ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 4
«Задача о рюкзаке»

Выполнил работу
Дышлевский Игорь
Академическая группа №J3111
Принято
Ментор, Владислав Вершинин

Санкт-Петербург
2024

1. Введение

Цель: написать программу, способную определить комбинаторным способом, какие предметы взять, чтобы максимизировать общую стоимость при заданном ограничении по весу.

Задачи:

- Написать программу
- Описать сложность и вес осноных элементов
- Построить график времени от количества данных

2. Теоретическая подготовка

Типы данных:

- Int
- Vector

Алгоритмы:

- Преобразование числа десятичной системы счисления в двоичную

3. Реализация

Главная особенность реализации - способ перебора вариантов (брать/не брать вещь). Варианты перебираются с помощью масок, полученных переводом чисел в двоичную систему заданной длины (длина маски равна количеству элементов). Такой подход гарантирует перебор всевозможных комбинаций (0/1). Получается маска, по которой можно выбирать предметы.

```
std::vector<int> dec2bin(int decimal, int len) {
    std::vector<int> binary(len, 0);
    for (int i = 0; i < len; i++) {
        binary[i] = decimal % 2;
        decimal /= 2;
    }
    return binary;
}
```

Рисунок 1 - Код для перевода числа в двоичную запись (маску)

```cpp
int find_dec_mask(std::vector<std::vector<int>> &data, int weight_limit) {
    int len = data.size();
    int max_dec = pow(2, len);

    int max_price = 0;
    int max_price_dec = 0;
    for (int i = 0; i < max_dec; i++) {
        int local_price = 0;
        int local_weight = weight_limit;
        int last_j = 0;
        std::vector<int> mask = dec2bin(i, len);
        for (int j = 0; (j < len) && (local_weight >= 0); j++) {
            local_price += data[j][1] * mask[j];
            local_weight -= data[j][0] * mask[j];
            last_j = j;
        }
        if ((local_price > max_price) && (local_weight >= 0) && (last_j == (len - 1))) {
            max_price = local_price;
            max_price_dec = i;
        }
    }
    return max_price_dec;
}
```

Рисунок 2 - Код для поиска оптимальной маски

Перебор масок от нулевой (со всеми нулями) до маски со всеми единицами с подсчетом наибольшей суммы при заданных ограничениях по весу реализован в отдельную функцию для улобства и создания нового слоя абстракции, с которым легче работать. Функция возвращает целое число (представление маски) чтобы экономить память и иметь возможность хранить много таких масок с минимальным весом. Сохраненная маска позволяет сохранить больше данных для анализа не только максимальной стоимость при ограничениях в весе, но и вещей, которые были выбраны и формируют эту стоимость.

```cpp
std::vector<int> mask = dec2bin(find_dec_mask(data, wlimit), len);
for (int i = 0; i < len; i++) {
    price += data[i][1] * mask[i];
}
```

Рисунок 3 - Код использования маски для поиска суммы

Тесты были реализованы и успешно пройдены. Они вынесены в отдельную функцию с выводом результатов тестирования (OK/Error).

```
std::vector<std::vector<int>> data = {{2, 3}, {3, 4}, {4, 5}};
int wlimit = 5;
int price = 0;
int len = data.size();
std::vector<int> mask = dec2bin(find_dec_mask(data, wlimit), len);
for (int i = 0; i < len; i++) {
    price += data[i][1] * mask[i];
}
if (price == 7) {
    std::cout << "OK" << std::endl;
} else {
    std::cout << "Error" << std::endl;
}
```

Рисунок 4 - Реализация одного из четырех тестов

4. Экспериментальная часть

Подсчёт по памяти (только для циклов и сложных структур) – как в лабораторной работе №2.

*Подсчеты приведены без учета веса самой структуры - только её содержимого

- вес(vector<vector<int>> data) = n * (2 * size_of(int)) = n * 2 * 4 = 8n Байт
- вес(vector<int> mask) = n * size_of(int) = 4n Байт

Подсчёт асимптотики (только для циклов и сложных структур) – как в лабораторной работе №3.

