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 + Динамика

Вариант 17

Выполнил:

Прокопец С. Р.

К3139

Проверила:

Артамонова В.Е.

Санкт-Петербург

2023 г.

# Содержание отчета

[Содержание отчета 2](#_Toc127984533)

[Задачи по варианту 2](#_Toc127984534)

[Задача №3. Максимальный доход от рекламы [0.5 баллов] 2](#_Toc127984535)

[Задача №4. Сбор подписей [0.5 баллов] 3](#_Toc127984536)

[Задача №10. Яблоки [1.5 баллов] 3](#_Toc127984537)

[Задача №13. Сувениры [1.5 баллов] 4](#_Toc127984538)

[Задача №18. Кафе [2.5 баллов] 6](#_Toc127984539)

[Дополнительные задачи 8](#_Toc127984540)

[Задача №1. Максимальная стоимость добычи [0.5 баллов] 8](#_Toc127984541)

[Задача №2. Заправки [0.5 баллов] 8](#_Toc127984542)

[Задача №5. Максимальное количество призов [0.5 баллов] 9](#_Toc127984543)

[Задача №6. Произведение матриц [0.5 баллов] 10](#_Toc127984544)

[Задача №7. Распечатка [0.5 баллов] 12](#_Toc127984545)

[Задача №8. Расписание лекций [1 баллов] 13](#_Toc127984546)

[Задача №9. Распечатка [1 баллов] 14](#_Toc127984547)

[Задача №11. Максимальное количество золота [1 баллов] 15](#_Toc127984548)

[Задача №12. Последовательность [1 баллов] 16](#_Toc127984549)

[Задача №14. Макс значение арифметического выражения [2 баллов] 17](#_Toc127984550)

[Задача №15. Удаление скобок [2 балла] 19](#_Toc127984551)

[Задача №16. Продавец [2 балла] 19](#_Toc127984551)

[Задача №17. Ход конем [2.5 баллов] 21](#_Toc127984552)

[Задача №19. Произведение матриц [3 баллов] 22](#_Toc127984553)

[Задача №20. Почти палиндром [3 баллов] 22](#_Toc127984553)

[Задача №21. Игра в дурака [3 баллов] 23](#_Toc127984554)

[Задача №22. Симпатичный узор [4 баллов] 25](#_Toc127984555)

[Вывод 27](#_Toc127984556)

# Задачи по варианту

## Задача №3. Максимальный доход от рекламы [0.5 баллов]

Даны две последовательности a1, a2, ..., an (ai - при- быль за клик по i-му объявлению) и b1, b2, ..., bn (bi - среднее количество кликов в день i-го слота), нужно разбить их на n пар (ai, bj ) так, чтобы сумма их произведений была максимальной.

with open('input.txt') as file:

    value\_of\_ads = int(file.readline())

    cost\_of\_ad = list(map(int, file.readline().split()))

    value\_of\_clicks = list(map(int, file.readline().split()))

    cost\_of\_ad.sort()

    value\_of\_clicks.sort()

def find\_max\_profit(value\_of\_ads, cost\_of\_ad, value\_of\_clicks):

    profit = 0

    for i in range(value\_of\_ads):

        profit += cost\_of\_ad[i] \* value\_of\_clicks[i]

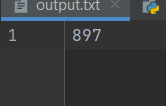
    return profit

profit = find\_max\_profit(value\_of\_ads, cost\_of\_ad, value\_of\_clicks)

f = open('output.txt', 'w')

После сортировки a-шек и b-шек мы просто их перемножаем, так как минимальные значения идут по порядку.

Изображение выглядит как текст, устройство, счетчик

Автоматически созданное описание 

## Задача №4. Сбор подписей [0.5 баллов]

Необходимо из данных отрезков выбрать такие точки, чтобы хотя бы одна из них присутствовала в этих отрезках.

with open('input.txt') as fp:

    lines = fp.readlines()

    segments = lines[1:]

def optimal\_points(segment):

    points = []

    segment = [[int(i) for i in segment.split()] for segment in segment]

    segment.sort(key = lambda x: x[1])

    while segment:

        point = segment[0][1]

        points.append(point)

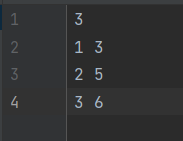
        segment = [segment for segment in segment if segment[1] < point or segment[0] > point]

    return points

f = open('output.txt', 'w')

f.write(str(len(optimal\_points(segments))) + '\n' + ' '.join([str(i) for i in optimal\_points(segments)]))

После сортировки отрезков, выбираем конечную точку первого отрезка и формируем список из условия, что следующий отрезок не пересекается с нашим.

