POJ 1195

Mobile Phones

Prof. Edson Alves – UnB/FGA

Problema

Suppose that the fourth generation mobile phone base stations in the Tampere area operate as follows. The area is divided into squares. The squares form an $S \times S$ matrix with the rows and columns numbered from 0 to $S-1$. Each square contains a base station. The number of active mobile phones inside a square can change because a phone is moved from a square to another or a phone is switched on or off. At times, each base station reports the change in the number of active phones to the main base station along with the row and the column of the matrix.

Write a program, which receives these reports and answers queries about the current total number of active mobile phones in any rectangle-shaped area.

Entrada e saída

Input

The input is read from standard input as integers and the answers to the queries are written to standard output as integers. The input is encoded as follows. Each input comes on a separate line, and consists of one instruction integer and a number of parameter integers according to the following table.

Instruction	Parameters	Meaning
0	S	Initialize the matrix size to $S \times S$ containing all zeros. This instruction is given only once and it will be the first instruction

Entrada e saída

Instruction	Parameters	Meaning
1	$X\ Y\ A$	Add A to the number of active phons in table square (X, Y). A may be positve or negative.
2	$L\ B\ R\ T$	Query the current sum of numbers of active mobile phones in squares (X, Y), where $L \leq X \leq R, B \leq Y \leq T$
3		Terminate program. This instruction is given only once and it will be the last instruction.

Entrada e saída

The values will always be in range, so there is no need to check them. In particular, if A is negative, it can be assumed that it will not reduce the square value below zero. The indexing starts at 0, e.g. for a table of size 4×4, we have $0 \leq X \leq 3$ and $0 \leq Y \leq 3$.

Table size: $1 \times 1 \leq S \times S \leq 1024 \times 1024$

Cell value V at any time: $0 \leq V \leq 32767$

Update amount: $-32768 \leq A \leq 32767$

No of instructions in input: $3 \leq U \leq 60002$

Maximum number of phones in the whole table: $M = 2^{30}$

Entrada e saída

Output

Your program should not answer anything to lines with an instruction other than 2. If the instruction is 2, then your program is expected to answer the query by writing the answer as a single line containing a single integer to standard output.

Exemplo de entradas e saídas

Sample Input

```
0 4
1 1 2 3
2 0 0 2 2
1 1 1 2
1 1 2 -1
2 1 1 2 3
3
```

Sample Output

```
3
4
```

Solução

- Este problema pode ser resolvido diretamente através da implementação de uma árvore de Fenwick bidimensional
- Deve-se tomar alguns cuidados, porque o juiz roda em servidores antigos e os compiladores não são os mais recentes disponíveis
- Evite qualquer elemento que foi introduzido na linguagem C++ após a versão 98
- Evite os contêiners da STL
- Os índices usados começam em zero, e isto deve ser refletido na implementação
- O cabeçalho `bits/stdc++.h` não está disponível

Solução com complexidade $O(T(N+K)\log(N+K))$

```cpp
#include <iostream>
#include <vector>

using namespace std;

const int MAX { 1030 };
int ft[MAX][MAX];

class BITree2D {
public:
    int N;

    BITree2D() : N(0) {}

    void set_n(size_t n) { N = n; }

    int RSQ(int a, int b, int c, int d)          // Range query
    {
        return RSQ(c, d) - RSQ(c, b-1) - RSQ(a-1, d) + RSQ(a-1, b-1);
    }
```

Solução com complexidade $O(T(N+K)\log(N+K))$

```
22      void add(int x, int y, int v)               // Point update
23      {
24          for (int i = x; i <= N; i += LSB(i))
25              for (int j = y; j <= N; j += LSB(j))
26                  ft[i][j] += v;
27      }
28
29  private:
30      int LSB(int n) { return n & -n; }
31
32      int RSQ(int a, int b) {
33          int sum = 0;
34
35          for (int i = a; i > 0; i -= LSB(i))
36              for (int j = b; j > 0; j -= LSB(j))
37                  sum += ft[i][j];
38
39          return sum;
40      }
41  };
```

Solução com complexidade $O(T(N+K)\log(N+K))$

```
43 int main()
44 {
45     int cmd, a, b, c, d;
46     BITree2D ft;
47 
48     while (cin >> cmd, cmd != 3)
49     {
50         switch (cmd) {
51         case 0:
52             cin >> a;
53             ft.set_n(a);
54             break;
55 
56         case 1:
57             cin >> a >> b >> c;
58             ft.add(a + 1, b + 1, c);
59             break;
```

Solução com complexidade $O(T(N+K)\log(N+K))$

```
61          default:
62              cin >> a >> b >> c >> d;
63              cout << ft.RSQ(a + 1, b + 1, c + 1, d + 1) << '\n';
64          }
65      }
66
67      return 0;
68  }
```