- O(N * 2^N) - асимптотика формируется алгоритмом перебора всех масок (O(2^N), масок 2^N штук), внутри происходит подсчет суммы по маске (O(N)).

График зависимости времени от числа элементов. Пример выполнения:

Согласно требованиям моего варианта, на вход к моему алгоритму подаётся до 25 элементов. Теоретически заданная сложность задачи составляет O(2^N) и более. Для тестирования алгоритма была собрана статистика, приведенная в таблице №1.

Таблица №1 - Подсчёт сложности реализованного алгоритма

Размер входного набора	1	5	10	15	20	25

Время выполнения программы, с	5E-06	2.1E-05	0.0009	0.023	0.55	20
O(N * 2^N), с	0.000001	0.00002	0.00061	0.019	0.56	22.2

График представляющий визуально удобный формат данных из таблицы №1 представлен на изображении №1.

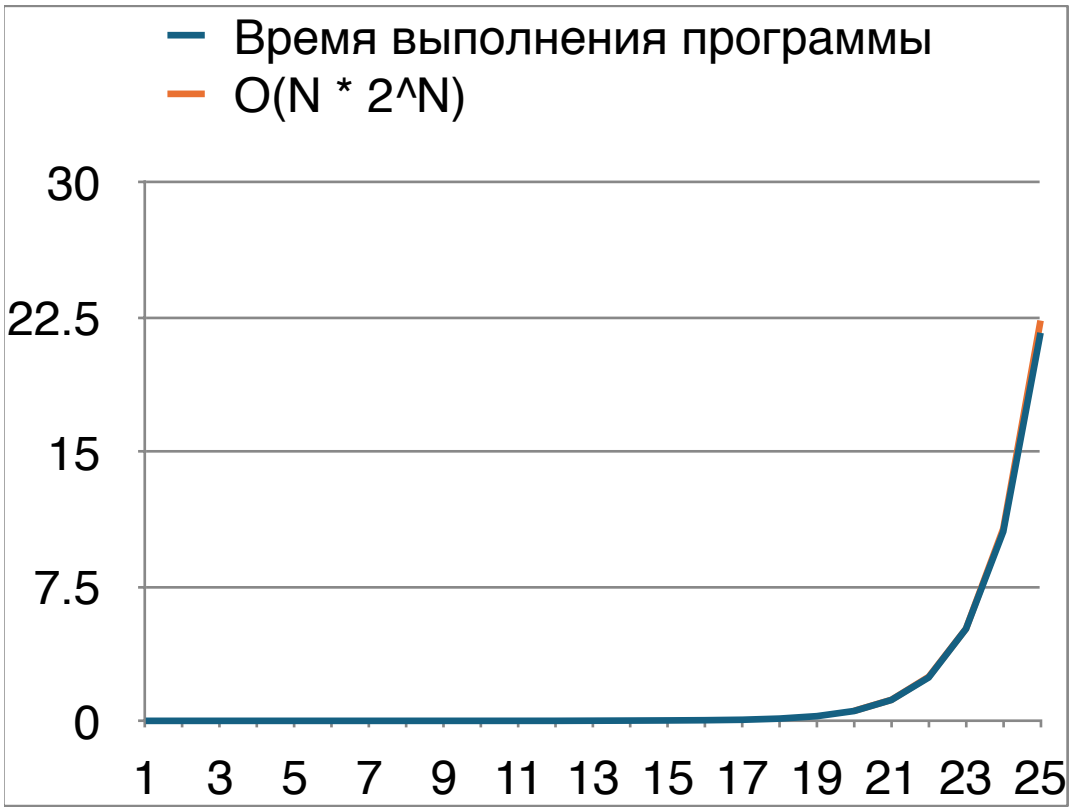

Рисунок 5 - График со временем работы

Далее необходимо привести анализ графика и таблицы.

