# МИНОБРНАУКИ РОССИИ

# Санкт-Петербургский государственный

# электротехнический университет

# «ЛЭТИ» им. В.И. Ульянова (Ленина)

# Кафедра САПР

# Курсовая работа

# по дисциплине «Алгоритмы и структуры данных»

# Тема: Задача об ограниченном рюкзаке

# Вариант 3

|  |  |  |
| --- | --- | --- |
| Студент гр. 8302 |  | Гришаков И.В. |
| Преподаватель |  | Тутуева А.В. |

Санкт-Петербург

2019

**ЗАДАНИЕ**

**НА КУРСОВУЮ РАБОТУ**

|  |
| --- |
| Студент Гришаков И.В. |
| Группа 8302 |
| Тема работы : Задача об ограниченном рюкзаке |
| Исходные данные:  количество предметов, предметы: строковое обозначения предмета name, ценность предмета value, вес предмета weight, ограничение на количество экземпляров предмета count  максимальный вес рюкзака max\_weight |
|  |
| Дата выдачи задания: 11.11.2019 |
| Дата сдачи реферата: 09.12.2019 |
| Дата защиты реферата: 09.12.2019 |

|  |  |  |
| --- | --- | --- |
| Студент гр. 8302 |  | Гришаков И.В. |
| Преподаватель |  | Тутуева А.В. |

1. Постановка задачи.

Необходимо реализовать решение задачи об ограниченном рюкзаке. Пусть имеется набор предметов, каждый из которых имеет два параметра — вес и ценность. Также имеется рюкзак определенной вместимости. Задача заключается в том, чтобы собрать рюкзак с максимальной ценностью предметов внутри, соблюдая при этом ограничение рюкзака на суммарный вес и ограничения на число экземпляров каждого предмета. Для всех функций должны быть написаны unit-тесты. Для написания unit-тестов используется библиотека Google Tests.

1. Описание выбора используемых структур данных

Для хранения вещей используется стандартный массив с фиксированным размером и статически выделенной памятью, так как необходим доступ к элементам по индексу, а в массиве это происходит за O(1). Также используются массивы, как дополнительные структуры для хранения данных в коде. Были написаны два класса:

* Item – класс вещи, содержащий поля имя, вес, ценность и доступное количество
* Backpack – класс рюкзака, при инициализации которого выполняется алгоритм сборки оптимального рюкзака.

1. Описание алгоритма решения

Сначала создаётся матрица размером КОЛИЧЕСТВО\_ВЕЩЕЙ \* МАКСИМАЛЬНЫЙ\_ВЕС х КОЛИЧЕСТВО\_ВЕЩЕЙ. Алгоритм заполнения ячейки [i][j] такой – в ячейку сохраняется значение ячейки[i-1][j]. После этого стартует цикл по количеству предмета. Если взятые предметы превышают доступный вес, то цикл останавливается. Считается ценность предмета, если бы мы его положили в рюкзак. И если эта ценность больше вычисленной ценности в [i-1][j], мы её сохраняем. После создания матрицы мы сохраняем в массив количество экземпляров каждой вещи из которых состоит наш рюкзак. Скорость алгоритма O(NW2) + вывод O(N).

1. Пример работы

int main() {

Item items\_list[] = {

{"map", 9, 150, 1},

{"compass", 13, 35, 1},

{"water", 15, 20, 20},

{"sandwich", 50, 60, 2},

{"glucose", 15, 60, 2},

{"note-case", 22, 80, 1},

{"sunglasses", 7, 20, 1},

{"towel", 18, 12, 2},

{"socks", 4, 50, 1},

{"book", 30, 10, 2},

};

Backpack back = Backpack(items\_list, 100);

back.print\_result(items\_list);

return 0;

}

Вывод:

map 1 9 150

compass 1 13 35

water 1 15 20

glucose 2 30 120

note-case 1 22 80

sunglasses 1 7 20

socks 1 4 50

Result weight: 100

1. Листинг

Листинг в IDE.