УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Отчёт

По дисциплине «Алгоритмы и структуры данных»  
по лабораторной работе №6  
«**Задача о рюкзаке**»

Выполнил: студент группы ПО-11  
Антонюк Н.А.  
Проверила:

Глущенко Т.А.

Брест 2024

***Задание 1.***

*Динамическим программированием* решить *классическую* *задачу о рюкзаке*. Выбор входных параметров провести самостоятельно.

*using System;*

*using System.Collections.Generic;*

*namespace KnapsackProblem*

*{*

*internal class Program*

*{*

*static void Main(string[] args)*

*{*

*int[] weights = { 3, 4, 5, 8, 9 };*

*int[] prices = { 1, 6, 4, 7, 6 };*

*int count = weights.Length;*

*int maxWeight = 13;*

*int[][] A;*

*A = new int[count + 1][];*

*for (int i = 0; i < count + 1; i++)*

*{*

*A[i] = new int[maxWeight + 1];*

*}*

*for (int k = 0; k <= count; k++)*

*{*

*for (int s = 0; s <= maxWeight; s++)*

*{*

*if (k == 0 || s == 0)*

*{*

*A[k][s] = 0;*

*}*

*else*

*{*

*if (s >= weights[k - 1])*

*{*

*A[k][s] = Math.Max(A[k - 1][s], A[k - 1][s - weights[k - 1]] + prices[k - 1]);*

*}*

*else*

*{*

*A[k][s] = A[k - 1][s];*

*}*

*}*

*}*

*}*

*List<int> result = new List<int>();*

*traceResult(A, weights, count, maxWeight, result);*

*int value = 0;*

*Console.WriteLine("Оптимальное содержимое рюкзака:");*

*for (int i = 0; i < result.Count; i++)*

*{*

*Console.WriteLine(result[i]);*

*value += prices[result[i] - 1];*

*}*

*Console.WriteLine($"Стоимость рюкзака: {value}");*

*}*

*private static void traceResult(int[][] A, int[] weight, int k, int s, List<int> result)*

*{*

*if (A[k][s] == 0)*

*{*

*return;*

*}*

*if (A[k - 1][s] == A[k][s])*

*{*

*traceResult(A, weight, k - 1, s, result);*

*}*

*else*

*{*

*traceResult(A, weight, k - 1, s - weight[k - 1], result);*

*result.Add(k);*

*}*

*}*

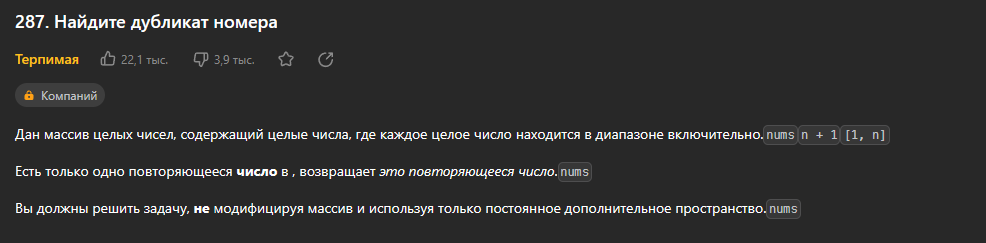
*}*

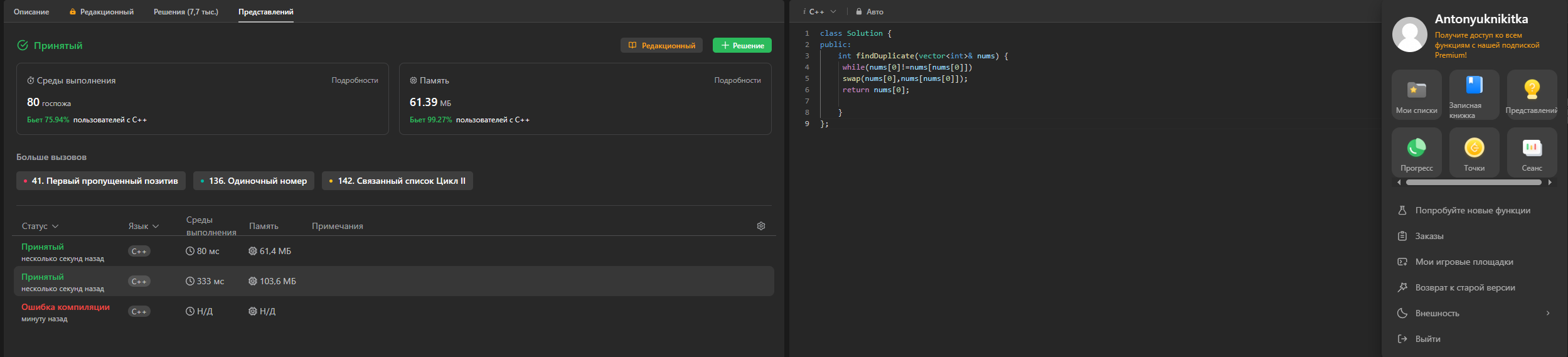
*}*

***Задача о рюкзаке*** (также задача о ранце) — [NP-полная задача](https://ru.wikipedia.org/wiki/NP-%D0%BF%D0%BE%D0%BB%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0" \o "NP-полная задача) [комбинаторной оптимизации](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F_%D0%BE%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F" \o "Комбинаторная оптимизация). Своё название получила от конечной цели: уложить как можно большее число ценных вещей в рюкзак при условии, что вместимость рюкзака ограничена.

