МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт информационных технологий
Кафедра математического обеспечения и стандартизации ИТ

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2.7: Алгоритмические стратегии. Разработка и программная реализация задач с применением метода сокращения числа переборов.
ПО ДИСЦИПЛИНЕ
« СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ»

Выполнил студент группы ИКБО-01-21 Маров Г.А.

Принял старший преподаватель Туманова М.Б.

Практическая работа выполнена «20» ноября 2022 г. _____ *(подпись студента)*

«Зачтено» « » сентября 2022 г. _____ *(подпись руководителя)*

Москва 2022

Цель: Получить знания и навыки работы с стратегиями алгоритмического программирования. Разработать алгоритм и написать программу для решения поставленной в рамках варианта задачи и протестировать её на примерах. Оценить эффективность по отношению к методу «в лоб».

Ход работы

Индивидуальный вариант (4):

Задача	Метод
Имеется рюкзак с ограниченной вместимостью по массе; также имеется набор вещей с определенным весом и ценностью. Необходимо подобрать такой набор вещей, чтобы он помещался в рюкзаке и имел максимальную ценность (стоимость).	Динамическое программирование

Математическая модель решения:

Дано N предметов, W — вместимость рюкзака, w={w1,w2,…,wN} — соответствующий ему набор положительных целых весов, p={p1,p2,…,pN} — соответствующий ему набор положительных целых стоимостей.

Пусть A(k,s) есть максимальная стоимость предметов, которые можно уложить в рюкзак вместимости s, если можно использовать только первые k предметов, то есть {n1,n2,…,nk}, назовем этот набор допустимых предметов для A(k,s).

A(k,0)=0

A(0,s)=0

Найдем A(k,s). Возможны 2 варианта:

1. Если предмет k не попал в рюкзак. Тогда A(k,s) равно максимальной стоимости рюкзака с такой же вместимостью и набором допустимых предметов {n1,n2,…,nk−1}, то есть A(k,s)=A(k−1,s)
2. Если k попал в рюкзак. Тогда A(k,s) равно максимальной стоимости рюкзака, где вес s уменьшаем на вес k-ого предмета и набор допустимых предметов {n1,n2,…,nk−1} плюс стоимость k, то есть A(k−1,s−w[k])+p[k]

То есть: A(k,s)=max(A(k−1,s),A(k−1,s−w[k])+p[k])

Стоимость искомого набора равна A(N,W), так как нужно найти максимальную стоимость рюкзака, где все предметы допустимы и вместимость рюкзака W.

Восстановим набор предметов, входящих в рюкзак

Будем определять, входит ли n[i] предмет в искомый набор. Начинаем с элемента A(i,w), где i=N, w=W. Для этого сравниваем A(i,w) со следующими значениями:

1. Максимальная стоимость рюкзака с такой же вместимостью и набором допустимых предметов {n1,n2,…,ni−1}, то есть A(i−1,w)
2. Максимальная стоимость рюкзака с вместимостью на wi меньше и набором допустимых предметов {n1,n2,…,ni−1} плюс стоимость p[i], то есть A(i−1,w−w[i])+p[i]

Заметим, что при построении A мы выбирали максимум из этих значений и записывали в A(i,w). Тогда будем сравнивать A(i,w) с A(i−1,w), если равны, тогда n[i] не входит в искомый набор, иначе входит.

Сравнение переборов метода с методом «в лоб»

Сложность метода грубой силы: $O(2^N)$

Сложность метода динамического программирования: O(NW)

Код программы с комментариями:

