

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерных технологий

Продвинутые алгоритмы и структуры данных

Отчет о задаче № В

Выполнила
студентка

Ершова А. И.

Группа № Р4115

Преподаватель: Косяков Михаил Сергеевич

г. Санкт-Петербург

2024

Условие задачи

В. Количество единиц в двоичной записи

Ограничение времени	1 секунда
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Вам дано число $x = 0$, а также n операций, которые необходимо над ним совершить. Операции бывают 2-х типов:

- «+ S » — прибавить к числу x число 2^S ;
- «− S » — отнять от числа x число 2^S .

После выполнения каждой операции необходимо вывести количество единиц в двоичной записи числа x .

Формат ввода

Первая строка входных данных содержит целое число n ($1 \leq n \leq 10^5$) — количество операций.

Следующие n строк описывают операции.

Каждая операция задается символом «+» или «−» и числом S ($0 \leq S \leq 10^5$), записанными через пробел.

Гарантируется, что после выполнения любого количества заданных операций x остается неотрицательным.


Формат вывода

Для каждой из n операций в отдельной строке выведите единственное число — количество единиц в двоичной записи числа x после выполнения данной операции.

Пример 1

Ввод 	Вывод 
2	1
+ 1	0
− 1	

Пример 2

Ввод 	Вывод 
4	1
+ 2	2
+ 8	6
− 3	7
− 0	

Решение:

```
#include <fstream>
#include <iostream>
#include <vector>

using namespace std;

vector<bool> bits;
int bits_size = 0;
int countOnes = 0;

void add(int S) {
    if (bits.empty()) {
        for (int i = 0; i <= S; i++) {
            if (i == S) {
                bits.push_back(true);
                countOnes++;
                bits_size++;
            } else {
                bits.push_back(false);
                bits_size++;
            }
        }
        return;
    }
    if (bits_size < S + 1) {
        for (int i = bits_size; i <= S; i++) {
            if (i == S) {
                bits.push_back(true);
                countOnes++;
                bits_size++;
            } else {
                bits.push_back(false);
                bits_size++;
            }
        }
        return;
    }
    if (!bits[S]) {
        bits[S] = true;
        countOnes++;
        return;
    }
    for (int i = S; i < bits_size; i++) {
        if (!bits[i]) {
            bits[i] = true;
            countOnes++;
        }
    }
}
```

```

        return;
    } else {
        bits[i] = false;
        countOnes--;
    }
    if (i + 1 == bits_size) {
        bits.push_back(true);
        countOnes++;
        bits_size++;
        return;
    }
}

}

void sub(int S) {
    if (bits[S]) {
        bits[S] = false;
        countOnes--;
        return;
    }
    for (int i = S; i < bits_size; i++) {
        if (bits[i]) {
            bits[i] = false;
            countOnes--;
            return;
        } else {
            bits[i] = true;
            countOnes++;
        }
    }
}

int main() {
    ifstream fin("../input.txt");
    int n;
    fin >> n;

    for (int i = 0; i < n; i++) {
        char op;
        int S;
        fin >> op >> S;
        if (op == '+') {
            add(S);
        }
        if (op == '-') {
            sub(S);
        }
        cout << countOnes << '\n';
    }
}

```

Сложность: $O(N)$.

Объяснение решения:

Число x хранится в двоичном виде в bool массиве `bits`. В этом массиве `false` означает 0, а `true` означает 1. Изначально массив `bits` не содержит элементов, он расширяется по мере необходимости.

В методах `add()` и `sub()` реализуется сложение и вычитание столбиком со степенью двойки S . Так как по условию x не может быть отрицательным, в функции `add()` гораздо больше логики, т.к. в нем еще реализовано динамическое расширение массива `bits`.

Также пришлось для удобства создать специальную переменную `bits_size`, в которой хранится размер массива `bits`, т. к. санитайзер ругается на использование `bits.size()` и приведения его к `int`.