В общем виде задачу можно сформулировать так: из заданного [множества](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE" \o "Множество) предметов со свойствами «стоимость» и «вес» требуется отобрать [подмножество](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE" \o "Подмножество) с максимальной полной стоимостью, соблюдая при этом ограничение на суммарный вес.

***Задание 2: Решение задачи на LEETCODE***





***Код задачи:***

class Solution {

public:

    int findDuplicate(vector<int>& nums) {

     while(nums[0]!=nums[nums[0]])

     swap(nums[0],nums[nums[0]]);

     return nums[0];

    }

};

***Задание 3.***

**Письменно подробно ответить на вопросы.**

1. **Какие точные методы решения данной задачи вы знаете?**

Как было сказано выше, задача о рюкзаке относится к классу [NP-полных](https://ru.wikipedia.org/wiki/NP-%D0%BF%D0%BE%D0%BB%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0" \o "NP-полная задача), и для неё пока что не найден [полиномиальный алгоритм](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BD%D0%BE%D0%BC%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC" \o "Полиномиальный алгоритм), решающий её за разумное время. Поэтому при решении задачи о рюкзаке необходимо выбирать между точными алгоритмами, которые неприменимы для «больших» рюкзаков, и приближенными, которые работают быстро, но не гарантируют оптимального решения задачи.

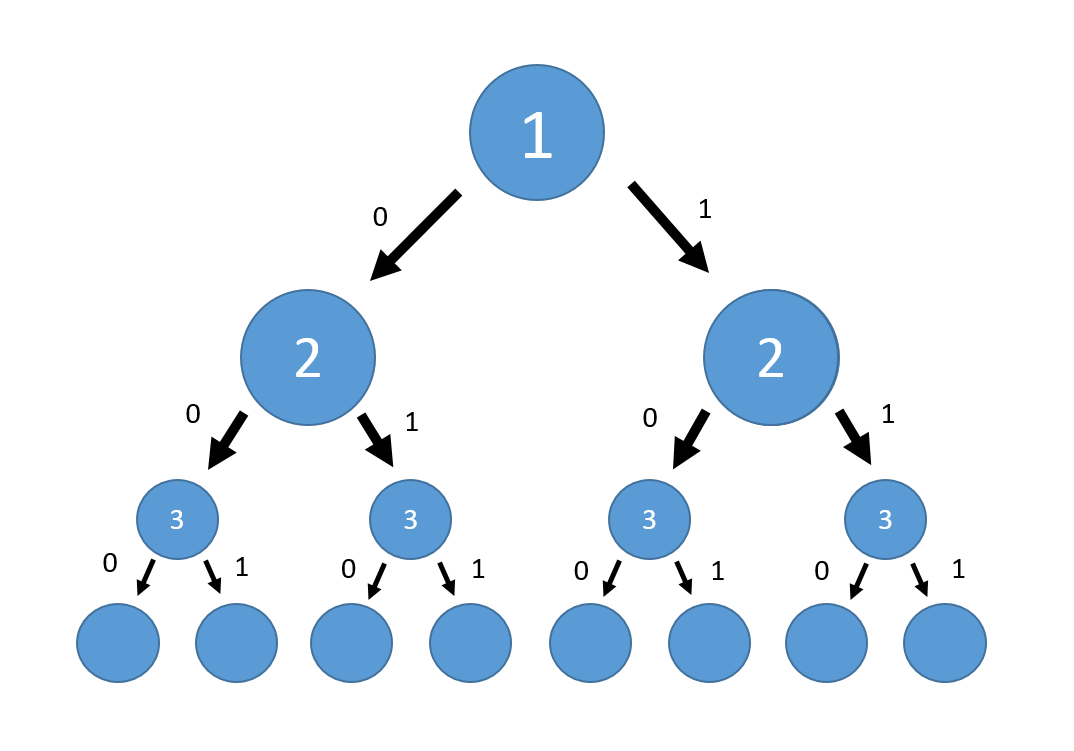
**Полный перебор**

Как и для других [дискретных задач](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5" \o "Дискретное программирование), задачу о рюкзаке можно решить, [полностью перебрав](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D0%B5%D0%B1%D0%BE%D1%80" \o "Полный перебор) все возможные решения.

