

Árvore de Fenwick

Definição, *RSQ* e *update*: problemas resolvidos

Prof. Edson Alves - UnB/FGA

2019

1. SPOJ – Fenwick Trees
2. UVA 12798 – Handball

SPOJ – Fenwick Trees

Problema

Mr. Fenwick has an array a with many integers, and his children love to do operations on the array with their father. The operations can be a query or an update.

For each query the children say two indices l and r , and their father answers back with the sum of the elements from indices l to r (both included).

When there is an update, the children say an index i and a value x , and Fenwick will add x to a_i (so the new value of a_i is $a_i + x$).

Because indexing the array from zero is too obscure for children, all indices start from 1. Fenwick is now too busy to play games, so he needs your help with a program that plays with his children for him, and he gave you an input/output specification.

Input

The first line of the input contains N ($1 \leq N \leq 10^6$). The second line contains N integers a_i ($-10^9 \leq a_i \leq 10^9$), the initial values of the array. The third line contains Q ($1 \leq Q \leq 3 \times 10^5$), the number of operations that will be made. Each of the next Q lines contains an operation. Query operations are of the form “q l r ” ($1 \leq l \leq r \leq N$), while update operations are of the form “u i x ” ($1 \leq i \leq N$, $-10^9 \leq x \leq 10^9$).

Output

You have to print the answer for every query in a different line, in the same order of the input.

Exemplo de entradas e saídas

Sample Input

10

3 2 4 0 42 33 -1 -2 4 4

6

q 3 5

q 1 10

u 5 -2

q 3 5

u 6 7

q 4 7

Sample Output

46

89

44

79

Solução

- A solução *naive* consiste em percorrer cada intervalo a cada consulta, de modo que a complexidade seria igual a $O(QN)$, onde Q é o número de *queries* do tipo q
- Como $Q \leq 3 \times 10^5$ e $N \leq 10^6$, esta solução levaria ao TLE
- O uso de uma árvore de Fenwick permite responder cada uma das *queries* com complexidade $O(\log N)$
- A construção da árvore tem complexidade $O(N \log N)$, de modo que a solução teria complexidade $O((N + Q) \log N)$
- É preciso tomar cuidado com possíveis *overflows*, usando o tipo **long long** para armazenar as informações dos nós da árvore

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

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4 using ll = long long;
5
6 struct BITree {
7     vector<ll> ts;
8     size_t N;
9
10    BITree(size_t n) : ts(n + 1, 0), N(n) {}
11
12    ll LSB(ll n) { return n & (-n); }
13
14    void add(size_t i, ll x)
15    {
16        while (i <= N)
17        {
18            ts[i] += x;
19            i += LSB(i);
20        }
21    }
```


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

```
22
23     ll RSQ(size_t i, size_t j)
24     {
25         return RSQ(j) - RSQ(i - 1);
26     }
27
28     ll RSQ(size_t k)
29     {
30         ll sum = 0;
31
32         while (k)
33         {
34             sum += ts[k];
35             k -= LSB(k);
36         }
37
38         return sum;
39     }
40 };
41
```

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

```
42 int main()
43 {
44     ios::sync_with_stdio(false);
45
46     size_t N;
47     cin >> N;
48
49     BITree ft(N);
50
51     for (size_t i = 1; i <= N; ++i)
52     {
53         int a;
54         cin >> a;
55
56         ft.add(i, a);
57     }
58
59     int Q;
60     cin >> Q;
61
```

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

```
62  while (Q--)
63  {
64      string cmd;
65      ll L, R;
66
67      cin >> cmd >> L >> R;
68
69      switch (cmd[0]) {
70      case 'q':
71          cout << ft.RSQ(L, R) << '\n';
72          break;
73
74      default:
75          ft.add(L, R);
76      }
77  }
78
79  return 0;
80 }
```

UVA 12798 – Handball

Frustrated and disappointed with the results of its football team, the Super Brazilian Club (SBC) decided to invest in the handball team. In order to better rate the players, the coaches would like to analyse their regularity. Specifically, they are interested in knowing how many players scored goals in all matches.

As the data volume is very big, they would like to have a computer program to do this counting.

Input

The input contains several test cases. The first line of a test case contains two integers N and M ($1 \leq N \leq 100$ and $1 \leq M \leq 100$) indicating, respectively, the number of players and the number of matches. Each one of the next N lines describes the performance of one player: the i -th line contains M integers X_j ($0 \leq X_j \leq 100$, for $1 \leq j \leq M$), giving the number of goals that the i -th player scored in each match.

Output

For each test case in the input your program must output one line, containing one integer, the number of players that scored goals in all matches!

Exemplo de entradas e saídas

Sample Input

