

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1
по курсу «Алгоритмы и структуры данных»
Тема: Жадные алгоритмы. Динамическое
программирование №2
Вариант 8

Выполнил:
Макунина А.А.
К32421

Проверила:
Артамонова В.Е.

Санкт-Петербург
2022 г.

Содержание отчета

Задачи по варианту	3
Задача №1. Максимальная стоимость добычи (0.5 балла)	3
Задача №4. Сбор подписей (0.5 балла)	6
Задача №12. Последовательность (1 балл)	8
Задача №15. Удаление скобок (2 балла)	10
Задача №17. Ход конем (2.5 балла, замена 19ой задаче)	14
Дополнительные задачи	17
Задача №13. Сувениры (1.5 балла)	17
Задача №18. Кафе (2.5 балла)	19
Задача №21. Игра в дурака (3 балла)	22
Задача №22. Симпатичные узоры (4 балла)	26

Задачи по варианту

Задача №1. Максимальная стоимость добычи (0.5 балла)

Вор находит гораздо больше добычи, чем может поместиться в его сумке. Помогите ему найти самую ценную комбинацию предметов, предполагая, что любая часть предмета добычи может быть помещена в его сумку.

Цель - реализовать алгоритм для задачи о дробном рюкзаке.

- **Формат ввода / входного файла (input.txt).** В первой строке входных данных задано целое число n - количество предметов, и W - вместимость сумки. Следующие n строк определяют значения веса и стоимости предметов. В i -ой строке содержатся целые числа p_i и w_i – стоимость и вес i -го предмета, соответственно.
- **Ограничения на входные данные.** $1 \leq n \leq 10^3$, $0 \leq W \leq 2 \cdot 10^6$, $0 \leq p_i \leq 2 \cdot 10^6$, $0 \leq w_i \leq 2 \cdot 10^6$ для всех $1 \leq i \leq n$. Все числа - целые.
- **Формат вывода / выходного файла (output.txt).** Выведите максимальное значение стоимости долей предметов, которые помещаются в сумку. Абсолютная погрешность между ответом вашей программы и оптимальным значением должно быть не более 10^{-3} . Для этого выведите свой ответ как минимум с четырьмя знаками после запятой (иначе ваш ответ, хотя и будет рассчитан правильно, может оказаться неверным из-за проблем с округлением).
- Ограничение по времени. 2 сек.

В списке data из файла мы считываем информацию о каждом предмете и в переменную weight мы записываем начальную вместимость рюкзака. Мы разделяем информацию из файла на 2 списка для каждого элемента: в списке costs - стоимость каждого товара, в списке weights – вес. Создавая новый список товаров items, в котором мы записываем вес каждого товара и его "полезность" — это цена, деленная на вес, мы сортируем полученный список по значениям "полезности". В переменной total_cost мы сохраним общую стоимость всей добычи, в переменной left_weight - оставшееся пространство в рюкзаке. Мы циклически перебираем все элементы списка items, и, если разница между оставшимся пространством и следующим элементом больше нуля, то элемент помещается полностью, если он не помещается, мы проверяем, больше ли оставшееся пространство нуля, если да, то мы помещаем часть очередного элемента, по весу равному оставшемуся пространству и приравниваем оставшееся пространство к нулю.

```

# В список data из файла считываем информацию о
каждом предмете,
# в переменную weight записываем начальную
вместимость рюкзака
f = open('input.txt').readlines()
data = [line.split() for line in f]
weight = int(data[0][1])
data.pop(0)

# Разделяем информацию из файла на 2 списка для
каждого предмета:
# в списке costs - стоимость каждого предмета, в
списке weights - вес

costs = [int(data[i][0]) for i in range(len(data))]
weights = [int(data[i][1]) for i in range(len(data))]

def knapsack(allowed_weight, costs, weights):
    # Создаем новый список items, в который
    записываем вес каждого предмета и его 'полезность' -
    цена,
    # поделенная на вес, сортируем полученный список
    по значениям "полезности"

    items = []
    for i in range(len(costs)):
        one_item = [weights[i], costs[i] /
weights[i]]
        items.append(one_item)
        items.sort(key=lambda x: x[1], reverse=True)

    # В переменной total_cost будем хранить итоговую
    стоимость всей добычи,
    # в переменной left_weight - оставшееся местов
    рюкзаке

    total_cost, weight_left = 0, allowed_weight

    # Циклом проходимся по всем элементам списка
    items, если разность оставшегося места и очередного
    предмета больше
    # нуля, значит предмет помещается целиком, если

```

не помещается, проверяем, больше ли нуля оставшееся пространство,

если да, кладем в него часть очередного предмета, по массе равную оставшемуся месту и
приравниваем оставшееся пространство нулю

```
for item in items:
    if weight_left - item[0] >= 0:
        total_cost += item[0] * item[1]
        weight_left -= item[0]
    elif weight_left > 0:
        total_cost += item[1] * weight_left
        weight_left = 0
return total_cost
```

```
result = knapsack(weight, costs, weights)
w = open('output.txt', 'w')
w.write(str(result))
w.close()
```

Результат работы кода на примере из текста задачи:

input.txt	output.txt
3 50	1 180.0
60 20	
100 50	
120 30	

input.txt	output.txt
1 10	1 166.66666666666669
500 30	

	Время выполнения, с	Затраты памяти, мб
Пример 1 из задачи	0.00124780000000000072	0.017287254333496094
Пример 2 из задачи	0.00120359999999999991	0.017393112182617188

Вывод по задаче: реализован простой жадный алгоритм.