По условию задачи имеется �N предметов, которые можно укладывать в рюкзак, и нужно определить максимальную стоимость груза, вес которого не превышает� W.

Для каждого предмета существует 2 варианта: предмет кладётся в рюкзак либо предмет не кладётся в рюкзак. Тогда перебор всех возможных вариантов имеет [временну́ю сложность](https://ru.wikipedia.org/wiki/%D0%92%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C" \o "Временная сложность) �(2�)O(2N), что позволяет его использовать лишь для небольшого количества предметов. С ростом числа предметов задача становится неразрешимой данным методом за приемлемое время.

На рисунке показано дерево перебора для трёх предметов. Каждый [лист](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)" \o "Дерево (теория графов)) соответствует некоторому подмножеству предметов. После составления дерева необходимо найти лист с максимальной ценностью среди тех, вес которых не превышает �W.



Дерево полного перебора, соответствующее поиску решения для трех предметов. В каждом узле определяется, будет ли данный предмет уложен в рюкзак. Цифра в узле соответствует номеру предмета. Цифры на рёбрах: 0 означает, что предмет не был взят, 1 — что был

### Метод ветвей и границ

[Метод ветвей и границ](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B2%D0%B5%D1%82%D0%B2%D0%B5%D0%B9_%D0%B8_%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%86" \o "Метод ветвей и границ) является вариацией метода полного перебора с той разницей, что исключаются заведомо неоптимальные ветви дерева полного перебора. Как и метод полного перебора, он позволяет найти оптимальное решение и поэтому относится к точным алгоритмам.

Оригинальный алгоритм, предложенный Питером Колесаром ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Английский язык) *Peter Kolesar*) в 1967 году, предлагает отсортировать предметы по их удельной стоимости (отношению ценности к весу) и строить дерево [полного перебора](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D0%B5%D0%B1%D0%BE%D1%80" \o "Полный перебор). Его улучшение заключается в том, что в процессе построения дерева для каждого узла оценивается [верхняя граница](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%B2%D0%B5%D1%80%D1%85%D0%BD%D1%8F%D1%8F_%D0%B8_%D0%BD%D0%B8%D0%B6%D0%BD%D1%8F%D1%8F_%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D1%8B" \o "Точная верхняя и нижняя границы) ценности решения, и продолжается построение дерева только для узла с максимальной оценкой. Когда максимальная верхняя граница оказывается в листе дерева, алгоритм заканчивает свою работу.

Способность метода ветвей и границ уменьшать количество вариантов перебора сильно опирается на входные данные. Его целесообразно применять только если удельные ценности предметов значительно отличаются.

### Методы динамического программирования

При дополнительном ограничении на веса предметов, задачу о рюкзаке можно решить за [псевдополиномиальное](https://ru.wikipedia.org/wiki/%D0%9F%D1%81%D0%B5%D0%B2%D0%B4%D0%BE%D0%BF%D0%BE%D0%BB%D0%B8%D0%BD%D0%BE%D0%BC%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC" \o "Псевдополиномиальный алгоритм) время методами [динамического программирования](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5" \o "Динамическое программирование).

Пусть вес каждого предмета ��wi является целым положительным числом. Тогда для решения задачи необходимо вычислить оптимальные решения для всех �∈�:0≤�≤�w ∈ ℤ : 0 ⩽ w ⩽ W , где �W — заданная грузоподъемность. Определим �[�]m[w] как максимальную ценность предметов, которые можно поместить в рюкзак грузоподъемностью �w.

Для �[�]m[w] можно записать [рекуррентные формулы](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BA%D1%83%D1%80%D1%80%D0%B5%D0%BD%D1%82%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D1%83%D0%BB%D0%B0" \o "Рекуррентная формула):

* �[0]=0m[0] = 0
* �[�]=max��≤�(��+�[�−��])m[w] = max(vi + m[w - wi]),

wi ⩽ w\

где ��,��vi, wi — ценность и вес �i-го предмета соответственно, а максимум из пустого множества следует считать равным нулю.

Фактически последнее уравнение является [функциональным уравнением Р. Беллмана](https://ru.wikipedia.org/wiki/%D0%A3%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%91%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0" \o "Уравнение Беллмана) или функциональным уравнением динамического программирования. В данном случае для его решения достаточно вычислить все значения �[�]m[w], начиная с 00 и до �W. Если дополнительно хранить на каждом шаге набор предметов, который реализует максимальную ценность, то алгоритм выдаст и оптимальный набор предметов.

