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.

Вариант 29

Выполнил:

Логачев Д.С.

К3139/К32402

Проверила:

Артамонова В.Е.

Санкт-Петербург

2022 г.

# Содержание отчета

[Содержание отчета 2](#_Toc115804658)

[Задачи по варианту 3](#_Toc115804659)

[3 задача. Максимальный доход от рекламы (0.5 балла) 3](#_Toc115804660)

[7 задача. Проблема сапожника (0.5 балла) 3](#_Toc115804661)

[12 задача. Последовательность (1 балл) 4](#_Toc115804662)

[17 задача. Ход конем (2,5 балла) 6](#_Toc115804663)

[20 задача. Почти палиндром (3 балла) 7](#_Toc115804664)

[Вывод 8](#_Toc115804665)

# Задачи по варианту

## 3 задача. Максимальный доход от рекламы (0.5 балла)

У вас есть n объявлений для размещения на популярной интернет-странице.

Для каждого объявления вы знаете, сколько рекламодатель готов платить за один клик по этому объявлению. Вы настроили n слотов на своей странице и оценили ожидаемое количество кликов в день для каждого слота. Теперь ваша цель - распределить рекламу по слотам, чтобы максимизировать общий доход.

def max\_income(array\_1, array\_2):

maximum = 0

for i in range(len(array\_1)):

maximum += array\_1[i] \* array\_2[i]

return maximum

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

n = int(f.readline())

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

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

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

f.write(str(max\_income(sorted(a),sorted(b))))

Чтобы получить максимальную сумму, надо максимальное значение умножать на максимальное кол-во кликов, минимальное – на минимальное. Таким образом нам надо отсортировать массив и просуммировать произведения.

 

## 7 задача. Проблема сапожника (0.5 балла)

В некоей воинской части есть сапожник. Рабочий день сапожника длится K минут. Заведующий складом оценивает работу сапожника по количеству починенной обуви, независимо от того, насколько сложный ремонт требовался в каждом случае. Дано n сапог, нуждающихся в починке. Определите, какое максимальное количество из них сапожник сможет починить за один рабочий день.

def counter(array, time):

array.sort()

count = 0

for i in array:

if time < i:

break

time -= i

count += 1

return count

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

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

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

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

f.write(str(counter(array,time)))

У нас есть все данные, нам просто надо пройтись по количеству сапог, как только сапожник починил сапог – удаляем затраченное время из наших рабочих часов, при этом для того, чтобы починить наибольшее количество сапог – надо отсортировать массив времени починки по возрастанию.

 

## 12 задача. Последовательность (1 балл)

Дана последовательность натуральных чисел a1, a2, ..., an, и известно, что ai ≤ i для любого 1 ≤ i ≤ 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.

def func(values: list):

halfSumElem = sum(values) // 2

first = list()

second = list()

for value in values:

if (sum(first) + value) <= halfSumElem:

first.append(value)

else:

second.append(value)

if sum(first) == sum(second) == halfSumElem:

return len(second), second

else:

return -1

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

n = f.readline()

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

foo = func(values)

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

if foo != -1:

length,array = foo[0],foo[1]

f.write(f"{length}\n{' '.join(map(str,array))}")

else:

f.write(str(foo))

Создадим две подпоследовательности A и B. Найдем сразу же условие «половине суммы всех элементов последовательности», от него и будем отталкиваться. Пойдем циклом по последовательности и будем добавлять в А, пока оно не достигнет половины суммы, если же оказывается так, что значение, которое надо добавить в A дает сумму больше половины всей суммы последовательности, то закинем ее в B. Если суммы подпоследовательностей равны половине сумме всех элементов последовательности - выводим результат.



## 17 задача. Ход конем (2,5 балла)

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

dict = {0: [4, 6], 1: [6, 8], 2: [7, 9], 3: [8, 4], 4: [3, 9, 0], 5: [], 6: [1, 7, 0], 7: [6, 2], 8: [1, 3], 9: [2, 4]}

startCount = {0: 0, 1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 8: 0, 9: 1}

endCount = {}

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

length = int(f.readline())

for i in range(length - 1):

for key in dict:

sum = 0

for j in dict[key]:

sum += startCount[j]

endCount[key] = sum

startCount = endCount

endCount = {}

sum = 0

for key in startCount:

sum += startCount[key]

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

f.write(str(sum % (10 \*\* 9)))

Создадим изначальный словарь, в котором будет храниться каждая

кнопка в виде ключа и места куда из нее можно прийти ходом конем в виде значений. Также создадим словарь с изначальным количеством возможных

номеров для каждой цифры, изначально они будут равняться 1, кроме цифр 8 и 0, далее этот словарь будет перезаписываться, чтобы не потерять значения (с помощью нового пустого словаря). Теперь в зависимости от количества цифр в номере мы будем пробегаться по нашему словарю и перезаписывать количество возможных комбинаций.



## 20 задача. Почти палиндром (3 балла)

Слово называется палиндромом, если его первая буква совпадает с последней, вторая – с предпоследней и т.д. Например: «abba», «madam», «x». Для заданного числа K слово называется почти палиндромом, если в нем можно изменить не более K любых букв так, чтобы получился палиндром. Например, при K = 2 слова «reactor», «kolobok», «madam» являются почти палиндромами (подчеркнуты буквы, заменой которых можно получить палиндром). Подсловом данного слова являются все слова, получающиеся путем вычеркивания из данного нескольких (возможно, одной или нуля) первых букв и нескольких последних. Например, подсловами слова «cat» являются слова «c», «a», «t», «ca», «at» и само слово «cat» (а «ct» подсловом слова «cat» не является). Требуется для данного числа K определить, сколько подслов данного слова S являются почти палиндромами.

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

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

string = f.read()

pali = [[0 for i in range(n)] for j in range(n)]

for l in range(n + 1, -1, -1):

for r in range(1, n):

if r > l:

pali[l][r] = pali[l + 1][r - 1] if string[l] == string[r] else pali[l + 1][r - 1] + 1

for line in pali:

print(line)

el\_to\_swap = 0

for l\_board in range(n):

for r\_board in range(l\_board, n):

if pali[l\_board][r\_board] <= k:

el\_to\_swap += 1

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

f.write(str(el\_to\_swap))

Создадим таблицу палиндромов pali. Пусть pali[l][r] = количество символов которые нужно заменить чтобы подстрока s[l:r] стала палиндромом

База pali[l][l] = 0

Переход pali[l][r] = pali[l+1][r-1] (+1 если s[l] != s[r]).

# Вывод

В ходе данной лабораторной работы была освоена темы «Жадные алгоритмы» и «Динамическое программирование», на практике применены такие методы работы, как нахождение наибольшего числа, работа с последовательностями и палиндромами.