 Изображение выглядит как текст

Автоматически созданное описание

Вывод по задаче:

## Задача №10. Яблоки [1.5 баллов]

Алиса очень голодная и хочет съесть все n яблок, но боится, что в какой-то

момент ее рост s станет равным нулю или еще меньше, и она пропадет со-

всем. Помогите ей узнать, можно ли съесть яблоки в таком порядке, чтобы в любой момент времени рост Алисы был больше нуля.

def apples(h, x):

    x = sorted(x, key=lambda apple: apple[1])

    x.reverse()

    res = []

    while len(x) > 0:

        for i in range(len(x)):

            if h - x[i][0] > 0:

                h += x[i][1]

                res.append(x[i][2])

                del x[i]

                break

            elif i == len(x) - 1:

                return -1

    return " ".join(map(str, res))

if \_\_name\_\_ == "\_\_main\_\_":

    n, s = map(int, input().split())

    x = []

    for i in range(n):

        x.append(list(map(int, input().split())))

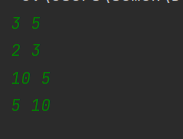
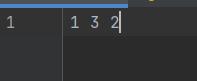
        x[i][1] = x[i][1] - x[i][0]

        x[i].append(i + 1)

    f = open('output10.txt', 'w')

    f.write(str((apples(s, x))))

Добавляем в массив прибавочный рост после съедения яблок и сортируем по этому значению. При этом сохраняем индексы. Далее проходимся по массиву пока в нем есть элементы. Смотрим не становится ли рост равным или меньшим нулю. Иначе съедаем яблоко. Если никакое яблоко не съедено, то выходим из цикла с -1.

## Задача №13. Сувениры [1.5 баллов]

Необходимо поровну разделить сувениры между 3 друзьями.

with(open('input13.txt')) as file :

    prices = list(map(int, file.readline().split()))

def we\_are\_still\_friends(prices):

    total\_cost = sum(prices)

    if len(prices) < 3 or total\_cost % 3 != 0:

        return 0

    else:

        prices.sort()

        prices = prices[-1:0:-1]

        part\_of\_each = total\_cost // 3

        for \_ in range(3):

            souvenirs\_of\_person = []

            index\_of\_souvenir = 0

            while sum(souvenirs\_of\_person) < part\_of\_each:

                if len(prices) - 1 < index\_of\_souvenir:

                    return 0

                else:

                    adding\_souvenir = prices[index\_of\_souvenir]

                    if sum(souvenirs\_of\_person) + adding\_souvenir <= part\_of\_each:

                        souvenirs\_of\_person.append(adding\_souvenir)

                        prices.pop(index\_of\_souvenir)

                    else:

                        index\_of\_souvenir += 1

        if prices is []:

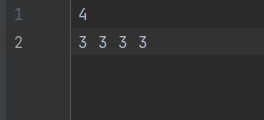
            return 1

result = we\_are\_still\_friends(prices)

f = open('output.13txt', 'w')

f.write(str(result))

Мы сортируем список сувениров по убыванию и добавляем по очереди каждому сувениры, пока их стоимость не будет равна общей стоимости сувениров / 3. После того как prices обрела итоговый вид смотрим, если списки пустые - то все распределилось поровну.

 Изображение выглядит как текст

Автоматически созданное описание

Вывод по задаче: Задача позволяет динамически найти равные подмножества из сумм компонентов.

## Задача №18. Кафе [2.5 баллов]

Требуется найти минимально возможную суммарную стоимость обедов и номера дней, в которые вам следует воспользоваться купонами.

with open('input18.txt', 'r') as file:

    days = int(file.readline())

    price = []

    for i in range(days):

        price.append(int(file.readline()))

def cafe(coupons\_count, price):

    A = [[100000 for \_ in range(coupons\_count + 1)] for \_ in range(coupons\_count + 1)]

    A[0][0] = 0

    coupons = [0, 0]

    lunch = []

    for i in range(1, len(A)):

        for j in range(len(A[0]) - 1):

            if price[i - 1] <= 100:

                A[i][j] = min(A[i - 1][j] + price[i - 1], A[i - 1][j + 1])

            else:

