Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет систем управления и робототехники

Лабораторная работа № 1
"Задачи 1005, 1155 и 2025"

по дисциплине Алгоритмы и структуры данных

Выполнила: студентка гр. **R3238**
поток **2.1**

Нечаева А. А.

Преподаватель: *Тропченко Андрей Александрович*

Санкт-Петербург, 2024

1 Цель

Разработать и реализовать алгоритмы для решения задач 1005, 1155 и 2025.

2 Задача 1005

1005. Куча камней

Ограничение времени: 1.0 секунды

Ограничение памяти: 64 МБ

У вас есть несколько камней известного веса w_1, …, 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 ($1 \le n \le 20$) и веса камней w_1, …, w_n ($1 \le w_i \le 100\,000$) — целые, разделённые пробельными символами.

Результат

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

Пример

исходные данные	результат
5 5 8 13 27 14	3

Источник задачи: Чемпионат УрГУ 1997

Рис. 1. Условие задачи 1005.

2.1 Основная идея

Заметим, что максимальное суммарное значение веса камней $20 \times 100000 = 2000000$, что на 3 порядка меньше, чем вмещает тип *integer*, значит, мы работаем с относительно "небольшими"данными. К тому же, задача подобна *Задаче о рюкзаке* ($NP-$полная задача, большие "рюкзаки"не решаются за разумное время), значит, мы можем точно решить данную задачу используя метод ***перебора*** всех возможных вариантов распределения камней на 2 кучи.

2.2 Краткое описание алгоритма

1. Входные данные: n – количество камней ($1 \leq n \leq 20$), $w_1, \ldots, w_n$ ($1 \leq w_i \leq 100000$) – вес каждого камня.
2. Находим суммарный вес камней (*full_sum*).
3. Далее перебираем все возможные наборы. Воспользуемся свойством двоичной системы: пусть **1** == добавление камня в кучу, **0** == сумма камней в куче не меняется. Заметим, что нам достаточно перебрать только половину всех возможных наборов (остальные будут симметричными), поэтому итерации цикла будут от 0 до 2^{n-1} с шагом 1. Для определения точного номера нужного камня введена дополнительная переменная *iter_numb*, использующаяся в последовательном вычислении остатков от деления номера итерации на 2.
4. Текущий минимум разности масс куч вычисляется с помощью стандартной функции *std::min* языка *C++*.
Пусть s_1 – масса камней в первой куче, s_2 – во второй, S – сумма всех камней. Формула для модуля текущей разницы масс:

$$|s_2 - s_1| = |S - s_1 - s_1| = |S - 2 \cdot s_1|$$

5. Выходные данные: целое неотрицательное число – минимальная разница масс куч.

2.3 Структуры данных

std :: *vector* < *int* > – вектор из стандартной библиотеки *C++* – динамический массив.

2.4 Листинг

Листинг 1. Исходный код для 1005

```
#include <iostream>
#include <vector>
#include <cmath>
#include <algorithm>

int main() {
    int number;
    std::cin >> number;

    std::vector<int> stones_weights(number);
```

```
11
12      int full_sum = 0;
13
14      for (int i = 0; i < number; ++i) {
15          std::cin >> stones_weights[i];
16          full_sum += stones_weights[i];
17      }
18
19      int min = full_sum;
20
21      for (int i = 0; i < pow(2, number - 1); ++i) {
22          int cur = i;
23          int iter_numb = 0;
24          int first_sum = 0;
25          while (cur != 0) {
26              if (cur % 2 == 1) {
27                  first_sum += stones_weights[iter_numb];
28              }
29              cur = cur / 2;
30              ++iter_numb;
31          }
32          min = std::min(min, std::abs(full_sum - 2 * first_sum)
                );
33      }
34      printf("%d", min);
35
36      return 0;
37  }
```

2.5 Результат

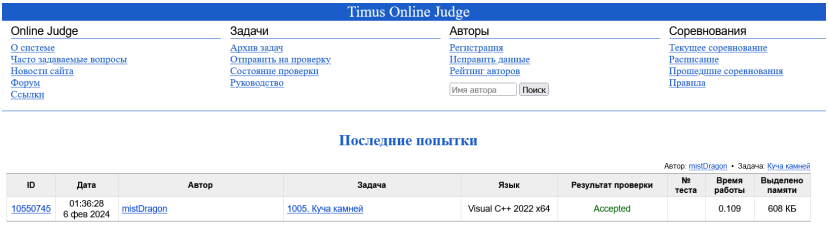

Рис. 2. Результат отправки задачи 1005.