Задача №4. Сбор подписей (0.5 балла)

Вы несете ответственность за сбор подписей всех жильцов определенного здания. Для каждого жильца вы знаете период времени, когда он или она находится дома. Вы хотите собрать все подписи, посетив здание как можно меньше раз.

Математическая модель этой задачи следующая. Вам дан набор отрезков на прямой, и ваша цель - отметить как можно меньше точек на прямой так, чтобы каждый отрезок содержал хотя бы одну отмеченную точку.

- **Постановка задачи.** Дан набор из n отрезков $[a_0, b_0], [a_1, b_1], \dots, [a_{n-1}, b_{n-1}]$ с координатами на прямой, найдите минимальное количество m точек такое, чтобы каждый отрезок содержал хотя бы одну точку. То есть найдите набор целых чисел X минимального размера такой, чтобы для любого отрезка $[a_i, b_i]$ существовала точка $x \in X$ такая, что $a_i \leq x \leq b_i$.
- **Формат ввода / входного файла (input.txt).** Первая строка входных данных содержит количество отрезков n . Каждая из следующих n строк содержит два целых числа a_i и b_i (через пробел), определяющие координаты концов i -го отрезка.
- **Ограничения на входные данные.** $1 \leq n \leq 10^2$, $0 \leq a_i, b_i \leq 10^9$ - целые для всех $1 \leq i \leq n$.
- **Формат вывода / выходного файла (output.txt).** Выведите минимальное количество m точек в первой строке и целочисленные координаты этих m точек (через пробел) во второй строке. Вывести точки можно в любом порядке. Если таких наборов точек несколько, можно вывести любой набор. (Нетрудно видеть, что всегда существует множество точек минимального размера, для которых все координаты точек - целые числа.)
- Ограничение по времени. 2 сек.

Нам нужно подсчитать интервалы, отсортировать их по конечным значениям. Присвойте конечной точке значение None, поскольку ни один из интервалов ранее не был охвачен, и при первом прохождении цикла присвоить переменной значение конца первого интервала - конечная точка, которая всегда добавляется в выходной список с новым назначением и назначается только тогда, когда координаты начала следующего отрезка больше, чем координаты этой переменной, так как в этом случае новый отрезок не будет покрыт предыдущей крайней точкой.

```
f = open('input.txt')
n = int(f.readline())
intervals = []
```

```

for i in range(n):
    interval = f.readline().split()
    intervals.append(list(map(int, interval)))
f.close()

# отсортировать все интервалы по их концам
intervals.sort(key=lambda i: i[1])

# присвоить изначально конечной точке значение None,
так как ни один из интервалов до этого не был
покрыт,
# и при первом прохождении цикла присвоить значение
конца первого интервала переменной-крайней точке,
# которая всегда добавляется в выходной список при
новом присваивании, и присваивается только тогда,
# когда координаты начала следующего отрезка больше
координаты этой переменной,
# так как в этом случае новый отрезок не будет
покрываться предыдущей крайней точкой

points, last = [], None
for r in intervals:
    if last == None or last < r[0]:
        last = r[1]
        points.append(last)

w = open('output.txt', 'w')
w.write(str(len(points)) + '\n' + str(points))
w.close()

```

Результат работы кода на примере из текста задачи:

py ×	input.txt ×	⋮	output.txt ×
3	✓	1	1
1 3		2	[3]
2 5			
3 6			

```

input.txt:
4
4 7
1 3
2 5
5 6

output.txt:
1 2
[3, 6]

```

	Время выполнения, с	Затраты памяти, мб
Пример 1 из задачи	0.0010765999999999997	0.017600059509277344
Пример 2 из задачи	0.00106320000000000003	0.017719268798828125

Вывод по задаче: жадный алгоритм по отсортировке посчитанных интервалов по их концам реализован успешно, судя по корректному выполнению примеров из задачи.

Задача №12. Последовательность (1 балл)

- **Постановка задачи.** Дана последовательность натуральных чисел a_1, a_2, \dots, a_n , и известно, что $a_i \leq i$ для любого $1 \leq i \leq n$. Требуется определить, можно ли разбить элементы последовательности на две части таким образом, что сумма элементов в каждой из частей будет равна половине суммы всех элементов последовательности.
- **Формат ввода / входного файла (input.txt).** В первой строке входного файла находится одно целое число n . Во второй строке находится n целых чисел a_1, a_2, \dots, a_n .
- **Ограничения на входные данные.** $1 \leq n \leq 40000$, $1 \leq a_i \leq i$.
- **Формат вывода / выходного файла (output.txt).** В первую строку выходного файла выведите количество элементов последовательности в любой из получившихся двух частей, а во вторую строку через пробел номера этих элементов. Если построить такое разбиение невозможно, выведите -1.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.