                A[i][j] = min(A[i - 1][j - 1] + price[i - 1], A[i - 1][j + 1])

    minimum = min(A[coupons\_count])

    for i in range(coupons\_count):

        if minimum == A[coupons\_count][i]:

            coupons[0] = i

    j = coupons[0]

    i = coupons\_count

    coupons[1] = 0

    while i != 0 or j != 0:

        if price[i - 1] <= 100:

            if A[i - 1][j] + price[i - 1] <= A[i - 1][j + 1]:

                i -= 1

            else:

                lunch.append(i)

                i -= 1

                j += 1

                coupons[1] += 1

        else:

            if A[i - 1][j - 1] + price[i - 1] <= A[i - 1][j + 1]:

                i -= 1

                j -= 1

            else:

                lunch.append(i)

                i -= 1

                j += 1

                coupons[1] += 1

    return minimum, coupons, sorted(lunch)

with open('output18.txt', 'w') as f:

    if 1 <= days <= 10 \*\* 2 and min(price) >= 0 and max(price) < 300:

        answ, coupons, coupon\_days = cafe(days, price)

        f.write(str(answ) + '\n')

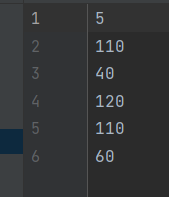
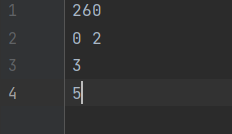
        for coupon in coupons:

            f.write(str(coupon) + ' ')

        for day in coupon\_days:

            f.write('\n' + str(day))

Добавляем в массив прибавочный рост после съедения яблок и сортируем по этому значению. При этом сохраняем индексы. Далее проходимся по массиву пока в нем есть элементы. Смотрим не становится ли рост равным или меньшим нулю. Иначе съедаем яблоко. Если никакое яблоко не съедено, то выходим из цикла с -1.

Вывод по задаче: научился Интерактивной динамики

# Дополнительные задачи

## Задача №1. Максимальная стоимость добычи [0.5 баллов]

Цель - реализовать алгоритм для задачи о дробном рюкзаке.

with open('input1.txt') as file:

    n, W = list(map(int, file.readline().split()))

    items = [list(map(int, file.readline().split())) for i in range(n)]

items.sort(key=lambda item: (item[0] / item[1]), reverse=True)

taken = 0

max\_parts = 0

for k, current in items:

    if taken + current <= W:

        max\_parts += k

        taken += current

    else:

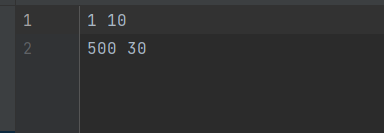
        max\_parts += (k / current) \* (W - taken)

        break

f = open('output1.txt', 'w')

f.write(str((round(max\_parts, 4))))

Считывая с файла строки, включающие вес и стоимость предметов соответственно, мы сортируем элементы по отношению веса к стоимости для нахождения наиболее выгодной части (или целого) предмета. Затем с помощью цикла высчитываем конкретные части, а в конце, когда количество целых становится больше, чем может поместиться, берем максимальное количество частей от следующего в списке предмета.

 Изображение выглядит как текст

Автоматически созданное описание

Вывод по задаче: Простая реализация жадного алгоритма позволяет компоновано и быстро решать задачи про дробный рюкзак и развивать понимание жадных алгоритмов.

## Задача №2. Заправки [0.5 баллов]

Необходимо найти минимальное количество заправочных станций, чтобы автомобиль смог доехать.

with open('input.txt') as fp:

    lines = fp.readlines()

    distance = int(lines[0])

    tank\_size = int(lines[1])

    gas\_stations = lines[3]

gas\_stations\_count = len(gas\_stations.split())

gas\_stations = [int(i) for i in gas\_stations.split()]

def car\_fueling(distance, tank\_size, gas\_stations\_count, gas\_stations):

    num\_refill, curr\_refill, limit = 0, 0, tank\_size

    while limit < distance:

        if curr\_refill >= gas\_stations\_count or gas\_stations[curr\_refill] > limit:

            return -1

        while curr\_refill < gas\_stations\_count-1 and gas\_stations[curr\_refill + 1] <= limit:

            curr\_refill += 1

        num\_refill += 1

        limit = gas\_stations[curr\_refill] + tank\_size

        curr\_refill += 1

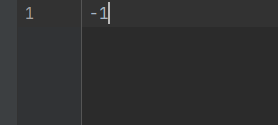
    return num\_refill

f = open('output.txt', 'w')

f.write(str(car\_fueling(distance, tank\_size, gas\_stations\_count, gas\_stations)))

После считывания данных мы заводим функцию, которая в цикле проверяет на возможность заправки автомобиля, если требуемый расход больше, чем количество км до след станции, то следственно возвращаем -1. Если же условия соблюдаются, то бы собираем в limit запас топлива и прибавляем к количеству заправок +1.