3 Задача 2025

2025. Стенка на стенку

Ограничение времени: 1.0 секунды

Ограничение памяти: 64 МБ

Бокс, каратэ, самбо… Классические боевые единоборства пресытили аудиторию. Поэтому известный спортивный канал запускает новый формат соревнований, основанный на традиционной русской забаве — боях стенка на стенку. В соревновании могут участвовать от двух до k команд, каждая из которых будет соперничать с остальными. Всего в соревновании примут участие n бойцов. Перед началом боя они должны разделиться на команды, каждый боец должен войти ровно в одну команду. За время боя два бойца сразятся, если они состоят в разных командах. Организаторы считают, что популярность соревнований будет тем выше, чем больше будет количество схваток между бойцами. Помогите распределить бойцов по командам так, чтобы максимизировать количество схваток между бойцами, и выведите это количество.

Исходные данные

В первой строке дано количество тестов T ($1 \le T \le 10$). В следующих T строках перечислены тесты. В каждой из них записаны целые числа n и k через пробел ($2 \le k \le n \le 10^4$).

Результат

Для каждого теста в отдельной строке выведите одно целое число — ответ на задачу.

Пример

исходные данные	результат
3 6 3 5 5 4 2	12 10 4

Автор задачи: Алексей Данилюк

Источник задачи: Уральская региональная командная олимпиада по программированию 2014

Рис. 3. Условие задачи 2025.

3.1 Основная идея

Для того, чтобы максимизировать число боев, необходимо, чтобы как можно меньше бойцов находились в одной команде. То есть, в идеале в каждой команде должно быть равное число бойцов, это выполнимо только в том случае, когда количество бойцов нацело делится на число команд. В случае ненулевого остатка от деления числа бойцов на число команд, оставшихся бойцов распределяем так же равномерно. Таким образом, число бойцов в каждой команде может различаться **не больше, чем на 1**.

3.2 Краткое описание алгоритма

1. Входные данные: t – количество тестов ($1 \le t \le 10$), в следующих t строках перечислены тесты, в каждой из них 2 целые числа n и k

$(2 \leq k \leq n \leq 10^4)$.
2. Найдем среднее число бойцов в команде: для этого разделим общее количество бойцов на количество команд $medium\_in\_team = \frac{n}{k}$.
3. Найдем число бойцов пока не распределенных по командам как остаток от деления числа бойцов на количество команд: $extra\_fighters = n\%k$
4. Формула для подсчета числа боев:

$$F = medium\_in\_team \cdot (k - extra\_fighters) \cdot (n - medium\_in\_team) + \\ + extra\_fighters \cdot (medium\_in\_team + 1) \cdot (n - (medium\_in\_team + 1))$$

Заметим, что сейчас симметричные бои учтены. Поэтому конечным результатом будет: $result = \frac{F}{2}$
5. Выходные данные: для каждого из t наборов целое неотрицательное число – максимальное количество боев.

3.3 Листинг

Листинг 2. Исходный код для 2025

```
#include <iostream>


int main() {
    int numb_test;
    std::cin >> numb_test;

    for (int i = 0; i < numb_test; ++i) {

        int fighters, teams;
        std::cin >> fighters;
        std::cin >> teams;

        int medium_in_team = fighters / teams;
        int extra_fighters = fighters % teams;
        int result = (medium_in_team * (teams - extra_fighters
            ) * (fighters - medium_in_team) +
                      extra_fighters * (medium_in_team + 1) *
                      (fighters - medium_in_team - 1)) / 2;

        std::cout << result << std::endl;
    }
}
```

3.4 Результат

Рис. 4. Результат отправки задачи 2025.

