкият от двата той ще е минимален за търсеният интервал [i, j]. Индексите на минималните елементи в двата интервала са M[i,k] и M[j-2^k+1, k]. Отговора на заявката е min(A[M[i,k]], A[M[j-2^k+1, k]]). Имаме следната формула:

$$RMQ(i,j) = \begin{cases} A[M[i,k]] \le A[M[j-2^k+1,k]] & M[i,k] \\ \text{Otherwise} & M[j-2^k+1,k] \end{cases}$$



Примерна реализация на статично RMQ, за търсене на минимален елемент:

```
vector <int> A;
vector <vector<int>> M;

int minA(int i, int j)
{
      return A[i] < A[j] ? i : j;
}

int Log2(int val)
{
    int ret = 0;
    while (val >>= 1)
         ++ret;
    return ret;
}

void precompute()
{
      int i, j, ln = Log2(N);
      M.clear();
      M.resize(ln + 1, vector<int>(N));

      //initialize M for the intervals with length 1
      for(int i = 0; i < N; i++)
            M[0][i] = i;

      //compute values from smaller to bigger intervals
      for(i = 1; 1 << i <= N; i++)
            for(j = 0; j + (1 << i) - 1 < N; j++)
            {
                  int k = j + (1 << (i - 1));
                  M[i][j] = minA(M[i - 1][j], M[i - 1][k]);
            }
}

int rmqMin(int beg, int end)
{
      beg--,end--;
      int ln = Log2(end - beg+1);
      int k = end - (1 << ln) + 1;
      return min(A[M[ln][beg]], A[M[ln][k]]);
}
```

По аналогичен начин се намира и максималния елемент в даден интервал. Това е реализирано в авторското решение.

Използвана литература:

http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor

http://judge.openfmi.net:9080/mediawiki/index.php/RMQ