

Санкт-Петербургский Национальный Исследовательский Университет
Информационных Технологий, Механики и Оптики

Кафедра Систем Управления и Информатики

Лабораторная работа №7

Выполнил(и):

Доценников Н.А.

Проверил:

Мусаев А.А.

Санкт-Петербург,

2025

Введение

В данной лабораторной работе рассматриваются жадные алгоритмы. Будут решены задачи: подбор минимального количества монет для выдачи сдачи, задача о воре с жадным подходом, а также реализован алгоритм Дейкстры для нахождения кратчайших путей в графе на примере улиц Санкт-Петербурга. Цель работы – закрепить понимание применения жадных стратегий и их ограничений.

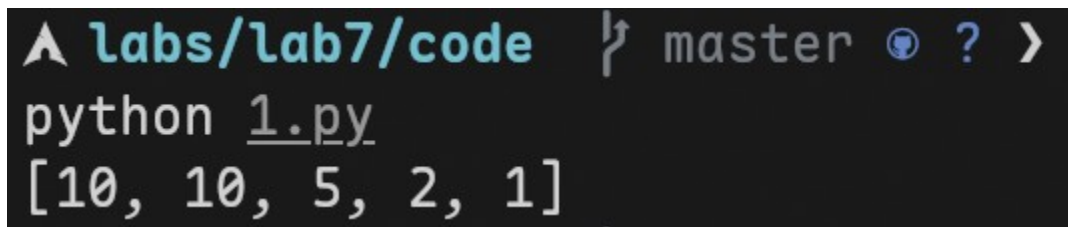
Задача 1. Задача о сдаче

Условие: Пользователю необходимо дать сдачу N рублей. У него имеется M1 монет номиналом S1, M2 монет номиналом S2 и так далее. Необходимо найти наименьшую комбинацию из заданных монет.

Решение: используем жадный подход — сортируем монеты по убыванию и выбираем самые крупные, пока это возможно. Таким образом минимизируется количество использованных монет.

Реализация (Python) для задачи 1

```
def greedy_change(n, coins):  
    result = []  
    for value, count in sorted(coins, key=lambda x: -x[0]):  
        while count > 0 and n >= value:  
            n -= value  
            count -= 1  
            result.append(value)  
    return result if n == 0 else None  
  
if __name__ == "__main__":  
    coins = [(10, 3), (5, 2), (2, 5), (1, 10)]  
    N = 28  
    print(greedy_change(N, coins))
```



```
^ labs/lab7/code master ? >  
python 1.py  
[10, 10, 5, 2, 1]
```

Рисунок 1 – Скриншот работы кода для задачи 1

Задача 2. Задача о воре (жадный алгоритм)

Условие: Вор пробрался в музей и хочет украсть N экспонатов. Каждый экспонат имеет вес и цену. Вор может сделать M заходов, каждый раз унося K кг. Определить, что должен унести вор, используя жадный алгоритм.

Решение: экспонаты сортируются по отношению цена/вес. Затем вор последовательно выбирает самые 'выгодные' экспонаты, пока не исчерпает лимит веса.

Реализация (Python) для задачи 2

```
def greedy_thief(weights, values, M, K):
    items = list(zip(weights, values))
    items.sort(key=lambda x: x[1]/x[0], reverse=True)
    cap = M * K
    total_value = 0
    chosen = []
    for w, v in items:
        if w <= cap:
            chosen.append((w, v))
            cap -= w
            total_value += v
    return total_value, chosen

if __name__ == "__main__":
    weights = [2, 3, 4, 5, 9, 7, 3]
    values = [3, 4, 5, 8, 10, 7, 6]
    M = 2
    K = 10
    print(greedy_thief(weights, values, M, K))
```

```
^ labs/lab7/code | master ? >  
python 2.py  
(26, [(3, 6), (5, 8), (2, 3), (3, 4), (4, 5)])
```

Рисунок 2 – Скриншот работы кода для задачи 2

Задача 3. Выводы

На основе решения задач 1 и 2 можно сделать вывод, что жадные алгоритмы позволяют быстро находить решения, выбирая на каждом шаге локально наилучший вариант. Однако такие решения не всегда приводят к оптимальному глобальному результату. В задаче о сдаче жадный алгоритм может не найти решение даже при его существовании. В задаче о воре жадный выбор предметов по цене/весу также может дать неоптимальный результат по сравнению с динамическим программированием.

Задача 4. Алгоритм Дейкстры (улицы Санкт-Петербурга)

Условие: Реализовать алгоритм Дейкстры для нахождения кратчайших путей. Данные: граф, вершины которого соответствуют улицам Санкт-Петербурга.

Решение: алгоритм Дейкстры работает по жадному принципу — на каждом шаге выбирается вершина с минимальным известным расстоянием, после чего это расстояние фиксируется, а значения для соседей пересчитываются.

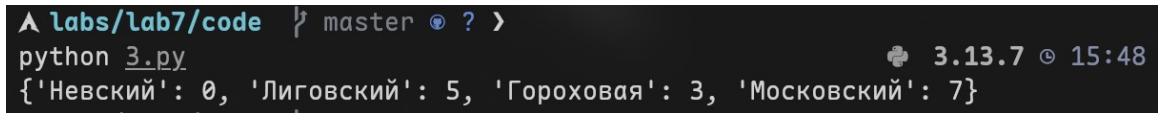
Реализация (Python) для задачи 4

```
import heapq

def dijkstra(graph, start):
    dist = {v: float('inf') for v in graph}
    dist[start] = 0
    pq = [(0, start)]
    while pq:
        d, v = heapq.heappop(pq)
        if d > dist[v]:
            continue
        for u, w in graph[v]:
            if dist[v] + w < dist[u]:
                dist[u] = dist[v] + w
                heapq.heappush(pq, (dist[u], u))
    return dist

if __name__ == "__main__":
    graph = {
        "Невский": [("Лиговский", 5), ("Гороховая", 3)],
        "Лиговский": [("Московский", 7)],
        "Гороховая": [("Московский", 4)],
        "Московский": []
    }
```

```
}  
print(dijkstra(graph, "Невский"))
```

A terminal window with a dark background. The prompt is 'labs/lab7/code' followed by a shell icon and 'master'. The command 'python 3.py' has been executed, resulting in the output: {'Невский': 0, 'Лиговский': 5, 'Гороховая': 3, 'Московский': 7}. The terminal also shows the Python version '3.13.7' and the time '15:48'.

```
labs/lab7/code master ? >  
python 3.py  
{'Невский': 0, 'Лиговский': 5, 'Гороховая': 3, 'Московский': 7}
```

Рисунок 3 – Скриншот работы кода для задачи 4

Заключение

В ходе выполнения лабораторной работы были рассмотрены жадные алгоритмы. Были решены задачи о сдаче, о воре с жадным выбором, сделаны выводы о применимости жадных стратегий, а также реализован алгоритм Дейкстры для нахождения кратчайших путей. Работа показала, что жадные алгоритмы эффективны по скорости, но могут давать неоптимальные решения в задачах, где требуется глобальная оптимизация.

Список литературы

1. Википедия. Жадный алгоритм. [Электронный ресурс] – https://ru.wikipedia.org/wiki/Жадный_алгоритм (11.05.2025);
2. Aho A.V., Hopcroft J.E., Ullman J.D. Data Structures and Algorithms. – Addison-Wesley, 1983. – 427 p. [Электронный ресурс] – <https://doc.lagout.org/Alfred%20V.%20Aho%20-%20Data%20Structures%20and%20Algorithms.pdf>. (Глава 10.3 Greedy Algorithms) (11.05.2025).