Так как на каждом шаге необходимо найти максимум из �n предметов, алгоритм имеет вычислительную сложность �(��)O(nW). Поскольку �W может зависеть экспоненциально от размера входных данных, алгоритм является [псевдополиномиальным](https://ru.wikipedia.org/wiki/%D0%9F%D1%81%D0%B5%D0%B2%D0%B4%D0%BE%D0%BF%D0%BE%D0%BB%D0%B8%D0%BD%D0%BE%D0%BC%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC" \o "Псевдополиномиальный алгоритм). Поэтому эффективность данного алгоритма определяется значением �W. Алгоритм даёт отличные результаты при �≤1000W ⩽ 1000, но не очень хорошие для �≥10 000 000W ⩾ 10 000 000.

1. **Чему равна временная сложность решения данной задачи *полным перебором?***

Для каждого предмета существует 2 варианта: предмет кладётся в рюкзак либо предмет не кладётся в рюкзак. Тогда перебор всех возможных вариантов имеет [временну́ю сложность](https://ru.wikipedia.org/wiki/%D0%92%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C" \o "Временная сложность) �(2�)O(2N), что позволяет его использовать лишь для небольшого количества предметов. С ростом числа предметов задача становится неразрешимой данным методом за приемлемое время.

1. **Чему равна временная сложность решения данной задачи *динамическим программированием?***

Так как на каждом шаге необходимо найти максимум из �n предметов, алгоритм имеет вычислительную сложность �(��)O(nW). Поскольку �W может зависеть экспоненциально от размера входных данных, алгоритм является [псевдополиномиальным](https://ru.wikipedia.org/wiki/%D0%9F%D1%81%D0%B5%D0%B2%D0%B4%D0%BE%D0%BF%D0%BE%D0%BB%D0%B8%D0%BD%D0%BE%D0%BC%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC" \o "Псевдополиномиальный алгоритм). Поэтому эффективность данного алгоритма определяется значением �W. Алгоритм даёт отличные результаты при �≤1000W ⩽ 1000, но не очень хорошие для �≥10 000 000W ⩾ 10 000 000.

1. **Описать решение *задачи о рюкзаке* с приведенными ниже входными данными *жадным алгоритмом* и *методом* *динамического программирования*. Каким из методов мы находим *оптимальное решение?*****Грузоподъеность рюкзака, *W = 4*. Даны *3* предмета с весом и ценой :**

**1**

**Жадный алгоритм:** жадный алгоритм выбирает предметы с наибольшим соотношением цены к весу до тех пор, пока вместимость рюкзака не будет исчерпана. В примере с такими входными данными, соотношения цены к весу для предметов равны 30 (для w1), 25 (для w2) и 20 (для w3). Таким образом, жадный алгоритм сначала выберет предмет w1, затем w2. После этого вместимость рюкзака будет исчерпана, и общая ценность составит 80.

**Жадный алгоритм:**

1. Выбираем предмет с максимальным отношением цены к весу (pi / wi).

2. Взять предмет (1, 30).

3. Остаточная грузоподъемность: 4 - 1 = 3.

4. Выбрать следующий предмет с максимальным отношением (pi / wi).

5. Взять предмет (2, 50).

6. Итоговая стоимость: 80.

**Метод динамического программирования**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| k0 | 0 | 0 | 0 | 0 | 0 |
| k1 | 0 | 30 | 30 | 30 | 30 |
| k2 | 0 | 30 | 50 | 80 | 80 |
| k3 | 0 | 30 | 50 | 80 | 90 |

В данной таблице выписываем максимальные стоимости для всех возможных комбинаций количества предметов(k) и весов(w) используя принцип динамического программирования, т. е. считаем новые значения, используя посчитанные ранее.

**A(k,w)=max(A(k−1,w),A(k−1,w−wk)+pk)**

Общая стоимость предметов:

Общий вес:

1. **Что такое *NP-полная* задача и является ли *задача о рюкзаке* *NP-полной* задачей.**

**NP-полная задача** — в [теории алгоритмов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%BE%D0%B2" \o "Теория алгоритмов) [задача с ответом «да» или «нет»](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B5%D1%88%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8" \o "Проблема разрешимости) из [класса NP](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_NP" \o "Класс NP), к которой можно свести любую другую задачу из этого класса за [полиномиальное время](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BD%D0%BE%D0%BC%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%B2%D1%80%D0%B5%D0%BC%D1%8F" \o "Полиномиальное время) (то есть при помощи операций, число которых не превышает некоторого полинома в зависимости от размера исходных данных). Таким образом, NP-полные задачи образуют в некотором смысле подмножество «типовых» задач в классе NP: если для какой-то из них будет найден «полиномиально быстрый» алгоритм решения, то и любую другую задачу из класса NP можно будет решить так же «быстро».

**Задача о рюкзаке** (также задача о ранце) — [NP-полная задача](https://ru.wikipedia.org/wiki/NP-%D0%BF%D0%BE%D0%BB%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0" \o "NP-полная задача) [комбинаторной оптимизации](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F_%D0%BE%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F" \o "Комбинаторная оптимизация).