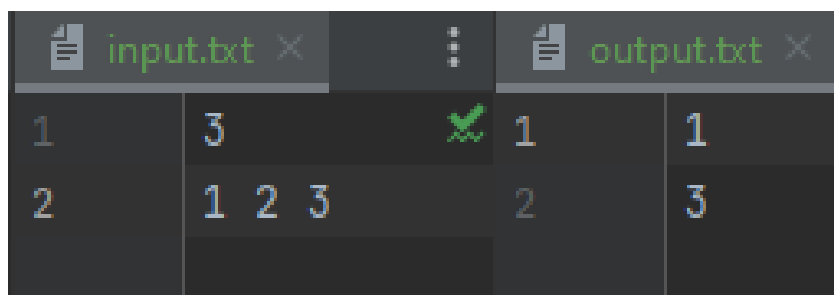
Получили на вход последовательность, проверили делимость суммы чисел на 2, отсортировали в порядке убывания. Далее в алгоритме сортируем

массивы, чтобы они подходили под условие задачи. По условию задачи член последовательности меньше либо равен своему номеру, значит, если мы можем «собрать» половину суммы чисел в первой половине множества, то можем и во второй. Проходим по каждому элементу массива и суммируем их пока сумма станет больше либо равна исходной, если нам удалось собрать сумму равную исходной – выведем последовательность, иначе – «-1».

```
def find_a_middle(x):
    if sum(x) % 2 == 0:
        s = sum(x) / 2
        x = sorted(x, reverse=True)
        a = []
        for i in range(2):
            y = []
            j = 0
            while sum(y) != s:
                if (sum(y) + x[j]) <= s:
                    y.append(x[j])
                    del (x[j])
                else:
                    j += 1
                    if j > len(x):
                        return '-1'
            a.append(y)
        return a
    else:
        return '-1'

f = open('input.txt')
ln = f.readlines()
arr = list(map(int, ln[1].split()))
ans = find_a_middle(arr)
f.close()
w = open('output.txt', 'w')
w.write(str(len(ans[0])) + "\n")
out = ''
for i in range(len(ans[0])):
    out += str(ans[0][i]) + " "
w.write(out)
w.write()
```

Результат работы кода на примере из текста задачи:



	Время выполнения, с	Затраты памяти, мб
Пример из задачи	0.0008461000000000024	0.01741313934326172

Вывод по задаче: несложная задача..

Задача №15. Удаление скобок (2 балла)

- **Постановка задачи.** Дана строка, составленная из круглых, квадратных и фигурных скобок. Определите, какое наименьшее количество символов необходимо удалить из этой строки, чтобы оставшиеся символы образовывали правильную скобочную последовательность.
- **Формат ввода / входного файла (input.txt).** Во входном файле записана строка, состоящая из s символов: круглых, квадратных и фигурных скобок $()$, $[]$, $\{\}$. Длина строки не превосходит 100 символов.
- **Ограничения на входные данные.** $1 \leq s \leq 100$.
- **Формат вывода / выходного файла (output.txt).** Выведите строку максимальной длины, являющейся правильной скобочной последовательностью, которую можно получить из исходной строки удалением некоторых символов.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.

Считываем скобочную последовательность в строку s . Двумерный массив динамики состоит из элементов $\text{smallest}[i][j]$ – наименьшего количества символов, которые надо удалить из подстроки $s[i..j]$, левый символ которой имеет позицию i , правый символ – j . Размер таблицы $n*n$, где n – длина

строки s . $BASE - smallest[i][i]=1$. Если строка состоит из одной скобки (левая граница подстроки совпадает с правой границей), то нужно удалить эту единственную скобку. Начальное заполнение двумерной таблицы динамики проходит по диагонали. Другие значения $smallest[i][j]$ заполняются нулями, так как левая граница строки не может быть больше правой границы. Рассмотрим сначала отдельно случай, когда в строке $s[l..r]$ на позициях l и r стоят соответствующие друг другу по типу открывающаяся и закрывающаяся скобки. В этом случае количество удаляемых скобок будет равно этому количеству для подстроки $s[l+1..r-1]$. Исключения составляют ситуации совокупности изначально правильных последовательностей. Чтобы вычислить $help[l][r]$ – наименьшее количество скобок, которое нужно удалить из подстроки $s[l..r]$ разобьем всевозможными способами строку $s[l..r]$ на две подстроки $s[l,k]$ и $s[k+1,r]$, где $k=l \dots r-1$. что если подстроки $s[l,k]$ и $s[k+1,r]$ сделать правильными скобочными последовательностями, удалив лишние строки в них, то и строка $s[l..r]$ станет правильной скобочной последовательностью. Поэтому остается найти минимальное значение суммарного количества удаляемых скобок для всевозможных разбиений строки на две подстроки. Используется вспомогательный массив, элементы которого $help[l][r]$ хранят индексы k , указывающие на оптимальное разбиение строки $s[l, k]$ на две подстроки $s[l, k]$ и $s[k + 1, r]$. Восстановление ответа для задачи про удаление скобок удобно реализовать при помощи рекурсивной процедуры $rec()$. Выход из рекурсии происходит в том случае, если из скобочной последовательности нужно удалить все скобки. Если из последовательности $s[l..r]$ ни одной скобки удалить не нужно, то это правильная скобочная последовательность – в этом случае печатается полностью подстрока $s[l..r]$ и осуществляется выход из рекурсии.

```
f = open("input.txt")
s = f.readline()
n = len(s)
smallest, help = [], []
for i in range(n):
    smallest.append([0] * n)
    help.append([0] * n)
for r in range(n):
    for l in range(r, -1, -1):
        if l == r:
            smallest[l][r] = 1
```

```

        else:
            best = 100
            mk = -1
            if s[l] == '(' and s[r] == ')' or (s[l]
== '[' and s[r] == ']') or (s[l] == '{'
and s[r] == '}'):
                best = smallest[l + 1][r - 1]

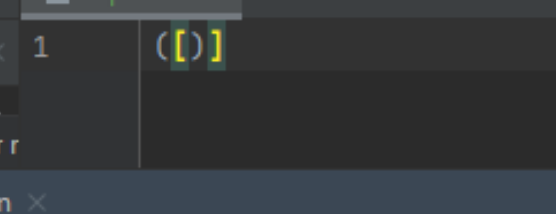
            for k in range(l, r):
                if smallest[l][k] + smallest[k +
1][r] < best:
                    best = smallest[l][k] +
smallest[k + 1][r]
                    mk = k # поиск оптимального
разбития строки
            smallest[l][r] = best
            help[l][r] = mk

# восстановление строк
def rec(l, r):
    if smallest[l][r] == r - l + 1:
        return
    elif smallest[l][r] == 0:
        print(s[l: r - l + 1])
        return
    elif help[l][r] == -1: # Если подстрока имеет в
начале и конце соответствующего типа правильные
скобки,
        print(s[l], end='') # то печатаем левую
скобку
        rec(l + 1, r - 1) # вызов рекурсию вложенной
подстроки
        print(s[r]) # печатаем правую скобку
        return
    rec(l, help[l][r]) # вызов рекурсии от левой
подстроки
    rec(help[l][r] + 1, r) # вызов рекурсии от
правой подстроки

rec(0, n - 1)