4 Задача 1155

1155. Дуоны

Ограничение времени: 0.5 секунды
Ограничение памяти: 64 МБ

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.

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.

Примеры

исходные данные	результат
1 0 1 0 3 1 0 0	EF- EA- AD+ AE- DC-
0 1 0 1 2 3 2 2	IMPOSSIBLE

Источник задачи: Командный чемпионат Урала по программированию. Пермь, апрель 2001 г., английский тур.

Рис. 5. Условие задачи 1155.

4.1 Основная идея

В начале проверим условие возможности аннигиляции всех дуонов куба: суммы дуонов на несмежных вершинах должны быть равны между собой. То есть $A + C + F + H = B + E + D + G$. Основная идея состоит в том, чтобы последовательно обнулять вершины, начиная с A, далее B, D, E и G,

обнуление этих вершин повлечет за собой обнуление и C, F, H, что следует из условия выполнения проверки перед началом выполнения алгоритма.

4.2 Листинг

Листинг 3. Исходный код для 1155

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>

int cube[8];
std::vector<std::string> v_names = {"A", "B", "C", "D", "E", "
    F", "G", "H"};

void annihilate(int ind_1, int ind_2) {
    int min_numb = std::min(cube[ind_1], cube[ind_2]);

    for (int i = 0; i < min_numb; ++i) {
        std::cout << v_names[ind_1] + v_names[ind_2] + "-" <<
            std::endl;
        --cube[ind_1];
        --cube[ind_2];
    }
}

void create(int ind_1, int ind_2, int cost) {

    for (int i = 0; i < cost; ++i) {
        std::cout << v_names[ind_1] + v_names[ind_2] + "+" <<
            std::endl;
        ++cube[ind_1];
        ++cube[ind_2];
    }
}

int main() {
    int sum_of_all = 0;

    for (int i = 0; i < 8; ++i) {
        std::cin >> cube[i];
        sum_of_all += cube[i];
```

```
34      }
35
36      // checking the possibility of annihilating duons
37      if ((cube[0] + cube[2] + cube[5] + cube[7]) != (cube[1] +
            cube[3] + cube[4] + cube[6])) {
38          std::cout << "IMPOSSIBLE" << std::endl;
39      } else {
40          // take A and annihilate it using adjacent vertices
41          annihilate(0, 1);
42          annihilate(0, 3);
43          annihilate(0, 4);
44          // if after this vertex A (0) is not reset, we achieve
                one (paired with its adjacent vertex) from
                adjacent vertices to the value of A
45          if (cube[0] > 0) {
46              create(1, 2, cube[0]);
47          }
48          // here A goes to zero
49          annihilate(0, 1);
50
51          // take 2nd vertex B and annihilate with all adjacent
                vertices except zeroed A
52          annihilate(1, 2);
53          annihilate(1, 5);
54
55          if (cube[1] > 0) {
56              create(2, 6, cube[1]);
57          }
58          annihilate(1, 2);
59
60          //  D goes to zero
61          annihilate(3, 2);
62          annihilate(3, 7);
63
64          if (cube[3] > 0) {
65              create(7, 6, cube[3]);
66          }
67          annihilate(3, 7);
68
69          // E goes to zero
70          annihilate(4, 5);
71          annihilate(4, 7);
72
```

```
73          if (cube[4] > 0) {
74              create(7, 6, cube[4]);
75          }
76          annihilate(4, 7);
77
78          // G goes to zero
79          annihilate(6, 2);
80          annihilate(6, 5);
81          annihilate(6, 7);
82
83      }
84  }
```

4.3 Результат

Рис. 6. Результат отправки задачи 1155.

5 Вывод по работе

В ходе выполнения данной лабораторной работы были оеализованы алгоритмы для решения задач 1005, 2025 и 1155. Заметим, что они не содержат сложных структур данных.