Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №7 по курсу «Дискретный анализ»

Студент:	М. М. Парфенов
Преподаватель:	С. А. Михайлова
Группа:	М8О-301Б-22
Дата:	
Оценка:	
Подпись:	

Москва, 2025

Лабораторная работа №7

Задача: При помощи метода динамического программирования разработать алгоритм решения задачи, определяемой своим вариантом.

1 Описание

Рассматривается задача выбора подмножества предметов из заданного множества так, чтобы удовлетворить ограничения по весу и одновременно максимизировать выражение:

$$\left(\sum_{i \in I} c_i\right) \cdot |I|,$$

где $|I|$ — мощность множества выбранных предметов, c_i — стоимость предмета, а w_i — его вес.

Алгоритм

1. Входные данные содержат: - Количество предметов n, вместимость рюкзака m. - Два массива w и c, содержащие веса и стоимости предметов соответственно.

2. Используем трёхмерный динамический массив $dp[i][j][k]$, где: - i — номер текущего предмета, - j — текущий вес, - k — количество выбранных предметов.

3. Начальные условия: - $dp[0][0][0] = 0$, остальные элементы массива заполняются значением -1.

4. Переходы: - Если предмет не включён: $dp[i+1][j][k] = \max(dp[i+1][j][k], dp[i][j][k])$. - Если предмет включён (если его добавление не превышает допустимый вес и увеличивает количество предметов):

$$dp[i+1][j+w[i]][k+1] = \max(dp[i+1][j+w[i]][k+1], dp[i][j][k] + c[i]).$$

5. После заполнения массива dp, находим максимальное значение $dp[n][j][k] \cdot k$ для всех допустимых значений j и k, сохраняя соответствующие вес и количество предметов.

6. Восстанавливаем выбранное подмножество предметов, начиная с последнего, проверяя, было ли оно включено в итоговое решение.

Сложность

- Временная сложность алгоритма: $O(n \cdot m \cdot n)$. - Пространственная сложность: $O(n \cdot m \cdot n)$.

2 Исходный код

```
#include <bits/stdc++.h>

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0), cout.tie(0);
    int n, m;
    cin >> n >> m;

    vector<int> w(n), c(n);
    for (int i = 0; i < n; i++) {
        cin >> w[i] >> c[i];
    }

    vector<vector<vector<int>>> dp(n + 1, vector<vector<int>>(m + 1, vector<int>(n + 1,
         -1)));
    dp[0][0][0] = 0;

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j <= m; j++)
        {
            for (int k = 0; k <= n; k++)
            {
                if (dp[i][j][k] == -1)
                    continue;
                dp[i + 1][j][k] = max(dp[i + 1][j][k], dp[i][j][k]);
                if (j + w[i] <= m && k + 1 <= n)
                {
                    dp[i + 1][j + w[i]][k + 1] = max(dp[i + 1][j + w[i]][k + 1], dp[i][j
                        ][k] + c[i]);
                }
            }
        }
    }

    int max_value = 0, best_weight = 0, best_count = 0;
    for (int j = 0; j <= m; j++)
    {
        for (int k = 0; k <= n; k++)
        {
            if (dp[n][j][k] != -1)
            {
                int value = dp[n][j][k] * k;
                if (value > max_value)
                {
```

```
                    max_value = value;
                    best_weight = j;
                    best_count = k;
                }
            }
        }
    }

    cout << max_value << endl;
    vector<int> result;
    int j = best_weight, k = best_count;
    for (int i = n - 1; i >= 0; i--)
    {
        if (k > 0 && j >= w[i] && dp[i][j - w[i]][k - 1] + c[i] == dp[i + 1][j][k])
        {
            result.push_back(i + 1);
            j -= w[i];
            k--;
        }
    }

    reverse(result.begin(), result.end());
    for (int x : result)
    {
        cout << x << " ";
    }
    cout << endl;

    return 0;
}
```

3 Консоль

```
 [±main]-> cat input.txt
3 6
2 1
5 4
4 2

 [±main]-> g++ main.cpp
 [±main]-> ./a.out < input.txt
 [±main]->
6
1 3
```

4 Выводы

Выполнив задачу о рюкзаке с максимизацией взвешенной стоимости, я изучил подход к решению задач с использованием динамического программирования. Этот метод позволил эффективно решать задачу, разбивая её на подзадачи и постепенно находя оптимальные решения. Кроме того, я научился находить не только максимальное значение целевой функции, но и восстанавливать оптимальное подмножество предметов, что углубило моё понимание подхода и его практического применения.

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))

[2] Список использованных источников оформлять нужно по ГОСТ Р 7.05-2008