```

Результат работы кода на примере из текста задачи:



The screenshot shows a Jupyter Notebook interface. At the top, there is a tab labeled 'input.txt'. Below it, a code cell contains the following Python code:

```
>>> 1  
  
for r
```

The cursor is positioned at the end of the line 'for r'. Below the code cell, there is a 'run' button (a blue play icon) and a status bar. The status bar displays the following information:

- C:\Users\79006\AppData\Local\Program
- []
- Time: 0.0007471000000000005 s
- Max memory 0.017208099365234375 mb
- Process finished with exit code 0

[illegible]

Вывод по задаче: реализована жуткая рекурсия, это было сложно, но динамика соблюдена.

Задача №17. Ход конем (2.5 балла, замена 19ой задаче)

- **Постановка задачи.** Шахматная ассоциация решила оснастить всех своих сотрудников такими телефонными номерами, которые бы набирались на кнопочном телефоне ходом коня. Например, ходом коня набирается телефон 340-49-27. При этом телефонный номер не может начинаться ни с цифры 0, ни с цифры 8.

1	2	3
4	5	6
7	8	9
.	0	.

Напишите программу, определяющую количество телефонных номеров длины N , набираемых ходом коня. Поскольку таких номеров может быть очень много, выведите ответ по модулю 10^9 .

- **Формат ввода / входного файла (input.txt).** Во входном файле записано одно целое число N .
- **Ограничения на входные данные.** $1 \leq N \leq 1000$.
- **Формат вывода / выходного файла (output.txt).** Выведите в выходной файл искомое количество телефонных номеров по модулю 10^9 .
- Ограничение по времени. 1 сек.
- Ограничение по памяти. 256 мб.

Схема: ячейка -> перемещения.

0 -> 2 (из 4 и 6)
1 -> 2 (из 8 и 6)
2 -> 2 (из 7 и 9)
3 -> 2 (из 8 и 4)
4 -> 3 (от 3,9,0)
5 -> 0 (из ниоткуда)
6 -> 3 (из 1,7,0)
7 -> 2 (из 6 и 2)
8 -> 2 (из 1 и 3)
9 -> 2 (из 4 и 2)

Мы создаем таблицу с 10 строками в ней (каждая цифра имеет свою собственную строку) и n столбцами. Если длина числа равна 1, то вы можете получить 1 число для каждой цифры, это базовый вариант.

Подсчитать количество чисел длины n , используя количество чисел длины $n-1$, т.е. $\text{count}[n][d] = \text{count}[n-1][d*] + \text{count}[n-1][d]$, где d - цифра, с которой начинается текущее число. начинается, $d*$ — это цифры, от которых вы можете перейти к цифре d .

```
n = 1
table = [[0] * (n + 1) for i in range(10)]

for i in range(10):
    table[i][0] = 1

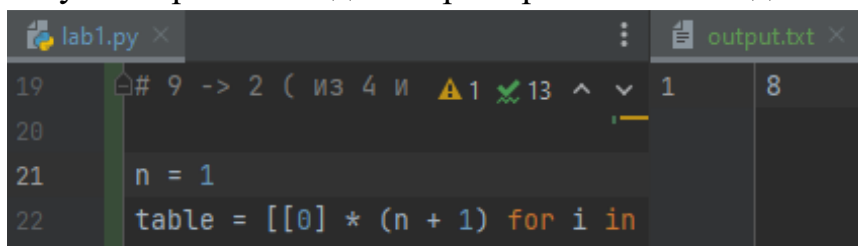
for i in range(1, n):
    table[0][i] = table[4][i - 1] + table[6][i - 1]
    table[1][i] = table[8][i - 1] + table[6][i - 1]
    table[2][i] = table[7][i - 1] + table[9][i - 1]
    table[3][i] = table[4][i - 1] + table[8][i - 1]
    table[4][i] = table[3][i - 1] + table[9][i - 1] +
table[0][i - 1]
    table[6][i] = table[1][i - 1] + table[7][i - 1] +
table[0][i - 1]
    table[7][i] = table[2][i - 1] + table[6][i - 1]
    table[8][i] = table[1][i - 1] + table[3][i - 1]
    table[9][i] = table[2][i - 1] + table[4][i - 1]

answer = 0

for i in range(10):
    answer += table[i][n - 1]
answer = answer - table[0][n - 1] - table[8][n - 1]

w = open('output.txt', 'w')
w.write(str(answer % 10 ** 9))
w.close()
```

