САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №7

по курсу «Алгоритмы и структуры данных»

Тема: Динамическое программирование.

Вариант 21

Выполнил:

Ступичев М. Н.

К3139

Проверил:

…

Санкт-Петербург

2024 г.

# Содержание отчета

[Содержание отчета 2](#_Toc185334561)

[Задачи по варианту 3](#_Toc185334562)

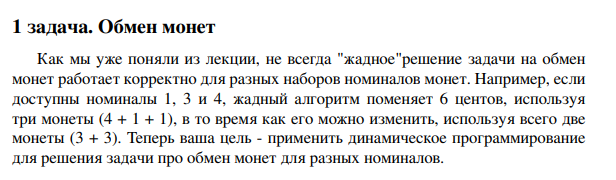
[Задача №1. Обмен монет. 3](#_Toc185334563)

[Задача №7. Шаблоны. 5](#_Toc185334564)

[Вывод 6](#_Toc185334565)

# Задачи по варианту

## Задача №1. Обмен монет.



Решение:

def dp\_change(money: int, coins: List[int]) -> int:

   min\_amount\_coins\_for\_change = [10 \*\* 9] \* (money + 1)

   for cur\_val in range(1, money + 1):

       if cur\_val in coins:

           min\_amount\_coins\_for\_change[cur\_val] = 1

       else:

           for coin in coins:

               if cur\_val >= coin:

                   min\_amount\_coins\_for\_change[cur\_val] = min(

                       min\_amount\_coins\_for\_change[cur\_val],

                       min\_amount\_coins\_for\_change[cur\_val - coin] + 1)

   ans = min\_amount\_coins\_for\_change[money]

   if ans == 10 \*\* 9:

       return -1

   return ans

Текстовое объяснение решения.

Функция dp\_change решает задачу о минимальном количестве монет, необходимых для размена заданной суммы денег, используя динамическое программирование.

Разберем функцию по частям:

1. **min\_amount\_coins\_for\_change = [10 \*\* 9] \* (money + 1)**: Создается список min\_amount\_coins\_for\_change длиной money + 1, который будет хранить минимальное количество монет, необходимое для размена каждой суммы от 0 до money. Изначально все значения инициализируются большим числом (10 \*\* 9), которое можно считать как бесконечность.
2. **for cur\_val in range(1, money + 1)**: Цикл перебирает все значения от 1 до money.
3. **if cur\_val in coins**: Если текущая сумма cur\_val соответствует номиналу какой-либо монеты, то min\_amount\_coins\_for\_change[cur\_val] устанавливается в 1 (одна монета для размена).
4. **else**: Если текущая сумма не соответствует ни одному номиналу, то:
   * **for coin in coins**: Цикл перебирает все номиналы монет.
     + **if cur\_val >= coin**: Если номинал монеты меньше или равен текущей сумме, то:
       - min\_amount\_coins\_for\_change[cur\_val] обновляется как минимум между текущим значением и значением min\_amount\_coins\_for\_change[cur\_val - coin] + 1. Это означает, что для получения суммы cur\_val можно попробовать использовать монету coin и прибавить 1 монету к оптимальному решению для суммы cur\_val - coin.
5. **if ans == 10 \*\* 9:** Если значение ans осталось равным 10 \*\* 9, то это значит, что разменять сумму money невозможно. Функция возвращает -1 в этом случае.
   * **return -1**
6. **return ans**: Если размен возможен, функция возвращает минимальное количество монет.

**Асимптотическая сложность:**

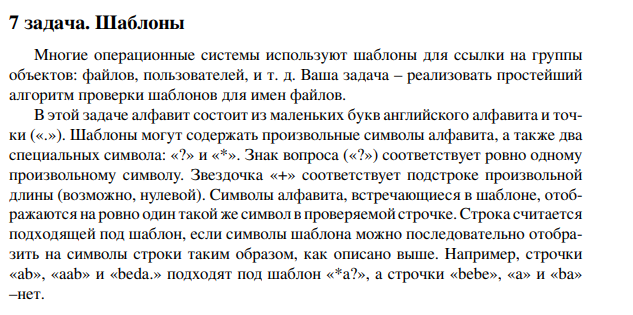
* Инициализация списка: O(money)
* Внешний цикл: O(money)
* Внутренний цикл (по номиналам монет): O(coins.size).

