Санкт-Петербургский национальный исследовательский университет ИТМО

Факультет систем управления и робототехники

Алгоритмы и структуры данных

Отчёт по лабораторной работе №3 (1155)

Преподаватель: Тропченко А. А.

Выполнил: Марухленко Д. С.

Группа: R3235

Санкт Петербург, 2021г.

1. Цель работы

Решить задачу №1155 на платформе Timus Online Judge
https://acm.timus.ru/problem.aspx?space=1&num=1155

2. Задача

Условие

Архангел по науке докладывает:
— Господи, эти физики там, внизу, — они открыли ещё одну элементарную частицу!
— Хорошо, добавим параметр в Общее Уравнение Вселенной.

С развитием техники физики находят всё новые и новые элементарные частицы, с непонятными и даже загадочными свойствами. Многие слышали про мюоны, глюоны, странные кварки и прочую нечисть. Недавно были обнаружены элементарные частицы дуоны. Эти частицы названы так потому, что учёным удаётся создавать или аннигилировать их только парами. Кстати, от дуонов одни неприятности, поэтому от них стараются избавляться до начала экспериментов. Помогите физикам избавиться от дуонов в их установке.

Экспериментальная установка состоит из восьми камер, которые расположены в вершинах куба. Камеры промаркированы латинскими буквами A, B, C, …, H. Технически возможно создать, или наоборот, аннигилировать, два дуона, находящихся в смежных камерах. Вам нужно автоматизировать процесс удаления дуонов из установки.
Ограничение времени: 1.0 секунды
Ограничение памяти: 64 МБ

Исходные данные

В единственной строке даны восемь целых чисел в пределах от 0 до 100, описывающих количество дуонов в камерах установки (сначала в камере A, потом в B, и т.д.).

Результат

Выведите последовательность действий для удаления всех дуонов или слово IMPOSSIBLE, если это невозможно. Каждое действие должно быть описано в отдельной строке, в следующем формате: маркер первой камеры, маркер второй (смежной с первой), далее плюс либо минус (создать или аннигилировать пару дуонов). Количество действий в последовательности не должно превосходить 1000.

Пример

Исходные данные	**Результат**
0 1 0 1 2 3 2 2	IMPOSSIBLE
1 0 1 0 3 1 0 0	EF- EA- AD+ AE- DC-

3. Материалы работы

3.1. Объяснение алгоритма

Для решения задачи разобьём вершины куба на две группы несмежных вершин: ACFH и BDEG. Решить задачу невозможно только в одном случае: сумма весов в группах вершин не совпадает. Для хранения весов вершин используем структуру, хранящую имя вершины и её вес. После проверки решаемости задач, используя цикл, для каждой вершины первой группы будем проверять возможность и проводить аннигиляцию со смежными вершинами. Если существует вершина из другой группы, не смежная с данной, но имеющая вес, добавим вас двум другим вершинам, соединяющим данную с вершиной из другой группы.

3.2. Код программы.

```
#include <iostream>

using namespace std;

struct Point{
    char ch;
    int n;
};

void minus_p(Point* x, Point* y) {
    --(x->n);
    --(y->n);
    cout << y->ch << x->ch << '-' << endl;
};

void plus_p(Point* x, Point* y) {
    ++(x->n);
    ++(y->n);
    cout << y->ch << x->ch << '+' << endl;
};

int main() {
    Point a = {'A'}, b = {'B'}, c = {'C'}, d = {'D'}, e = {'E'}, f = {'F'}, g = {'G'}, h = {'H'};
    cin >> a.n >> b.n >> c.n >> d.n >> e.n >> f.n >> g.n >> h.n;
    if (a.n + c.n + f.n + h.n != b.n + d.n + e.n + g.n) {
        cout << "IMPOSSIBLE" << endl;
        return 0;
    }

    while (a.n + c.n + f.n + h.n > 0) {
        if (a.n > 0) {
            if (b.n > 0) minus_p (&a, &b);
            else if (d.n > 0) minus_p(&a, &d);
            else if (e.n > 0) minus_p(&a, &e);
            else if (g.n > 0) {
                plus_p(&f,&b);
                minus_p (&a, &b);
            }
        }
        else if (h.n > 0) {
            if (g.n > 0) minus_p(&h, &g);
            else if (e.n > 0) minus_p(&h, &e);
            else if (d.n > 0) minus_p(&h, &d);
            else if (b.n > 0) {
                plus_p(&d,&c);
                minus_p(&h, &d);
            }
        }
        else if (f.n > 0) {
            if (b.n > 0) minus_p(&f, &b);
            else if (e.n > 0) minus_p(&f, &e);
            else if (g.n > 0) minus_p(&f, &g);
```

```
53.             else if (d.n > 0) {
54.                 plus_p(&a,&b);
55.                 minus_p(&f, &b);
56.             }
57.         }
58.         else if (c.n > 0) {
59.             if (g.n > 0) minus_p(&c, &g);
60.             else if (d.n > 0) minus_p(&c, &d);
61.             else if (b.n > 0) minus_p(&b, &c);
62.             else if (e.n > 0) {
63.                 plus_p(&f,&b);
64.                 minus_p(&b, &c);
65.             }
66.         }
67.     }
68.     return 0;
69. }
```

4. Результат выполнения и ссылка на репозиторий GitHub

9269633	22:57:14 15 мар 2021	Daniil Marukhlenko	1155	Visual C++ 2019	Accepted		0.001	212 КБ

https://github.com/japersik/algorithms_and_data_structures

5. Вывод

Работа выполнена, задача решена. Используемый алгоритм соответствует требованиям реализации по памяти и времени выполнения.