Изображение выглядит как текст

Автоматически созданное описание 

Вывод по задаче: Теперь смело можно покупать машину

## Задача №5. Максимальное количество призов [0.5 баллов]

Необходимо представить заданное натуральное число n в виде суммы как можно большего числа попарно различных натуральных чисел.

with open('input.txt') as file:

    inf\_from\_file = int(file.readline())

def find\_max\_det(number):

    for i in range(number - 1, 0, -1):

        if number % i == 0:

            return i

def happy\_kids(value\_of\_candies):

    value\_of\_pairs = find\_max\_det(value\_of\_candies)

    k = value\_of\_pairs

    candies = []

    while value\_of\_pairs != 0:

        candies.append(value\_of\_pairs)

        value\_of\_pairs -= 1

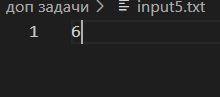
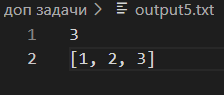
    return k, candies

k, candies = happy\_kids(inf\_from\_file)

f = open('output.txt', 'w')

f.write(str(k, candies))

Для удобства создаем функцию, которая находит наибольшее число, на которое без остатка делится данное, затем просто формируем список из этих чисел и кол-во мест.

Вывод по задаче: Дети счастливы

## Задача №6. Произведение матриц [0.5 баллов]

В качестве последнего вопроса успешного собеседования ваш начальник дет вам несколько листков бумаги с цифрами и просит составить из этих цифр наибольшее число. Полученное число будет вашей зарплатой, поэтому вы очень заинтересованы в максимизации этого числа. Как вы можете это сделать?

def compare(a, b):

    a = str(a)

    b = str(b)

    for i in range(min(len(a), len(b))):

        if a[i] < b[i]:

            return 1

        if a[i] > b[i]:

            return -1

    if len(a) < len(b):

        return -1

    if len(a) > len(b):

        return 1

    return 0

def special\_sort(array):

    if len(array) <= 1:

        return array

    state\_element = array[0]

    left = []

    right = []

    middle = []

    for i in array:

        if compare(i, state\_element) == 0:

            middle.append(i)

        if compare(i, state\_element) == 1:

            right.append(i)

        if compare(i, state\_element) == -1:

            left.append(i)

    left = special\_sort(left)

    right = special\_sort(right)

    return left + middle + right

f = open('input6.txt')

n = list(map(int, f.readline().split()))

file\_output = open('output6.txt', 'w')

file\_output.write(str(''.join(map(str, special\_sort(n)))))

Реализация: для начала создаем функцию для нахождения наибольшего числа среди двух, после создания функции мы проходясь по циклу смотрим, если число большее, то мы его добавляем в список левых элементов(больших) и также с равными и меньшими. После этого соединяем 3 списка и правильно выводим.

Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как текст

Автоматически созданное описание

## Задача №7. Распечатка [0.5 баллов]

Дано n сапог, нуждающихся в починке. Определите, какое максимальное количество из них сапожник сможет починить за один рабочий день.

with open('input.txt') as file:

    K, n = map(int, file.readline().split())

    time\_for\_each = list(map(int, file.readline().split()))

time\_spent = 0

shoes\_done = 0

time\_for\_each.sort()

while time\_spent < K and time\_for\_each:

    for item in time\_for\_each:

        if item + time\_spent <= K:

            shoes\_done += 1

            time\_spent += item

            del time\_for\_each[time\_for\_each.index(item)]

        else:

            continue

f = open('output.txt', 'w')

f.write(str(shoes\_done))

В цикле выписываем условие, при котором он будет выполняться до того момента, пока сумма времени не перевалит за границу или пока в массиве не закончатся элементы. Если время подходит под условие, то есть сапожник успеет выполнить задание – прибавляем к готовой обуви ещё одну, а к затраченному времени – время на ремонт этой пары. Затем удаляем эту пару из массива с помощью поиска по индексу.

Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как текст

Автоматически созданное описание

Вывод по задаче: Простая задача = халява

## Задача №8. Расписание лекций [1 баллов]

Необходимо выбрать из этих заявок такое подмножество, чтобы суммарно выполнить максимальное количество заявок.

with open('input.txt') as file:

    inf\_from\_file = list(file.read().splitlines())

def find\_beginnings\_and\_endings(file\_inf):

    beginnings = []

    endings = []

    for i in range(1, int(file\_inf[0]) + 1):

        begin, end = file\_inf[i].split()

        beginnings.append(int(begin))

        endings.append(int(end))

    return beginnings, endings

def find\_max\_values\_of\_lectures(beginnings, endings, total\_value\_of\_lectures):

    t, minutes\_left, counter = 0, 0, 0

    while minutes\_left < 1440:

        minutes\_left = 1440

        for i in range(total\_value\_of\_lectures):

            if beginnings[i] >= t and endings[i] < minutes\_left:

                minutes\_left = endings[i]

        if minutes\_left < 1440:

            t = minutes\_left

            counter += 1

    return counter

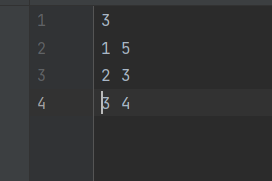
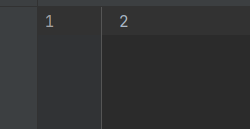
beginnings, endings = find\_beginnings\_and\_endings(inf\_from\_file)

result = find\_max\_values\_of\_lectures(beginnings, endings, int(inf\_from\_file[0]))

f = open('output.txt', 'w')

f.write(str(result))

После считывания данных мы в цикле перебираем те лекции, которые умещаются в 1440 минут, а затем если нашлась лекция, меньшая чем это число, то t(начало след лекции затем) мы меняем на значение окончания поступившей лекции.

Вывод по задаче: Опять делаем расписание вместо ИТМО.

## Задача №9. Распечатка [1 баллов]

Требуется по заданному объему заказа в листах N определить минимальную сумму денег в рублях, которой будет достаточно для выполнения заказа.

from cmath import inf

f = open('input.txt')

N = int(f.readline())

numbers = [[10\*\*j, int(f.readline())] for j in range(7)]

numbers = list(map(lambda k: [k[0], k[1], k[1] / k[0]], numbers))

numbers.sort(key=lambda amount: amount[2])

min\_in, min\_sum = inf, 0

N\_ = N

for i in numbers:

    if i[0] > N and i[1] < min\_in:

        min\_in = i[1]

        continue

    while N\_ >= i[0]:

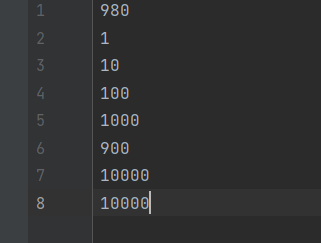
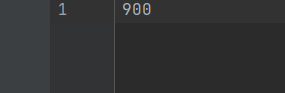
        N\_ -= i[0]

        min\_sum += i[1]

f = open('output.txt', 'w')

f.write(str(min(min\_in, min\_sum)))

C помощью двумерного массива numbers, состоящего из массивов из двух элементов (10^j как количество листов, о котором говорится в задании, и введенного из строки числа), затем отсортированного по отношению числа к 10^j для высчитывания более оптимального выбора. Для высчитывания оптимального выбора берем две переменные для различного поиска минимальных (чтобы потом выбрать среди них наименьшее), одна из которых получается путем присваивания целого элемента части массива, итерируемого как i, а вторая – путем суммы частей.

Вывод по задаче: смесь динамического программирования и циклов

## Задача №11. Максимальное количество золота [1 баллов]

Даны n золотых слитков, найдите максимальный вес золота, который поместится в сумку вместимостью W.

with open('input.txt') as file:

    W, n = map(int, file.readline().split())

    A = list(map(int, file.readline().split()))

F = [1] + [0] \* W

F\_new = F[:]

for j in range(len(A)):

    for i in range(A[j], W + 1):

        if F[i - A[j]] == 1:

            F\_new[i] = 1

    F = F\_new[:]

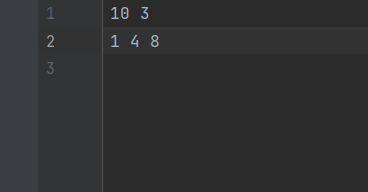
i = W

while F[i] == 0:

    i -= 1

f = open('output.txt', 'w')