main.cpp:

```cpp
#include <iostream>
#include <vector>
using namespace std;

vector<vector<int> > makeA(vector<int> w, vector<int> p, int n, int W) {
    vector<vector<int> > A(n + 1, vector<int>(W + 1, 0)); // first row is (0), first column is (0)
    for (int k = 1; k <= n; k++) {
        for (int s = 1; s <= W; s++) { // iterate over for each k all capacities
            if (s >= w[k]) // check if the weight of the item is less (equal) than the capacity
                A[k][s] = max(A[k-1][s], A[k-1][s-w[k]] + p[k]); // if it is, then we can either take it or not
            else
                A[k][s] = A[k-1][s]; // if it is not, then we cannot take it
        }
    }
    return A;
}
```

```
vector<int> findAns(vector<vector<int> > A, vector<int> w, int k, int s) {
    vector<int> ans;
    if (A[k][s] == 0) // if the value is 0, then we cannot take anything
        return ans;
    if (A[k-1][s] == A[k][s]) // if the value is the same as the value of the
previous row, then we cannot take the item
        ans = findAns(A, w, k-1, s);
    else {
        ans = findAns(A, w, k-1, s-w[k]); // if the value is different, then we
can take the item
        ans.push_back(k); // add the item to the answer
    }
}
int main() {
    cout << "Enter the number of items:";
    int n;
    cin >> n;
    vector<int> w(n+1), p(n+1);
    cout << "Enter the weights and profits of the items\n";
    for (int i = 1; i <= n; i++) {
        cout << "Item " << i << ":";
        cin >> w[i] >> p[i];
    }
    cout << "Enter the capacity:";
    int s;
    cin >> s;
    cout << endl;
    vector<vector<int> > A = makeA(w, p, n, s);
    for (int i = 0; i <= n; i++) {
        for (int j = 0; j <= s; j++)
            cout << A[i][j] << " ";
        cout << endl;
    }
    vector<int> ans = findAns(A, w, n, s);
    cout << "The items that can be taken are: ";
    for (int i = 0; i < ans.size(); i++)
        cout << ans[i] << " ";
    return 0;
}
```

Тестирование

Тестирование программы представлено на рисунке 1.

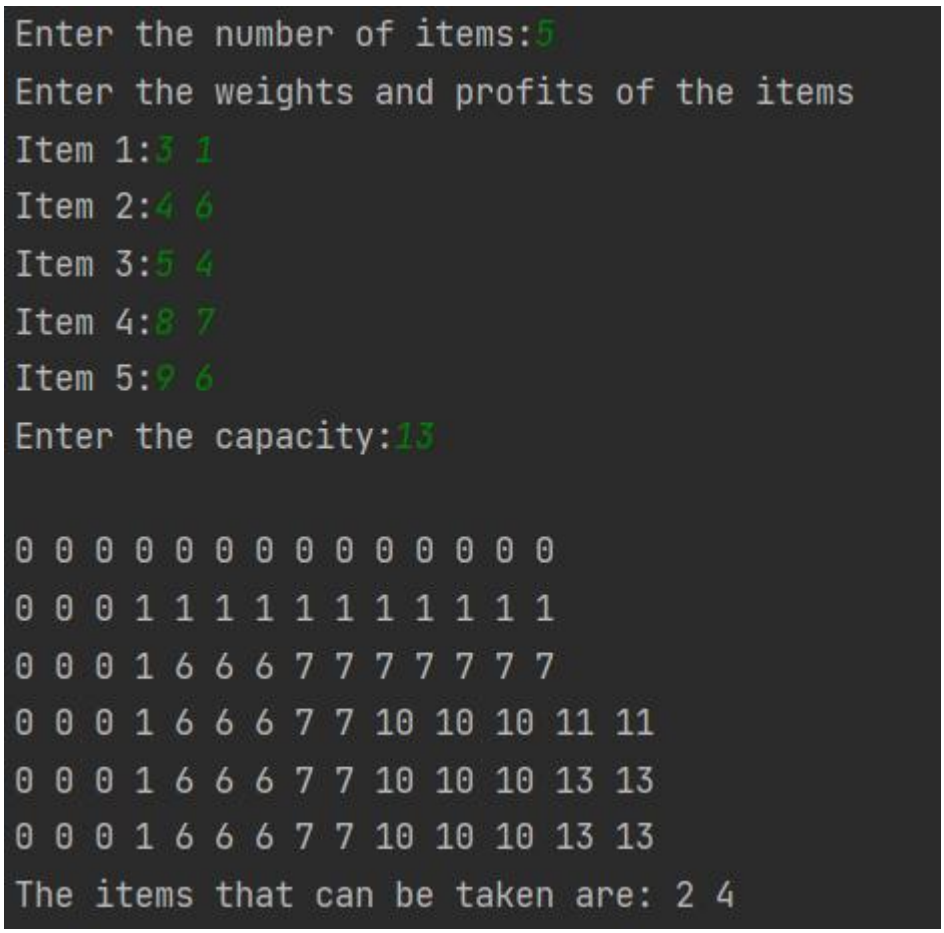

Рисунок 1 - Тестирование программы

По результатам тестирования видно, что программа работает корректно.

Вывод

В результате выполнения данной работы мной был освоен один из методов алгоритмического программирования, а именно динамическое программирование. Была написана программа для решения классической задачи о ранце, она была протестирована на примере. Проведена оценка сложности алгоритма и выполнено сравнение эффективности по отношению к алгоритму грубой силы. Таким образом, мной был получен практический опыт использования алгоритмов динамического программирования.