Результат работы кода на примере из текста задачи:



```
lab1.py x  output.txt x
19  # 9 -> 2 ( из 4 и 1 13 ^ v 1 8
20
21  n = 1
22  table = [[0] * (n + 1) for i in
```

```

19 # 9
20
21 n = 2
22 table = [[0] * (n

```

Результат работы кода на максимальном значении:

```

19 # 9
20
21 n = 1000

```

753250816

	Время выполнения, с	Затраты памяти, мб
Пример из задачи	0.0031828000000000041	0.012369155883789062
Пример из задачи	0.00110210000000000086	0.009459495544433594
Верхняя граница диапазона значений входных данных из текста задачи	0.020903600000000001	1.0225439071655273

Вывод по задаче: прикольная задачка на динамику, реализация не сложная.

Дополнительные задачи

Задача №13. Сувениры (1.5 балла)

Вы и двое ваших друзей только что вернулись домой после посещения разных стран. Теперь вы хотели бы поровну разделить все сувениры, которые все трое накупили.

- **Формат ввода / входного файла (input.txt).** В первой строке дано целое число n . Во второй строке даны целые числа v_1, v_2, \dots, v_n , разделенные пробелами.
- **Ограничения на входные данные.** $1 \leq n \leq 20, 1 \leq v_i \leq 30$ для всех i .
- **Формат вывода / выходного файла (output.txt).** Выведите 1, если можно разбить v_1, v_2, \dots, v_n на три подмножества с одинаковыми суммами и 0 в противном случае.
- **Ограничение по времени.** 5 сек.

Сначала мы узнаем количество всех сувениров. Если оно не делится на 3, то нет смысла принимать дальнейшие решения. Если оно все еще делимо, то сначала нам нужно создать трехмерный массив, в который будут записаны логические значения по такому принципу: если в последовательности чисел есть подпоследовательности, суммы которых равны $s1$ и $s2$ ($s1 = s2$), то тогда True, иначе - False. Если $s1 = s2$, то тогда оставшиеся числа в последовательности образуют $s3$ ($s3 = \text{сумма} - (s1 + s2)$). В третьем внешнем цикле мы заполняем все решения в трехмерном массиве.

```
n = int(input())
u_list = list(map(int, input().split()))
t_start = time.perf_counter()
tracemalloc.start()
s = sum(u_list)

if s % 3 != 0:
    print(0)

D = [[[None for l in range(s + 1)] for j in range(s + 1)] for i in range(n + 1)]

for i in range(n + 1):
    D[i][0][0] = True
```

```

for s1 in range(s + 1):
    for s2 in range(s + 1):
        D[0][s1][s2] = (s1 == 0) and (s2 == 0)

for i in range(1, n + 1):
    for s1 in range(0, s + 1):
        for s2 in range(0, s + 1):
            D[i][s1][s2] = D[i - 1][s1][s2] or D[i - 1][s1 - u_list[i - 1]][s2] \
                            or D[i - 1][s1][s2 - u_list[i - 1]]

print(int(D[n][s // 3][s // 3]))

```

Результат работы кода на примере из текста задачи:

The first screenshot shows the file path `C:\Users\79006\AppData\Local\Programs\Python\Python39\python.exe`, input values `4` and `3 3 3 3`, and the output `0`.

The second screenshot shows the file path `C:\Users\79006\AppData\Local\Programs\Python\Python39\python.exe`, input values `1` and `40`, and the output `0`.

The third screenshot shows the file path `C:\Users\79006\AppData\Local\Programs\Python\Python39\python.exe`, input values `11` and `17 59 34 57 17 23 67 1 18 2 59`, and the output `1`.

The fourth screenshot shows the file path `C:\Users\79006\AppData\Local\Programs\Python\Python39\python.exe`, input values `13` and `1 2 3 4 5 5 7 7 8 10 12 19 25`, and the output `1`.

	Время выполнения, с	Затраты памяти, мб
Пример 1 из задачи	0.0013833999999999236	0.01220703125
Пример 2 из задачи	0.0085285000000000161	0.0376434326171875

Пример 3 из задачи	2.2486776999999996	13.265380859375
Пример 4 из задачи	0.17568680000000114	1.5860137939453125

Вывод по задаче: я написала программу, которая поможет вам и двоим вашим друзья поделить сувениры, которые вы накопили.