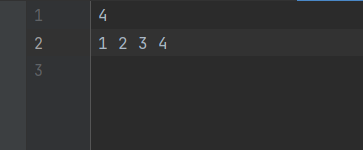
f.write(str(i))

## Задача №12. Последовательность [1 баллов]

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.

import time  
  
t\_start = time.perf\_counter()  
import tracemalloc  
  
tracemalloc.start()  
  
  
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'  
  
  
with open('input.txt') as inp:  
 ln = inp.readlines()  
arr = list(map(int, ln[1].split()))  
answ = find\_a\_middle(arr)  
print(answ)  
with open('output.txt', 'wt') as outp:  
 if answ == '-1':  
 outp.write("-1")  
 exit  
 outp.write(str(len(answ[0])) + "\n")  
 out = ''  
 for i in range(len(answ[0])):  
 out += str(answ[0][i]) + " "  
 outp.write(out)

## Задача №14. Макс значение арифметического выражения [2 баллов]

В этой задаче ваша цель - добавить скобки к заданному арифметическому

выражению, чтобы максимизировать его значение.

with open('input.txt') as file:

    str1 = file.readline()

d, op, m, M = [], [], [], []

for i in str1:

    if i == "+" or i == "\*" or i == "-":

        op.append(i)

    else:

        d.append(int(i))

def MinandMax(i, j, m, M, op):

    minimum1 = float("+inf")

    maximum1 = float("-inf")

    for k in range(i, j):

        if op[k] == '\*':

            a = M[i][k] \* M[k + 1][j]

            b = M[i][k] \* m[k + 1][j]

            c = m[i][k] \* M[k + 1][j]

            d = m[i][k] \* m[k + 1][j]

        elif op[k] == '+':

            a = M[i][k] + M[k + 1][j]

            b = M[i][k] + m[k + 1][j]

            c = m[i][k] + M[k + 1][j]

            d = m[i][k] + m[k + 1][j]

        else:

            a = M[i][k] - M[k + 1][j]

            b = M[i][k] - m[k + 1][j]

            c = m[i][k] - M[k + 1][j]

            d = m[i][k] - m[k + 1][j]

        minimum1 = min(minimum1, a, b, c, d)

        maximum1 = max(maximum1, a, b, c, d)

    return minimum1, maximum1

def maxValue(d, op):

    n = len(d)

    m = []

    M = []

    for i in range(len(d)):

        m.append([])

        M.append([])

        for j in range(len(d)):

            m[i].append(0)

            M[i].append(0)

    for i in range(n):

        m[i][i] = d[i]

        M[i][i] = d[i]

    for s in range(1, n):

        for i in range(n - s):

            j = i + s

            m[i][j], M[i][j] = MinandMax(i, j, m, M, op)

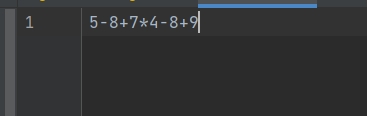
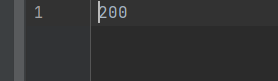
    return M[0][n - 1]

answer = maxValue(d, op)

with open("output.txt", "w") as f:

    f.write(str(answer))

Первая функция служит для распределения минимума и максимума наших операций

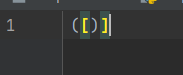
Вывод: стал великим математиком

## Задача №15. Удаление скобок [2 балла]

Реализовать алгоритм удаления скобок

maximum = 10\*\*9  
  
with open('input.txt') as file:  
 s = file.readline()  
  
  
n = len(s)  
  
dp = [[maximum]\*(n + 1) for i in range(n + 1)]  
best = [[-1]\*(n + 1) for i in range(n + 1)]  
  
for i in range(n + 1):  
 dp[i][i] = 1  
  
for i in range(n - 1):  
 if (s[i] == '(' and s[i + 1] == ')') or (s[i] == '[' and s[i + 1] == ']') or (s[i] == '{' and s[i + 1] == '}'):  
 dp[i][i + 1] = 0  
 best[i][i + 1] = -2  
  
for L in range(2, n + 1):  
 for left in range(n - L + 1):  
 right = left + L - 1  
 if (s[left] == '(' and s[right] == ')') or (s[left] == '[' and s[right] == ']') or (s[left] == '{' and s[right] == '}'):  
 if dp[left][right] > dp[left + 1][right - 1]:  
 dp[left][right] = dp[left + 1][right - 1]  
 best[left][right] = -2  
 for mid in range(left, right):  
 cur = dp[left][mid] + dp[mid + 1][right]  
 if dp[left][right] > cur:  
 dp[left][right] = cur  
 best[left][right] = mid  
  
  
res = list(s)  
  
  
def ans(left, right):  
 if best[left][right] == -1:  
 res[left] = res[right] = ''  
 if left == right:  
 return  
 if best[left][right] == -2:  
 if left + 1 == right:  
 return  
 ans(left + 1, right - 1)  
 else:  
 ans(left, best[left][right])  
 ans(best[left][right] + 1, right)  
 return  
  
  
ans(0, n - 1)  
print(\*res, sep='')

