

Лабораторная работа 7 - Динамическое программирование

Цель

Разобрать базовые приёмы динамического программирования и применить их в задачах.

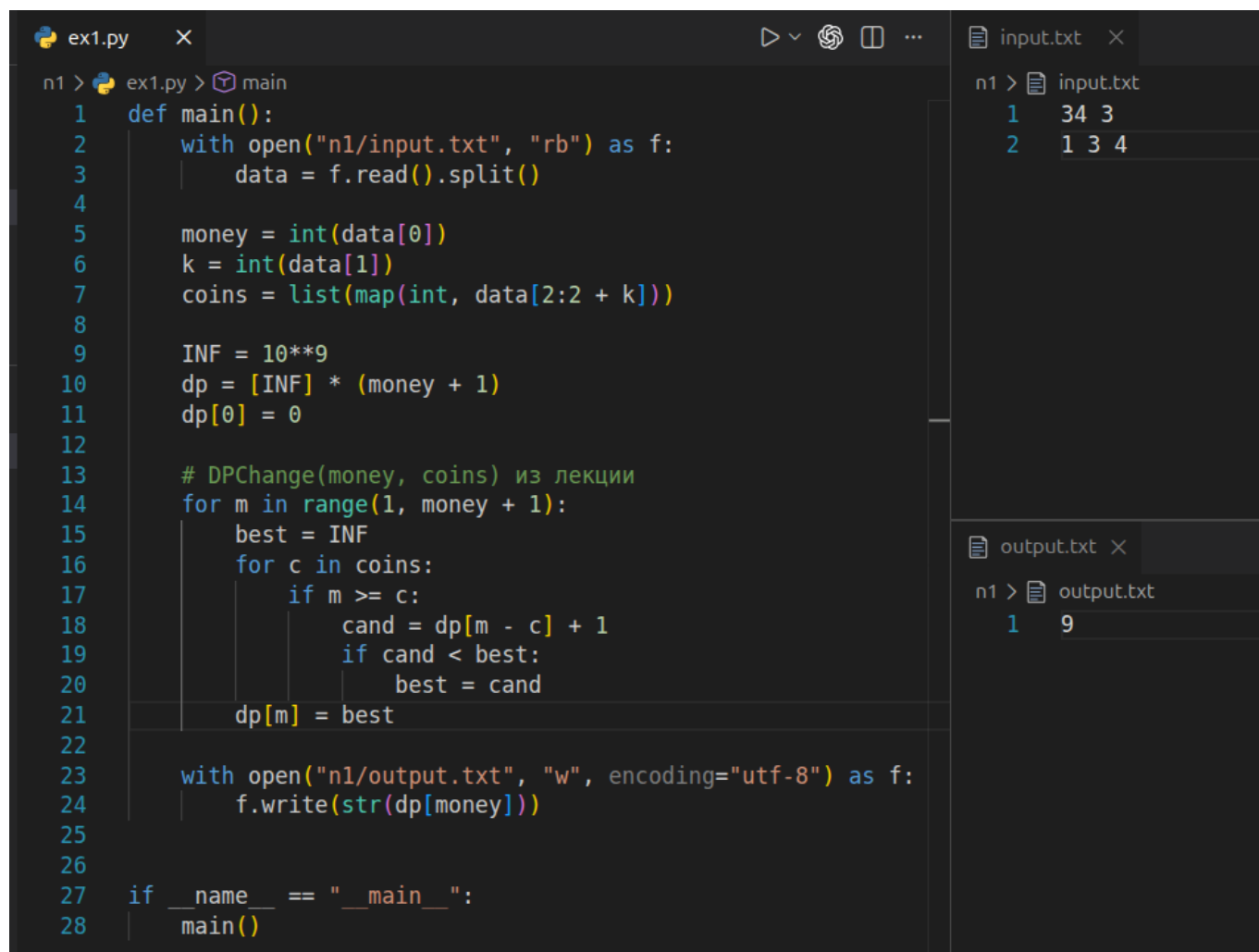
Задания и решения

Задание 1 - Обмен монет (DPChange)

Нужно найти минимальное количество монет, чтобы набрать сумму **money**.

Сделано:

- **dp[m]** - минимум монет для суммы **m**
- переход: **dp[m] = min(dp[m - coin] + 1)** по всем **coin**
- заполняем **dp** снизу вверх от 0 до **money**



```
ex1.py  X  [Icons] ...
n1 > ex1.py > main
1 def main():
2     with open("n1/input.txt", "rb") as f:
3         data = f.read().split()
4
5     money = int(data[0])
6     k = int(data[1])
7     coins = list(map(int, data[2:2 + k]))
8
9     INF = 10**9
10    dp = [INF] * (money + 1)
11    dp[0] = 0
12
13    # DPChange(money, coins) из лекции
14    for m in range(1, money + 1):
15        best = INF
16        for c in coins:
17            if m >= c:
18                cand = dp[m - c] + 1
19                if cand < best:
20                    best = cand
21        dp[m] = best
22
23    with open("n1/output.txt", "w", encoding="utf-8") as f:
24        f.write(str(dp[money]))
25
26
27 if __name__ == "__main__":
28     main()

input.txt  X
n1 > input.txt
1 34 3
2 1 3 4

output.txt  X
n1 > output.txt
1 9
```

Проверено на нескольких input-файлах (разные наборы монет, маленькие и большие суммы) - работает корректно.

Задание 2 - Примитивный калькулятор

Нужно из 1 получить n с минимальным числом операций (+1, *2, *3) и вывести:

- минимальное число операций
- саму последовательность чисел

Сделано:

- $dp[i]$ - минимум операций до i
- $prev[i]$ - откуда пришли в i , чтобы восстановить путь
- для каждого i сравниваем варианты: $i-1$, $i/2$ (если делится), $i/3$ (если делится)
- по $prev[]$ восстанавливаем ответ назад от n до 1

```
from array import array

def main():
    with open("n2/input.txt", "rb") as f:
        data = f.read().split()

    n = int(data[0])

    dp = array("I", [0]) * (n + 1) # dp[i] - минимум операций до i
    prev = array("I", [0]) * (n + 1) # prev[i] - откуда пришли в i

    for i in range(2, n + 1):
        best = dp[i - 1] + 1
        p = i - 1

        if i % 2 == 0:
            cand = dp[i // 2] + 1
            if cand < best:
                best = cand
                p = i // 2

        if i % 3 == 0:
            cand = dp[i // 3] + 1
            if cand < best:
                best = cand
                p = i // 3

        dp[i] = best
        prev[i] = p

    seq = []
    cur = n
    while True:
        seq.append(cur)
        if cur == 1:
            break
        cur = prev[cur]
```

```
seq.reverse()

out = [str(len(seq) - 1), " ".join(map(str, seq))]
with open("n2/output.txt", "w", encoding="utf-8") as f:
    f.write("\n".join(out))

if __name__ == "__main__":
    main()
```

Проверено на нескольких input-файлах (маленькие **n**, большие **n**, случаи с несколькими оптимальными путями) - вывод корректный.