Задача №18. Кафе (2.5 балла)

- **Постановка задачи.** Около университета недавно открылось новое кафе, в котором действует следующая система скидок: при каждой покупке более чем на 100 рублей покупатель получает купон, дающий право на один бесплатный обед (при покупке на сумму 100 рублей и меньше такой купон покупатель не получает). Однажды вам на глаза попался преysкурant на ближайшие n дней. Внимательно его изучив, вы решили, что будете обедать в этом кафе все n дней, причем каждый день вы будете покупать в кафе ровно один обед. Однако стипендия у вас небольшая, и поэтому вы хотите по максимуму использовать предоставляемую систему скидок так, чтобы ваши суммарные затраты были минимальны. Требуется найти минимально возможную суммарную стоимость обедов и номера дней, в которые вам следует воспользоваться купонами.
- **Формат ввода / входного файла (input.txt).** В первой строке входного файла дается целое число n - количество дней. В каждой из последующих n строк дано одно неотрицательное целое число s_i – стоимость обеда в рублях на соответствующий день i .
- **Ограничения на входные данные.** $0 \leq n \leq 100$, $0 \leq s_i \leq 300$ для всех $0 \leq i \leq n$.
- **Формат вывода / выходного файла (output.txt).** В первой строке выдайте минимальную возможную суммарную стоимость обедов. Во второй строке выдайте два числа k_1 и k_2 – количество купонов, которые останутся у вас неиспользованными после этих n дней и количество использованных вами купонов соответственно. В последующих k_2 строках выдайте в возрастающем порядке номера дней, когда вам следует воспользоваться купонами. Если существует несколько решений с минимальной суммарной стоимостью, то выдайте то из них, в котором значение k_1 максимально (на случай, если вы когда-нибудь ещё решите заглянуть в это кафе). Если таких решений несколько, выведите любое из них.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 64 мб.

Используем двумерную динамику: по горизонтали количество купонов, по вертикали количество дней. Рассмотрим случаи выбора: обед < 100 , идем либо из прошлого дня (клетка сверху), либо используем купон (клетка сверху справа). Обед > 100 , идем справа сверху (используем купон) или сверху слева (получаем купон). Затем нужно восстановить решение, чтобы найти $k1$ и $k2$, используем цикл `while`, проходим путь по таблице по тем же правилам перехода, только в другую сторону, если происходит сдвиг вправо, значит купон был использован, и мы нашли нужный день.

```
def solve(arr, n):
    arr.insert(0, 0)
    table = []
    for i in range(n + 1):
        table.append([100000000] * (n + 1))
    table[0][0] = 0
    for i in range(1, n + 1):
        if arr[i] <= 100:
            for j in range(n):
                table[i][j] = min(table[i - 1][j] +
arr[i], table[i - 1][j + 1])
        else:
            for j in range(n):
                table[i][j] = min(table[i - 1][j - 1]
+ arr[i], table[i - 1][j + 1])
    res = min(table[n])
    for i in range(n):
        if res == table[n][i]:
            k1 = i

    j = k1
    i = n
    k2 = 0
    q = []
    while i != 0 or j != 0:
        if arr[i] <= 100:
            if table[i - 1][j] + arr[i] <= table[i -
1][j + 1]:
                i -= 1
            else:
                q.append(i)
                i -= 1
                j += 1
                k2 += 1
```

```

        else:
            if table[i - 1][j - 1] + arr[i] <=
table[i - 1][j + 1]:
                i -= 1
                j -= 1
            else:
                q.append(i)
                i -= 1
                j += 1
                k2 += 1
    return res, k1, k2, q

f = open('input.txt')
n = int(f.readline())
arr = []
for i in range(n):
    arr.append(int(f.readline()))
f.close()
t_start = time.perf_counter()
tracemalloc.start()
res = solve(arr, n)
w = open('output.txt', 'w')
w.write(str(res[0]) + '\n')
w.write(str(res[1]) + ' ' + str(res[2]) + '\n')
b = sorted(res[3])
for i in range(len(b)):
    w.write(str(b[i]) + '\n')
w.close()

```

Результат работы кода на примере из текста задачи:

input.txt	output.txt	input.txt	output.txt
5	1 260	3	1 220
110	2 0 2	110	2 1 1
40	3 3	110	3 2
120	4 5	110	4
110	5		
60			

	Время выполнения, с	Затраты памяти, мб
--	---------------------	--------------------

Пример 1 из задачи	0.0010941999999999896	0.009419441223144531
Пример 2 из задачи	0.0012867999999999907	0.009282112121582031

Вывод по задаче: реализован алгоритм динамического программирования с использованием двумерной таблицы.

Задача №21. Игра в дурака (3 балла)

- **Постановка задачи.** Петя очень любит программировать. Недавно он решил реализовать популярную карточную игру «Дурак». Но у Пети пока маловато опыта, ему срочно нужна Ваша помощь.

Как известно, в «Дурака» играют колодой из 36 карт. В Петиной программе каждая карта представляется в виде строки из двух символов, где первый символ означает ранг ('6', '7', '8', '9', 'T', 'J', 'Q', 'K', 'A') карты, а второй символ означает масть ('S', 'C', 'D', 'H'). Ранги перечислены в порядке возрастания старшинства.