Первый делом определяем какое кол-во скобок можно удалить, чтобы получилась правильная последовательность. Вторым делом выбираем наилучший вариант и записываем его в ответ.

## Задача №17. Ход конем [2.5 баллов]

Напишите программу, определяющую количество телефонных номеров длины N, набираемых ходом коня.

import copy

with open('input.txt') as file:

    length\_of\_number = int(file.readline())

def find\_value\_of\_numbers(length\_of\_number):

    previos = [4, 2, 1, 0]

    current = [0 for \_ in range(4)]

    if length\_of\_number > 1:

        for i in range(length\_of\_number - 1):

            current[0] += previos[1] \* 2 + previos[2] \* 2

            current[1] += previos[0] + previos[3] \* 2

            current[2] += previos[0]

            current[3] += previos[1]

            previos = copy.copy(current)

            current = [0 for \_ in range(4)]

        return sum(previos)

    else:

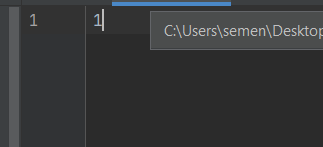
        return 8

result = find\_value\_of\_numbers(length\_of\_number)

f = open('output.txt', 'w')

f.write(str(result))

Для решения этой задачи применим метод динамического программирования. Пусть b[d][k] – количество номеров, набираемых ходом коня, которые начинаются с цифры d и состоят из k цифр. Тогда b[d][1]=1 для всех d, а b[d][k] для любого d вычисляется через сумму b[i][k-1] для k>1. Так, например, b[4][k] = b[0][k-1]+b[3][k-1]+b[9][k-1]. Увеличивая k от 2 до n мы получим значения b[d][n], сумма которых (за вычетом b[0][n] и b[8][n]) и даст ответ на поставленную задачу.

Вывод по задаче: благодаря простому динамическому программированию задача решается просто независимо от коня.

## Задача №19. Произведение матриц [3 баллов]

Требуется найти оптимальную расстановку скобок в произведении после-

довательности матриц.

import copy

with open('input17.txt') as file:

    length\_of\_number = int(file.readline())

def find\_value\_of\_numbers(length\_of\_number):

    previos = [4, 2, 1, 0]

    current = [0 for \_ in range(4)]

    if length\_of\_number > 1:

        for i in range(length\_of\_number - 1):

            current[0] += previos[1] \* 2 + previos[2] \* 2

            current[1] += previos[0] + previos[3] \* 2

            current[2] += previos[0]

            current[3] += previos[1]

            previos = copy.copy(current)

            current = [0 for \_ in range(4)]

        return sum(previos)

    else:

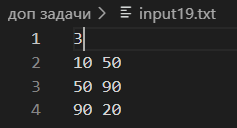
        return 8

result = find\_value\_of\_numbers(length\_of\_number)

f = open('output17.txt', 'w')

f.write(str(result))

Реализация: для работы с матрицами создаем три массива, два из которых – двумерные: matr\_sizes для хранения размеров матриц, min\_ops – для минимального количества матриц, choices – для хранения нужных k. Создаем две функции: matrixMult для самого перемножения матриц и matrixCreation для подбора нужной комбинации скобок.

 Изображение выглядит как текст

Автоматически созданное описание

Вывод по задаче: одна из сложнейших задач в этой лабе, после продумывания алгоритма правильного перемножения возникали ошибки правильности скобок, но вконце концов результат меня порадовал.