```
5 3
0 0 0
1 0 5
0 0 0
0 1 2
1 1 0
12 5
4 4 2 3 7
0 0 0 1 0
7 4 7 0 6
1 2 3 3 2
0 0 0 0 0
4 0 9 10 10
0 1 0 0 0
1 2 0 2 3
10 10 10 1 0
0 3 3 3 4
10 10 0 10 10
1 1 2 0 9
```

Sample Output

```
0
2
```

Solução com complexidade $O(NM \log MN)$

- O problema consiste em determinar quantos jogadores fizeram gols em todas as partidas
- Uma maneira de se manter este registro é inicializar uma árvore de Fenwick onde todos os elementos da sequência a_k são iguais a um
- Assim, para cada jogador, caso ele não marque nenhum gol em uma determinada partida, basta fazer $a_i = 0$ atrás da soma de -1 na posição i
- Este processo deve ser feito, no máximo, uma vez por jogador
- Ao final, a resposta será $RSQ(1, N)$

Solução com complexidade $O(NM \log NM)$

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 struct BITree {
6     vector<int> ts;
7     size_t N;
8
9     BITree(size_t n) : ts(n + 1, 0), N(n) {}
10
11     int LSB(int n) { return n & (-n); }
12
13     void add(size_t i, int x)
14     {
15         while (i <= N)
16         {
17             ts[i] += x;
18             i += LSB(i);
19         }
20     }
21 }
```

Solução com complexidade $O(NM \log NM)$

```
22  int RSQ(size_t i, size_t j)
23  {
24      return RSQ(j) - RSQ(i - 1);
25  }
26
27  int RSQ(size_t k)
28  {
29      int sum = 0;
30
31      while (k)
32      {
33          sum += ts[k];
34          k -= LSB(k);
35      }
36
37      return sum;
38  }
39
```

Solução com complexidade $O(NM \log NM)$

```
40 void reset(size_t i)
41 {
42     if (RSQ(i, i) == 1)
43         add(i, -1);
44 }
45 };
46
47 int main()
48 {
49     ios::sync_with_stdio(false);
50
51     size_t N, M;
52
53     while (cin >> N >> M)
54     {
55         BITree ft(N);
56
57         for (size_t i = 1; i <= N; ++i)
58         {
59             ft.add(i, 1);
60         }
```

Solução com complexidade $O(NM \log NM)$

```
61         for (size_t j = 1; j <= M; ++j)
62         {
63             int goals;
64             cin >> goals;
65
66             if (goals == 0)
67                 ft.reset(i);
68         }
69     }
70
71     cout << ft.RSQ(1, N) << '\n';
72 }
73
74 return 0;
75 }
```

Solução $O(NM)$

- Observe que o problema é estático: o número de gols marcados por cada jogador não muda dinamicamente, e a pergunta só é feita uma vez, ao final
- Assim, não é necessário o uso de uma árvore de Fenwick
- O registro de cada jogador pode ser feito em um vetor de *bits*, onde 1 significa que ele marcou gols em todos os jogos
- A implementação fica semelhante à anterior, porém com acesso aos elementos em $O(1)$
- O método `count()` do `bitset` da STL pode ser utilizada para totalizar o número de jogadores que marcaram gols em todos os jogos

Solução com complexidade $O(NM)$

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 const int MAX { 110 };
6
7 int main()
8 {
9     ios::sync_with_stdio(false);
10
11     size_t N, M;
12
13     while (cin >> N >> M)
14     {
15         bitset<MAX> bs;
16
17         for (size_t i = 1; i <= N; ++i)
18         {
19             bs[i] = true;
20         }
```

Solução com complexidade $O(NM)$

```
21         for (size_t j = 1; j <= M; ++j)
22         {
23             int goals;
24             cin >> goals;
25
26             if (goals == 0)
27                 bs[i] = false;
28         }
29     }
30
31     cout << bs.count() << '\n';
32 }
33
34 return 0;
35 }
```

1. [SPOJ – Fenwick Trees](#)
2. [UVA 12798 – Handball](#)