Задача №1005. Куча камней

У вас есть несколько камней известного веса $w_1, \ldots, w_n$. Напишите программу, которая распределит камни в две кучи так, что разность весов этих двух куч будет минимальной.

Исходные данные:

> Ввод содержит количество камней n ($1 \leq n \leq 20$) и веса камней $w_1, \ldots, w_n$ ($1 \leq w_i \leq 100000$) — целые, разделённые пробельными символами.

Результат:

> Ваша программа должна вывести одно число — минимальную разность весов двух куч.

Рабочий код

```cpp
#include <iostream>
using namespace std;

long search(long array[], int size, int i = 0, long sum1 = 0, long sum2 = 0)
{
    static long min_ = 100000;
    if (i == size) min_ = min(min_, abs(sum1 - sum2));
    else
    {
        search(array, size, i + 1, sum1 + array[i], sum2);
        search(array, size, i + 1, sum1, sum2 + array[i]);
    }
    return min_;
}

int main()
{
    int n;
    cin >> n;
    long rocks[n];
    for (int i = 0; i < n; i++) cin >> rocks[i];
    cout << search(rocks, n) << endl;
    return 0;
}
```

Объяснение алгоритма

Этот код решает задачу распределения камней в две кучи так, чтобы разность весов этих куч была минимальной. В начале программы считывается количество камней и их веса. Затем программа вызывает рекурсивную функцию `search`, которая перебирает все возможные варианты распределения камней в две кучи и находит минимальную разность весов этих куч. Функция `search` принимает массив камней, его размер, индекс текущего камня, сумму весов камней в первой куче и сумму весов камней во второй куче. Если индекс текущего камня равен размеру массива, функция вычисляет разность весов куч и обновляет минимальное значение разности. В противном случае функция вызывает себя дважды: с текущим камнем в первой куче и с текущим камнем во второй куче. Функция возвращает минимальное значение разности весов куч. В конце программы выводится минимальное значение разности весов куч. Этот код решает задачу за $O(2^n)$ времени, где n — количество камней.

Задача №1155. Дуоны

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.

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.

Рабочий код

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

Объяснение алгоритма

Этот код решает задачу удаления дуонов из установки, используя жадный алгоритм. В начале программы создаются структуры `Point` для каждой камеры, содержащие маркер камеры и количество дуонов в ней. Затем программа считывает количество дуонов в каждой камере и проверяет, можно ли удалить все дуоны. Если это невозможно, программа выводит «IMPOSSIBLE» и завершает работу. В противном случае программа начинает удаление дуонов, используя жадный алгоритм. Пока в установке есть дуоны, программа проверяет, в каких камерах они находятся, и удаляет их, создавая или аннигилируя пары дуонов в смежных камерах. Каждое действие выводится в формате: маркер первой камеры, маркер второй (смежной с первой), далее плюс либо минус (создать или аннигилировать пару дуонов). Программа завершает работу, когда все дуоны удалены.

Задача №1207. Медиана на плоскости

На плоскости находятся N точек (N чётно). Никакие три точки не лежат на одной прямой. Ваша задача — выбрать две точки так, что прямая линия, проходящая через них, делит множество точек на две части одинакового размера.

Исходные данные:

> Первая строка содержит целое число N ($4 \le N \le 10^4$). Каждая из следующих N строк содержит пары целых чисел x_i, y_i ($-106 \le x_i, y_i \le 106$) — координаты i-й точки.

Результат:

> Выведите номера выбранных точек.

Рабочий код

```cpp
#include <iostream>
#include <algorithm>
#include <limits>
using namespace std;

struct Point {
    int x;
    int y;
    int number;
    double tang;
};

bool compare_angle(Point p1, Point p2) { return p2.tang > p1.tang; }

int main() {
    int n;
    cin >> n;
    Point points[n];
    int minIndex = 0;
    for (int i = 0; i < n; i++) {
        cin >> points[i].x >> points[i].y;
        points[i].number = i + 1;
        if (points[i].x < points[minIndex].x)
            minIndex = i;
    }
    int min_x = points[minIndex].x;
    int min_y = points[minIndex].y;
    for (int i = 0; i < n; i++) {
        points[i].x = points[i].x - min_x;
        points[i].y = points[i].y - min_y;
        if (points[i].x == 0 ) {
            if (points[i].y > 0) {points[i].tang = numeric_limits<double>::max();}
            else if (points[i].y < 0) {points[i].tang = numeric_limits<double>::lowest()+0.000000001;}
            else if (points[i].y == 0) {points[i].tang = numeric_limits<double>::lowest();}
        }
        else points[i].tang = ((double)points[i].y)/((double) points[i].x);

    }
    sort(points, points+n, compare_angle);

    cout << minIndex+1<< " " << points[n / 2].number << endl;
    return 0;
}
```

Объяснение алгоритма

Этот код решает задачу выбора двух точек так, чтобы прямая линия, проходящая через них, делит множество точек на две части одинакового размера. В начале программы считывается количество точек и их координаты. Затем программа находит точку с минимальной координатой x и пересчитывает координаты всех точек относительно этой точки. Для каждой точки программа вычисляет тангенс угла между прямой, проходящей через точку и точку с минимальной координатой x, и осью x. Если x-координата точки равна 0, программа присваивает максимальное значение тангенса, если y-координата больше 0, и минимальное значение тангенса, если y-координата меньше 0. Затем программа сортирует точки по убыванию тангенса угла. В конце программы выводятся номера точек с минимальной координатой x и средней по тангенсу угла точек. Этот код решает задачу за $O(n \log n)$ времени, где n — количество точек.