## Задача №21. Игра в дурака [3 баллов]

Пете необходимо решить следующую задачу: сможет ли игрок, обладая

набором из N карт, отбить M карт, которыми под него сделан ход?.

from itertools import groupby

from collections import defaultdict

cards\_priority = {

    "6": 0,

    "7": 1,

    "8": 2,

    "9": 3,

    "T": 4,

    "J": 5,

    "Q": 6,

    "K": 7,

    "A": 8,

}

def beat(deck, card):

    for c in deck:

        if cards\_priority[c[0]] > cards\_priority[card[0]]:

            return c

with open("input.txt") as f:

    \_, \_, trump = f.readline().split()

    deck = [(i[0], i[1]) for i in f.readline().split()]

    cards = [(i[0], i[1]) for i in f.readline().split()]

    deck.sort(key=lambda x: (x[1] != trump, cards\_priority[x[0]]))

    cards.sort(key=lambda x: (x[1] != trump, cards\_priority[x[0]]))

    grouped\_deck = {"S": [], "C": [], "D": [], "H": []}

    for key, items in groupby(deck, lambda x: x[1]):

        grouped\_deck[key] = list(items)

    for card in cards:

        val, suit = card

        if suit != trump:

            res = beat(grouped\_deck[suit], card)

            if res is not None:

                grouped\_deck[suit].remove(res)

                continue

        if suit == trump:

            res = beat(grouped\_deck[trump], card)

        elif len(grouped\_deck[trump]) > 0:

            res = grouped\_deck[trump][0]

        else:

            res = None

        if res is not None:

            grouped\_deck[trump].remove(res)

            continue

        print("NO")

        quit()

    print("YES")

Мы читаем входные данные и сортируем колоду и карты. Затем мы группируем колоду по мастям и перебираем все карты в руке, если найденная карта козырная. Если да, то берем первую карту из козырной колоды. Если это не козырь, то мы пытаемся побить его картами той же масти. Если мы не можем побить ее картами той же масти, тогда мы пытаемся побить ее козырными картами. Если мы не можем побить его козырями, то печатаем NO и выходим из программы. Если мы можем побить его, то убираем карту из колоды и продолжаем со следующей карты в руке. Если мы побили все карты, то выводим YES.

Результат работы кода на примерах из текста задачи

Вывод по задаче: Задача помогает разобраться со входными данными и правильной работой со словарями.

## Задача №22. Симпатичный узор [4 баллов]

Нужно создать как можно больше симпатичных узоров.

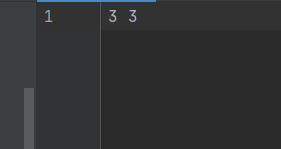
def binnary(i, N):  
 string = format(i, 'b')  
 string = string[::-1]  
 while len(string) < N: string += "0"  
 return list(map(int, list(string)))  
  
  
def check(array\_1, array\_2):  
 for i in range(len(array\_1)): array\_1[i] += array\_2[i]  
 for i in range(1, len(array\_1)):  
 if array\_1[i] == 0 and array\_1[i - 1] == 0: return False  
 if array\_1[i] == 2 and array\_1[i - 1] == 2: return False  
 return True  
  
  
def opportunities(N):  
 results = []  
 var\_1 = generator\_options(N)  
 for i in var\_1:  
 results.append([])  
 var\_2 = generator\_options(N)  
 for j in var\_2:  
 if check(i[:], j[:]):  
 results[-1].append(1)  
 else:  
 results[-1].append(0)  
 return results  
  
  
def generator\_options(N):  
 for i in range(pow(2, N)): yield binnary(i, N)  
  
  
def main(array, W):  
 result = [1] \* len(array)  
 new = [0] \* len(array)  
  
 for i in range(W - 1):  
 for j in range(len(new)):  
 for k in range(len(array[j])):  
 if array[j][k] == 1: new[j] += result[k]  
 result = new.copy()  
 new = [0] \* len(array)  
  
 return sum(result)  
  
  
input\_file = open("input.txt", "r")  
H, W = map(int, input\_file.readline().split())  
result = main(opportunities(min(H, W)), max(H, W))  
  
out\_file = open("output.txt", "w")  
out\_file.write(str(result))

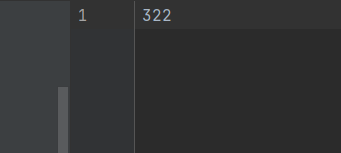
Создаем поле нужного размера.

Проверяем на наличие симпатичного узора все ячейки подряд кроме Правых крайних и нижнего ряда

Вывод:

Теперь смогу красиво плитку на даче положить.





# Вывод

Полное удовольствие при отрешивании задач динамического программирования, конечно сжатые сроки оставляли желать лучшего при начинании, но в целом результат в дальнейшем повлияет на ускорение алгоритмов и понимание работы с ними.