Пете необходимо решить следующую задачу: сможет ли игрок, обладая набором из N карт, отбить M карт, которыми под него сделан ход? Для того чтобы отбиться, игроку нужно покрыть каждую из карт, которыми под него сделан ход, картой из своей колоды. Карту можно покрыть либо старшей картой той же масти, либо картой козырной масти. Если кроющаяся карта сама является козырной, то её можно покрыть только старшим козырем. Одной картой можно покрыть только одну карту.

- **Формат входного файла (input.txt).** В первой строке входного файла находятся два натуральных числа N и M , а также символ R , означающий козырную масть. Во второй строке перечислены N карт, находящихся на руках у игрока. В третьей строке перечислены M карт, которые необходимо отбить. Все карты отделены друг от друга одним пробелом.
- **Ограничения на входные данные.** $N \leq 35$, $M \leq 4$, $M \leq N$.
- **Формат выходного файла (output.txt).** В выходной файл выведите «YES» в случае, если отбиться можно, либо «NO», если нельзя.
- Ограничение по времени. 1 сек.
- Ограничение по памяти. 16 мб.

Решение таково: перебираем свои карты от самых младших, до самых старших и ищем среди них подходящую, если нашли, то «бьемся» ей, нет – выводим ответ NO. Данный алгоритм можно назвать «жадный наоборот»,

потому что мы каждый раз берем наименьшую по рангу карту из ВОЗМОЖНЫХ.

```
def get_proper_array(arr):
    dict = {
        'S': [],
        'C': [],
        'D': [],
        'H': []
    }
    for card in arr:
        number = convert_priority_to_num(card[0])
        suit = card[1]
        dict[suit].append(number)
    dict['S'].sort()
    dict['C'].sort()
    dict['D'].sort()
    dict['H'].sort()
    return dict

def beats(number, suit, m):
    global a
    if len(a[suit]) != 0:
        for card in a[suit]:
            if card > number:
                a[suit].remove(card)
                return True
        if suit != m and len(a[m]) != 0:
            a[m].pop(0)
            return True
        return False
    elif suit != m and len(a[m]) != 0:
        a[m].pop(0)
        return True
    return False

def solve(arr, arr2, m):
    global a
    a = get_proper_array(arr)
    for card in arr2:
        number = convert_priority_to_num(card[0])
```

```

        suit = card[1]
        if not beats(number, suit, m):
            return False
    return True

def convert_priority_to_num(p):
    try:
        p = int(p)
    except ValueError:
        if p == 'T':
            p = 10
        if p == 'J':
            p = 11
        if p == 'Q':
            p = 12
        if p == 'K':
            p = 13
        if p == 'A':
            p = 14
    return p

f = open('input.txt')
m = f.readline().split()[2]
b = f.readline().split()
c = f.readline().split()
f.close()
a = {}
res = solve(b, c, m)
if res:
    res = 'YES'
else:
    res = 'NO'
w = open('output.txt', 'w')
w.write(str(res))
w.close()

```

Результаты теста:

Тест	Результат	Время	Память
1	Accepted	0,031	354 КБ
2	Accepted	0,015	350 КБ
3	Accepted	0,031	342 КБ
4	Accepted	0,015	350 КБ
5	Accepted	0,031	354 КБ
6	Accepted	0,015	350 КБ
7	Accepted	0,015	358 КБ
8	Accepted	0,015	350 КБ
9	Accepted	0,015	354 КБ
10	Accepted	0,015	354 КБ
11	Accepted	0,015	342 КБ
12	Accepted	0,015	350 КБ
13	Accepted	0,015	338 КБ
14	Accepted	0,046	350 КБ
15	Accepted	0,015	358 КБ
16	Accepted	0,015	354 КБ
17	Accepted	0,031	350 КБ
18	Accepted	0,015	354 КБ
19	Accepted	0,031	350 КБ
20	Accepted	0,015	350 КБ
21	Accepted	0,031	350 КБ
22	Accepted	0,031	358 КБ
23	Accepted	0,046	354 КБ
24	Accepted	0,015	354 КБ
25	Accepted	0,031	354 КБ
26	Accepted	0,015	350 КБ
27	Accepted	0,031	342 КБ
28	Accepted	0,031	338 КБ
29	Accepted	0,015	354 КБ
30	Accepted	0,015	354 КБ
31	Accepted	0,015	350 КБ
32	Accepted	0,015	346 КБ
33	Accepted	0,031	350 КБ
34	Accepted	0,015	350 КБ
35	Accepted	0,031	354 КБ
36	Accepted	0,031	342 КБ
37	Accepted	0,031	346 КБ
38	Accepted	0,046	354 КБ
39	Accepted	0,046	350 КБ
40	Accepted	0,031	354 КБ
41	Accepted	0,031	350 КБ
42	Accepted	0,031	350 КБ
43	Accepted	0,031	350 КБ
44	Accepted	0,015	358 КБ
45	Accepted	0,015	350 КБ
46	Accepted	0,046	346 КБ
47	Accepted	0,031	350 КБ
48	Accepted	0,015	354 КБ
49	Accepted	0,031	358 КБ
50	Accepted	0,031	354 КБ

Вывод по задаче: нормальная задача за свои баллы, решение пишется достаточно спокойно, если чесать свою тыковку.