Общая асимптотическая сложность — **O(money \* coins.size)**. Пространственная сложность - **O(money)**.

Вывод по задаче:

Реализован динамический алгоритм нахождения наименьшего количества монет для размена.

## Задача №7. Шаблоны.



Решение:

def match\_pattern\_dp(pattern: str, string: str) -> bool:

   n = len(pattern)

   m = len(string)

​

   dp = [[False] \* (m + 1) for \_ in range(n + 1)]

   dp[0][0] = True

​

   for ind\_in\_pat in range(1, n + 1):

       if pattern[ind\_in\_pat - 1] == '\*':

           dp[ind\_in\_pat][0] = dp[ind\_in\_pat - 1][0]

​

   for ind\_in\_pat in range(1, n + 1):

       for ind\_in\_str in range(1, m + 1):

           if pattern[ind\_in\_pat - 1] == string[ind\_in\_str - 1] or pattern[ind\_in\_pat - 1] == '?':

               dp[ind\_in\_pat][ind\_in\_str] = dp[ind\_in\_pat - 1][ind\_in\_str - 1]

           elif pattern[ind\_in\_pat - 1] == '\*':

               dp[ind\_in\_pat][ind\_in\_str] = dp[ind\_in\_pat - 1][ind\_in\_str] or dp[ind\_in\_pat][ind\_in\_str - 1]

​

   return dp[n][m]

Текстовое объяснение решения.

Функция match\_pattern\_dp проверяет, соответствует ли строка string шаблону pattern, используя динамическое программирование (DP). Шаблон может содержать следующие специальные символы:

* ?: Соответствует любому одному символу.
* \*: Соответствует нулю или более символам.

Разберем функцию по частям:

1. **n = len(pattern)**, **m = len(string)**: Определяются длины шаблона и строки.
2. **dp = [[False] \* (m + 1) for \_ in range(n + 1)]**: Создается двумерный массив (таблица) dp размера (n+1) x (m+1), который будет хранить результаты вычислений. dp[i][j] будет True, если подстрока шаблона pattern[:i] соответствует подстроке строки string[:j], и False в противном случае.
3. **dp[0][0] = True**: Базовый случай: пустой шаблон соответствует пустой строке.
4. **for ind\_in\_pat in range(1, n + 1)**: Цикл для инициализации первой колонки dp.
   * **if pattern[ind\_in\_pat - 1] == '\*'**: Если текущий символ шаблона — \*, то dp[ind\_in\_pat][0] равен dp[ind\_in\_pat - 1][0]. Это означает, что \* может соответствовать пустой подстроке.
5. **for ind\_in\_pat in range(1, n + 1)**, **for ind\_in\_str in range(1, m + 1)**: Двойной цикл для заполнения основной части таблицы dp:
   * **if pattern[ind\_in\_pat - 1] == string[ind\_in\_str - 1] or pattern[ind\_in\_pat - 1] == '?'**: Если текущие символы шаблона и строки совпадают или символ шаблона — ?, то dp[ind\_in\_pat][ind\_in\_str] равен dp[ind\_in\_pat - 1][ind\_in\_str - 1]. Это означает, что если подстроки до текущего символа совпадали, то и с текущими символами они совпадают.
   * **elif pattern[ind\_in\_pat - 1] == '\*'**: Если текущий символ шаблона — \*, то dp[ind\_in\_pat][ind\_in\_str] равен dp[ind\_in\_pat - 1][ind\_in\_str] or dp[ind\_in\_pat][ind\_in\_str - 1]. Это означает, что \* может соответствовать либо пустой подстроке (dp[ind\_in\_pat - 1][ind\_in\_str]), либо одному или нескольким символам строки (dp[ind\_in\_pat][ind\_in\_str - 1]).
6. **return dp[n][m]**: Возвращает значение dp[n][m], которое указывает, соответствует ли весь шаблон всей строке.

**Асимптотическая сложность:**

* Двойной цикл для заполнения dp: O(n\*m), где n — длина шаблона, m — длина строки.
* Остальные операции: O(1).

Следовательно, общая асимптотическая сложность — **O(n\*m)**. Пространственная сложность - **O(n\*m)** из-за хранения таблицы dp.

Вывод по задаче:

Реализован динамический алгоритм нахождения соответствия между строкой и паттерном.

# Вывод

Решены 1 и 7 задачи из 21 варианта.