SPOJ KGSS

Maximum Sum

Prof. Edson Alves – UnB/FCTE

Problema

You are given a sequence $A[1], A[2], \ldots, A[N]$ ($0 \leq A[i] \leq 10^8$, $2 \leq N \leq 10^5$). There are two types of operations and they are defined as follows:

Update:

This will be indicated in the input by a 'U' followed by space and then two integers i and x.

U i x, $1 \leq i \leq N$, and x, $0 \leq x \leq 10^8$.

This operation sets the value of $A[i]$ to x.

Query:

This will be indicated in the input by a 'Q' followed by a single space and then two integers x and y.

Q x y, $1 \leq x < y \leq N$.

You must find i and j such that $x \leq i, j \leq y$ and $i \neq j$, such that the sum $A[i] + A[j]$ is maximized. Print the sum $A[i] + A[j]$.

Entrada e saída

Input

The first line of input consists of an integer N representing the length of the sequence. Next line consists of N space separated integers $A[i]$. Next line contains an integer $Q, Q \leq 10^5$, representing the number of operations. Next Q lines contain the operations.

Output

Output the maximum sum mentioned above, in a separate line, for each Query.

Exemplo de entradas e saídas

Sample Input

```
5
1 2 3 4 5
6
Q 2 4
Q 2 5
U 1 6
Q 1 5
U 1 7
Q 1 5
```

Sample Output

```
7
9
11
12
```

Solução

- Um algoritmo *naive*, que percorre o intervalo $[x, y]$ em busca destes valores tem complexidade $O(QN)$ no pior caso, o que leva ao TLE
- Para melhorar esta complexidade, observe primeiro que os índices i, j que maximizam a soma $A[i] + A[j]$ correspondem aos dois maiores elementos no intervalo $[x, y]$
- Assim, pode-se utilizar uma árvore de segmentos para manter, para cada intervalo, os valores de seus dois maiores elementos
- Nas folhas, devem ser armazenados os pares $(A[i], 0)$
- Em cada nó, é preciso avaliar os pares armazenados nos filhos à esquerda e à direita, e escolher dentre eles os dois maiores
- Esta solução terá complexidade $O(Q \log N)$ no pior caso, de modo que a solução será aceita

Solução AC com complexidade $O(Q \log N)$

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;
using ii = pair<int, int>;

class SegmentTree
{
public:

    SegmentTree(const std::vector<ii>& xs) : N(xs.size()), ns(4*N)
    {
        for (size_t i = 0; i < xs.size(); ++i)
            update(i, xs[i]);
    }

    void update(int i, const ii& value)
    {
        update(1, 0, N - 1, i, value);
    }
```

Solução AC com complexidade $O(Q \log N)$

```
22      ll query(int a, int b)
23      {
24          auto ans = RSQ(1, 0, N - 1, a, b);
25          return ans.first + ans.second;
26      }
27
28  private:
29
30      int N;
31      std::vector<ii> ns;
32
33      void update(int node, int L, int R, int i, const ii& value) {
34          if (i > R or i < L)
35              return;
36
37          if (L == R)
38          {
39              ns[node] = value;
40              return;
41          }
```

Solução AC com complexidade $O(Q \log N)$

```
43          update(2*node, L, (L+R)/2, i, value);
44          update(2*node + 1, (L+R)/2 + 1, R, i, value);
45
46          vector<ll> ys { ns[2*node].first, ns[2*node + 1].first,
47              ns[2*node].second, ns[2*node + 1].second };
48
49          sort(ys.begin(), ys.end());
50
51          ns[node] = ii(ys[3], ys[2]);
52      }
53
54      ii RSQ(int node, int L, int R, int a, int b) {
55          if (a > R or b < L)
56              return ii(0, 0);
57
58          if (a ⩽ L and R ⩽ b)
59              return ns[node];
60
61          auto x = RSQ(2*node, L, (L + R)/2, a, b);
62          auto y = RSQ(2*node + 1, (L + R)/2 + 1, R, a, b);
```

Solução AC com complexidade $O(Q \log N)$

```
64          vector<ll> ys { x.first, x.second, y.first, y.second };
65
66          sort(ys.begin(), ys.end());
67
68          return ii(ys[3], ys[2]);
69      }
70  };
71
72  int main()
73  {
74      ios::sync_with_stdio(false);
75
76      int N; cin >> N;
77
78      vector<ii> xs(N, ii(0, 0));
79
80      for (int i = 0; i < N; ++i)
81          cin >> xs[i].first;
```

Solução AC com complexidade $O(Q \log N)$

```
83      auto tree = SegmentTree(xs);
84      int Q; cin >> Q;
85
86      while (Q--) {
87          string cmd;
88          int x, y;
89
90          cin >> cmd >> x >> y;
91
92          switch (cmd.front()) {
93          case 'U':
94              tree.update(x - 1, ii(y, 0));
95              break;
96          default:
97              cout << tree.query(x - 1, y - 1) << '\n';
98          }
99      }
100
101     return 0;
102 }
```