OJ 12978

Handball

Prof. Edson Alves – UnB/FCTE

Problema

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.

Entrada e saída

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 $(0 \leq X_j \leq 100, \text{for } 1 \leq j \leq M)$, 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 $O(NM \log N)$

- 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 Fenwinck 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 $O(NM \log N)$

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

using namespace std;

struct BITree {
    vector<int> ts;
    size_t N;

    BITree(size_t n) : ts(n + 1, 0), N(n) {}

    int LSB(int n) { return n & (-n); }

    void add(size_t i, int x)
    {
        while (i ⩽ N)
        {
            ts[i] += x;
            i += LSB(i);
        }
    }
```

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

```
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      }
```

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

```
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);
```

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

```
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 $O(NM)$

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

using namespace std;

const int MAX { 110 };

int main()
{
    ios::sync_with_stdio(false);

    size_t N, M;

    while (cin >> N >> M)
    {
        bitset<MAX> bs;

        for (size_t i = 1; i <= N; ++i)
        {
            bs[i] = true;
```

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  }
```