5. Заключение

В ходе выполнения работы мною был реализована комбинаторная программа для максимизации стоимости с ограничением на вес. Цель работы была достигнута путём тестирования на разных наборах данных. Полученные результаты также совпадают с теоретическими оценками сложности алгоритма.

В качестве дальнейших исследований можно предложить оптимизацию алгоритма с точки зрения изменения самого подхода на динамическое

программирование, а также рассмотреть параллельные версии алгоритма для краткого ускорения работы.

6. Приложения

ПРИЛОЖЕНИЕ А

Листинг кода файла main4.cpp

```
#include <iostream>
#include <vector>
#include <cmath>
std::vector<int> dec2bin(int decimal, int len) {
    std::vector<int> binary(len, 0);
    for (int i = 0; i < len; i++) {
        binary[i] = decimal % 2;
        decimal /= 2;
    }
    return binary;
}
int find_dec_mask(std::vector<std::vector<int>> &data, int weight_limit) {
    int len = data.size();
    int max_dec = pow(2, len);
    int max_price = 0;
    int max_price_dec = 0;
    for (int i = 0; i < max_dec; i++) {
        int local_price = 0;
        int local_weight = weight_limit;
        int last_j = 0;
        std::vector<int> mask = dec2bin(i, len);
        for (int j = 0; (j < len) && (local_weight >= 0); j++) {
            local_price += data[j][1] * mask[j];
            local_weight -= data[j][0] * mask[j];
            last_j = j;
```

```
        }
         if ((local_price > max_price) && (local_weight >= 0) && (last_j == (len - 1)))
{
            max_price = local_price;
            max_price_dec = i;
        }
    }
    return max_price_dec;
}
void test() {
    std::vector<std::vector<int>> data = {{2, 3}, {3, 4}, {4, 5}};
    int wlimit = 5;
    int price = 0;
    int len = data.size();
    std::vector<int> mask = dec2bin(find_dec_mask(data, wlimit), len);
    for (int i = 0; i < len; i++) {
        price += data[i][1] * mask[i];
    }
    if (price == 7) {
        std::cout << "OK" << std::endl;
    } else {
        std::cout << "Error" << std::endl;
    }
    data = {{1, 35}, {16, 22}, {10, 100}, {2, 77}, {4, 24}};
    wlimit = 23;
    price = 0;
    len = data.size();
    mask = dec2bin(find_dec_mask(data, wlimit), len);
    for (int i = 0; i < len; i++) {
        price += data[i][1] * mask[i];
```

```
}
if (price == 236) {
    std::cout << "OK" << std::endl;
} else {
    std::cout << "Error" << std::endl;
}
data = {{2, 10}, {3, 15}, {5, 40}, {7, 50}, {1, 3}, {4, 20}, {1, 5}};
wlimit = 15;
price = 0;
len = data.size();
mask = dec2bin(find_dec_mask(data, wlimit), len);
for (int i = 0; i < len; i++) {
    price += data[i][1] * mask[i];
}
if (price == 105) {
    std::cout << "OK" << std::endl;
} else {
    std::cout << "Error" << std::endl;
}
data =  {{6, 30}, {3, 14}, {4, 16}, {2, 9}, {5, 20}, {1, 3}};
wlimit = 10;
price = 0;
len = data.size();
mask = dec2bin(find_dec_mask(data, wlimit), len);
for (int i = 0; i < len; i++) {
    price += data[i][1] * mask[i];
}
if (price == 47) {
    std::cout << "OK" << std::endl;
} else {
```

```
        std::cout << "Error" << std::endl;
    }

}
int main() {
    test();
    std::vector<std::vector<int>> data = {{2, 3}, {3, 4}, {4, 5}};
    int wlimit = 5;
    int price = 0;
    int len = data.size();
    std::vector<int> mask = dec2bin(find_dec_mask(data, wlimit), len);
    for (int i = 0; i < len; i++) {
        price += data[i][1] * mask[i];
        std::cout << mask[i] << " ";
    }
    std::cout << std::endl << price;
}
```