Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Российский химико-технологический университет имени Д.И. Менделеева»
Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 10

Вариант 3

Выполнила студентка группы .....КС-33..... Георгиевская Анастасия Игоревна
Ссылка на репозиторий: .....

Приняли: .....Пысин Максим Дмитриевич
.....Лобанов Алексей Владимирович

Дата сдачи: 02.04.2025

Оглавление

Описание задачи.

В рамках лабораторной работы необходимо решить задачу

Динамическое программирование

У вас есть несколько камней известного веса W1, …, Wn. Напишите программу, которая распределит камни в две кучи так, что разность весов этих двух куч будет минимальной. Необходимо создать программу которая будет генерировать от 1 до 20 камней с различными весами от 1 до 10000. Итоговый набор камней должен подаваться в функцию раскладки и в ответ она должна выдавать разницу разделения общей кучи на две.

Источник задачи: https://acm.timus.ru/problem.aspx?space=1&num=1005

Для решения этой задачи необходимо воспользоваться решением задачи о рюкзаке и свести нашу задачу к решению этой, нам необходимо сложить максимальный по весу набор камней в рюкзак суммарный вес которого не может превышать S / 2, где S это суммарный вес всей кучи. Ответ на нашу задачу будет равен S - 2M, так как итоговый результат упаковывания наших камней является наилучшим, то лучше разделить камни на 2 кучи мы не сможем, а значит разница между ними, это то что по сути осталось после упаковки 2х идеальных рюкзаков.

После решения задачи требуется проверить правильность решения загрузив свое решение на проверку на сайте источнике задачи.

https://neerc.ifmo.ru/wiki/index.php?title=%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D1%80%D1%8E%D0%BA%D0%B7%D0%B0%D0%BA%D0%B5

https://proglib.io/p/python-i-dinamicheskoe-programmirovanie-na-primere-zadachi-o-ryukzake-2020-02-04

https://habr.com/ru/post/222577/

http://www.math.nsc.ru/LBRT/k5/TPR/lec4.pdf

Описание структуры задачи.

Алгоритм:

1. Нам нужно разделить камни на две кучки таким образом, чтобы разница их весов была минимальной.
2. Это можно свести к задаче о рюкзаке: нам нужно найти подмножество камней, суммарный вес которого не превышает S/2, где S — это общий вес всех камней. В этом случае разница весов между двумя кучками будет минимальной.
3. Задача сводится к нахождению максимальной суммы весов камней, которые можно упаковать в рюкзак с максимальной грузоподъемностью S/2.
4. Разница между весами двух кучек будет равна $S-2\times M$, где M — это максимально возможная сумма весов, которую мы можем упаковать в рюкзак.

Подход:

1. Сначала считаем сумму всех весов камней S.
2. Используем динамическое программирование для решения задачи о рюкзаке, чтобы найти максимальную сумму M, которую можно получить, не превышая S/2.
3. Разница в весах будет равна $S-2\times M$.

Динамическое программирование в теории управления и теории вычислительных систем — способ решения сложных задач путём разбиения их на более простые подзадачи. Он применим к задачам с оптимальной подструктурой, выглядящим как набор перекрывающихся подзадач, сложность которых чуть меньше исходной. В этом случае время вычислений, по сравнению с «наивными» методами, можно значительно сократить.

Задача о рюкзаке (англ. Knapsack problem) — дано N предметов, ni предмет имеет массу $wi>0$ и стоимость $pi>0$. Необходимо выбрать из этих предметов такой набор, чтобы суммарная масса не превосходила заданной величины W (вместимость рюкзака), а суммарная стоимость была максимальна.

Выполнение задачи.

Для выполнения задачи лабораторной был использован язык C.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAX_WEIGHT 10000

// Функция для решения задачи о рюкзаке
int pack(int* weights, int n, int max_weight) {
    int dp[max_weight + 1];
    for (int i = 0; i <= max_weight; i++) {
        dp[i] = 0;
    }

    for (int i = 0; i < n; i++) {
        for (int j = max_weight; j >= weights[i]; j--) {
            dp[j] = (dp[j] > dp[j - weights[i]] + weights[i]) ? dp[j] : dp[j
- weights[i]] + weights[i];
        }
    }

    return dp[max_weight];
}
```

- Функция pack
    - Вход: массив, содержащий веса камней, количество камней, максимальный допустимый вес рюкзака (в данном случае это половина общего веса камней).
    - Работа:
        - Внутри используется динамическое программирование, чтобы для каждого возможного веса (от 0 до max_weight) сохранить наибольшую сумму веса, которую можно достичь.
        - Итеративно обновляем массив dp (динамическое программирование), где dp[i] — это максимальная сумма веса, которую можно набрать с ограничением веса в i.
    - Выход: Возвращает максимальный вес, который можно получить с ограничением по весу max_weight.

```
int main() {
    int n;

    // Ввод количества камней и их весов
```

```
    // printf("Введите количество камней: ");
    scanf("%d", &n);

    int weights[n];
    int total_weight = 0;

    // printf("Введите веса камней: ");
    /*for (int i = 0; i < n; i++) {
        scanf("%d", &weights[i]);
        total_weight += weights[i];
    }*/

    while (scanf("%d", &weights[i]) != EOF)
    {
        total_weight += weights[i];
    }
    // Находим максимально возможный вес для одной из кучек
    int max_weight_for_one = total_weight / 2;

    // Используем рюкзак, чтобы найти наилучший вес для одной кучки
    int max_weight_in_pack = pack(weights, n, max_weight_for_one);

    // Разница в весах двух куч
    int difference = total_weight - 2 * max_weight_in_pack;

    // Выводим результат
    printf("Минимальная разница в весах двух куч: %d\n", difference);

    return 0;
}
```

Заключение.

Программа была проверена на предоставленном сайте.