Задача №22. Симпатичные узоры (4 балла)

- **Постановка задачи.** Компания BrokenTiles планирует заняться выкладыванием во дворах у состоятельных клиентов узор из черных и белых плиток, каждая из которых имеет размер 1×1 метр. Известно, что дворы всех состоятельных людей имеют наиболее модную на сегодня форму прямоугольника $M \times N$ метров.

Однако при составлении финансового плана у директора этой организации появилось целых две серьезных проблемы: во первых, каждый новый клиент очевидно захочет, чтобы узор, выложенный у него во дворе, отличался от узоров всех остальных клиентов этой фирмы, а во вторых, этот узор должен быть симпатичным. Как показало исследование, узор является **симпатичным**, если в нем нигде не встречается квадрата 2×2 метра, полностью покрытого плитками одного цвета.

Для составления финансового плана директору необходимо узнать, сколько клиентов он сможет обслужить, прежде чем симпатичные узоры данного размера закончатся. Помогите ему!

- **Формат входного файла (input.txt).** В первой строке входного файла находятся два положительных целых числа, разделенные пробелом – M и N .
- **Ограничения на входные данные.** $1 \leq N \times M \leq 30$.
- **Формат выходного файла (output.txt).** Выведите в выходной файл единственное число – количество различных симпатичных узоров, которые можно выложить во дворе размера $M \times N$. Узоры, получающиеся друг из друга сдвигом, поворотом или отражением считаются различными.
- Ограничение по времени. 1 сек.
- Ограничение по памяти. 16 мб.

Из ограничения $N \times M$ мы можем сделать вывод, что сторон не будет больше 5. Для удобства давайте назовем минимальную сторону столбцом. Каждый столбец кодируется 5 битами, что означает, что он может иметь 2^5 состояний. Число, которым кодируется столбец, называется профилем (1 означает черный квадрат, 0 означает белый). Мы предполагаем, что все столбцы с номерами от 1 до $k-1$ уже заполнены правильно и столбец $k-1$ имеет профиль, равный номеру M . Мы сохраним состояния в переменной D . И сохраним количество способов чтобы раскрасить первые $k-1$ столбцы переменной A .

```

n, m = map(int, input().split())
if n > m:
    n, m = m, n
D = [[1 for _ in range(2 ** 5)] for _ in range(2 ** 5)]

def get_bit(num, bit):
    return 1 if num & (2 ** bit) else 0

for i in range(2 ** n):
    for j in range(2 ** n):
        for bit in range(n - 1):
            sum = get_bit(i, bit) + get_bit(i, bit +
1) + \
                get_bit(j, bit) + get_bit(j, bit +
1)
            if sum == 4 or sum == 0:
                D[i][j] = 0
                break
A = [[0 for _ in range(2 ** n)] for _ in range(m)]
for i in range(2 ** n):
    A[0][i] = 1
for k in range(1, m):
    for i in range(2 ** n):
        for j in range(2 ** n):
            A[k][i] += A[k - 1][j] * D[j][i]
answer = 0
for i in range(2 ** n):
    answer += A[m - 1][i]
print(answer)

```

Проверка решения:

Исходник решения №17689132 задачи №83

Вернуться к задаче

Редактировать решение

```
1 n, m = map(int, input().split())
2 if n > m:
3     n, m = m, n
4 D = [[1 for _ in range(2 ** 5)] for _ in range(2 ** 5)]
5
6
7 def get_bit(num, bit):
8     return 1 if num & (2 ** bit) else 0
9
10
11 for i in range(2 ** n):
12     for j in range(2 ** n):
13         for bit in range(n - 1):
14             sum = get_bit(i, bit) + get_bit(i, bit + 1) + get_bit(j, bit) + get_bit(j, bit + 1)
15             if sum == 4 or sum == 0:
16                 D[i][j] = 0
17                 break
18 A = [[0 for _ in range(2 ** n)] for _ in range(m)]
19 for i in range(2 ** n):
20     A[0][i] = 1
21 for k in range(1, m):
22     for i in range(2 ** n):
23         for j in range(2 ** n):
24             A[k][i] += A[k - 1][j] * D[j][i]
25 answer = 0
26 for i in range(2 ** n):
27     answer += A[m - 1][i]
28 print(answer)
```

Размер кода: 487

Тест	Результат	Время	Память
1	Accepted	0,015	354 КБ
2	Accepted	0,015	358 КБ
3	Accepted	0,015	354 КБ
4	Accepted	0,015	358 КБ
5	Accepted	0,031	350 КБ
6	Accepted	0,031	350 КБ
7	Accepted	0,046	350 КБ
8	Accepted	0,015	358 КБ
9	Accepted	0,046	350 КБ
10	Accepted	0,046	350 КБ

Посылки решений:

ID	Дата	Язык	Результат	Тест	Время	Память
17689132	26.09.2022 13:53:49	Python	Accepted		0,046	358 КБ

Вывод по задаче: трудная задача, но зато 4 балла.

Вывод

Динамическое программирование всегда казалось заданием со звёздочкой, но после 27 задачи на ЕГЭ по информатике я уже ничего не боюсь. В общем, это очень интересный и красивый подход. Жадные алгоритмы — это вообще отдельная тема, они работают настолько прямолинейно, что в это трудно поверить, и поэтому страшно, что это неправильно, хотя это правильно!