ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

# Отчёт по лабораторной работе № 4

«Задача о рюкзаке»

Выполнил работу

Сурин Тимур

Академическая группа J3113

Принято

Дунаев Максим Владимирович

Санкт-Петербург

2024

## 1. Введение

Целью данной работы является решение задачи о рюкзаке с использованием комбинаторного метода. Задача заключается в том, чтобы определить, какие предметы следует взять, чтобы максимизировать общую стоимость при заданном ограничении по весу. Для решения задачи будет применен алгоритм перебора всех возможных комбинаций предметов, что соответствует сложности O(2^N).

## 2. Теоретическая подготовка

Задача о рюкзаке — это классическая задача оптимизации, в которой нужно выбрать предметы из набора, чтобы максимизировать их стоимость, при этом их суммарный вес не должен превышать заданного лимита. Решение задачи методом полного перебора требует проверки всех возможных комбинаций предметов, что приводит к экспоненциальной сложности алгоритма O(2^N), где N — количество предметов.

Для решения задачи используется следующие типы данных:
1. std::vector<std::pair<int, int>> — для хранения предметов, каждый из которых представляет собой пару (вес, стоимость);
2. int — для подсчета максимальной стоимости и общего веса комбинации предметов;
3. std::pow — для вычисления количества возможных комбинаций предметов, равного 2^N.

## 3. Реализация

В ходе реализации задачи был использован метод полного перебора всех возможных комбинаций предметов. Для этого было использовано представление всех комбинаций в виде битовых масок. Каждый предмет в наборе представляется как пара (вес, стоимость), и для каждой комбинации вычисляется суммарный вес и стоимость. Если вес комбинации не превышает ограничение, то обновляется максимальная стоимость.

Основные этапы выполнения задачи:

1. Создание структуры данных для хранения предметов.
2. Перебор всех возможных комбинаций предметов с использованием битовых масок.
3. Проверка, не превышает ли вес текущей комбинации лимит.
4. Обновление максимальной стоимости, если текущая комбинация лучше.

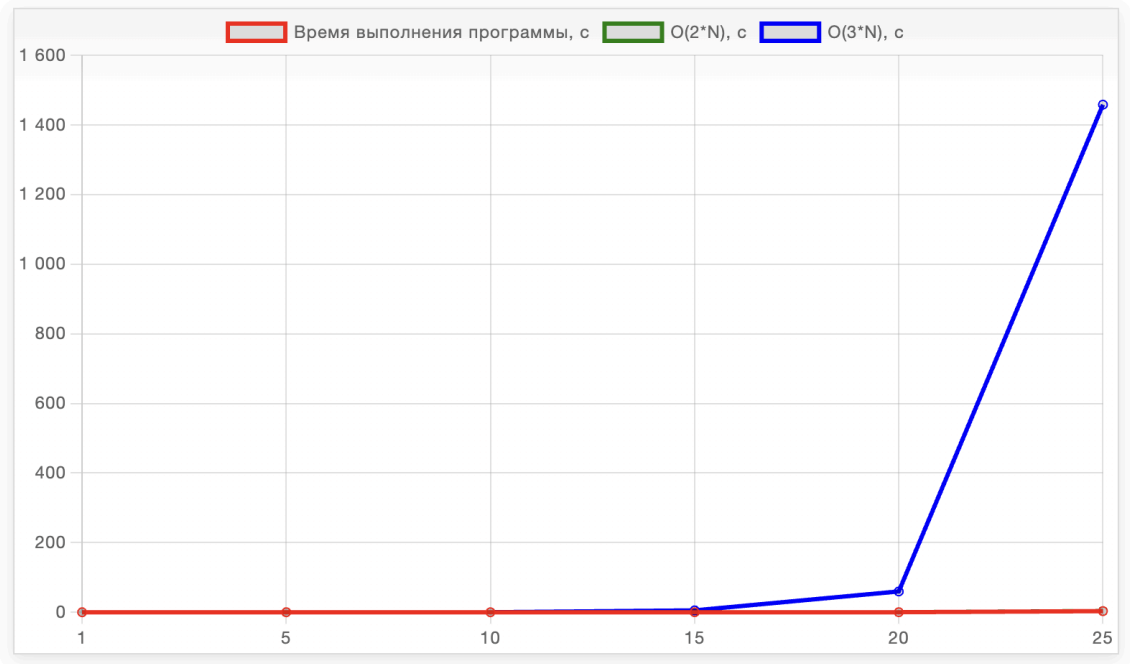
В реализации использованы библиотеки:

1. <iostream> для вывода результатов.
2. <vector> для хранения коллекции предметов.
3. <algorithm> для работы с максимальными значениями.
4. <cmath> для вычисления 2^N (общее количество комбинаций).

## 4. Экспериментальная часть

Для тестирования алгоритма была собрана статистика, приведенная в таблице ниже, для различных значений входных данных. Время выполнения зависит от количества предметов и заданного лимита по весу. Время выполнения и сложность оценивались для различных размеров входных данных.

| Размер входного набора | 1 | 5 | 10 | 15 | 20 | 25 |
|---|---|---|---|---|---|---|
| Время выполнения программы, с | 0.000000917 | 0.000002708 | 0.00010779 | 0.00619371 | 0.110321 | 3.18289 |
| O(2*N), с | 0.000002 | 0.000032 | 0.001024 | 0.0062768 | 0.11 | 3.38 |
| O(3*N), с | 0.000003 | 0.000243 | 0.039049 | 5.3489 | 59.78 | 1458 |




## 5. Заключение

В ходе выполнения работы был реализован алгоритм решения задачи о рюкзаке с использованием метода полного перебора. Цель работы была достигнута — алгоритм корректно решает задачу, выбирая оптимальные предметы для максимизации стоимости. Результаты тестирования подтверждают теоретические оценки сложности задачи.

В качестве дальнейших исследований можно предложить оптимизацию алгоритма, например, с использованием динамического программирования для уменьшения сложности, а также исследование параллельных решений для работы с большими входными данными.

## 6. Приложения

ПРИЛОЖЕНИЕ А

Листинг кода программы:

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


int f(const std::vector<std::pair<int, int>>& items, int weight_limit) {
  int n = items.size();
  int max_value = 0;
  for (int i = 0; i < std::pow(2, n); ++i) {
    int total_weight = 0;
    int total_value = 0;
    for (int j = 0; j < n; ++j) {
      if (i & (1 << j)) {
        total_weight += items[j].first;
        total_value += items[j].second;
      }
    }
```

```
        if (total_weight <= weight_limit) {

            max_value = std::max(max_value, total_value);
        }
    }
    return max_value;
}

int main() {
    std::vector<std::pair<int, int>> items = {
        {1, 2}, {2, 3}, {3, 4}, {4, 5}, {5, 6}, {6, 7}, {7, 8}, {8, 9}, {9, 10},
        {1, 2}, {3, 6}, {4, 7}, {2, 4}, {5, 9}, {6, 10}, {7, 11}, {8, 12}, {9, 13},
        {2, 5}, {3, 8}, {4, 10}, {5, 12}, {6, 14}, {7, 15}, {8, 16}, {9, 17}
    };

    int weight_limit = 100;
    int max_value = f(items, weight_limit);
```

```
    std::cout << "Максимальная стоимость: " << max_value << std::endl;



    return 0;

}
```