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: Тема работы

Вариант 7

Выполнил:

Крылов Михаил Максимович

К3240

Проверил:

Афанасьев А.В.

Санкт-Петербург

2023 г.

**Содержание отчета**

Оглавление

[**Содержание отчета** 2](#_Toc178413019)

[**Задачи по варианту** 3](#_Toc178413020)

[**Задача №1. Максимальная стоимость добычи [0.5 баллов]** 3](#_Toc178413021)

[**Задача №6. Максимальная зарплата [0.5 баллов]** 6](#_Toc178413022)

[**Задача №13. Сувениры [1.5 баллов]** 8](#_Toc178413023)

[**Задача №14. Максимальное значение арифметического выражения** **[2 балла] \*[Замена 15-го задания]** 10](#_Toc178413024)

[**Задача №18. Кафе** **[2.5 балла]** 12](#_Toc178413025)

[**Вывод** 16](#_Toc178413026)

**Задачи по варианту**

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

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 pi и wi – стоимость и вес i-го предмета, соответственно.

• Ограничения на входные данные. 1 ≤ n ≤ 103 , 0 ≤ W ≤ 2 · 106 , 0 ≤ pi ≤ 2 · 106 , 0 ≤ wi ≤ 2 · 106 для всех 1 ≤ i ≤ 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).

Листинг кода

def func(n, W, list):

    list.sort(key=lambda x: x[1], reverse=True)

    k = 0

    for i in list:

        if W < i[0]:

            k += W / i[0] \* (i[1] \* i[0])

            break

        k += i[0]\*i[1]

        W -= i[0]

    return f'{k:.4f}'

def main():

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

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

        list = []

        for i in range(n):

            p, w = map(int, file.readline().split())

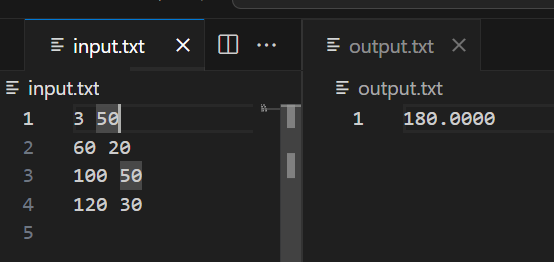
            list.append((w, p/w))

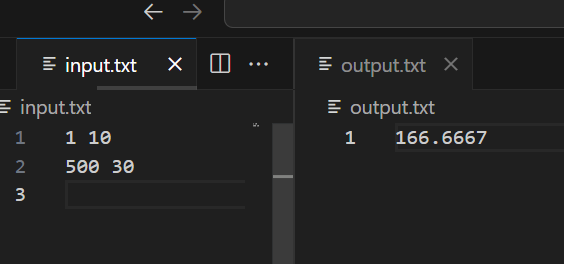
    result = func(n, W, list)

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

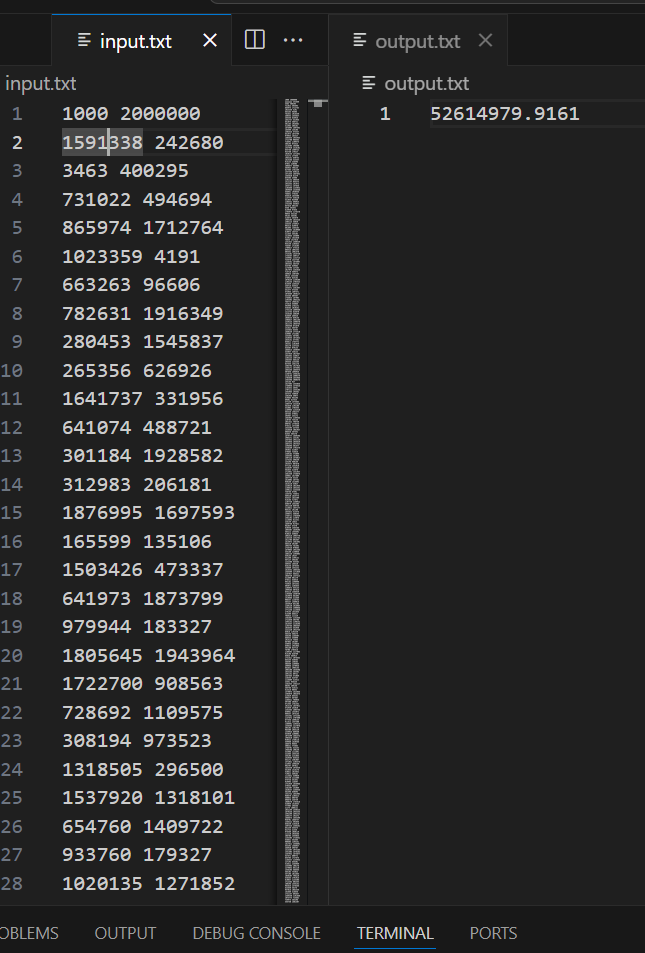
        file.write(result)

После считывания данных с файла высчитываем соотношение стоимости к весу. Далее сортируем массив по этому соотношению. Перебираем элементы массива. Проверяем, полностью ли помещается добыча в сумку, если да – добавляем её стоимость к переменной **k**. Если не помещается полностью – добавляем к переменной **k** стоимость той части добычи, что помещается. Далее возвращаем **k**.





Результат работы кода на максимальных и минимальных значениях:



|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.5401 | 22.2144 MiB |
| Пример из задачи | 0.5501 seconds | 23.2344 MiB |
| Пример из задачи | 0.5610 seconds | 23.3672 MiB |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.5544 seconds | 23.7070 MiB |

Вывод по задаче: Вспомнил работу с модулями для замерами времени и памяти. Незначительная разница в времени выполнения объясняется низкой сложностью алгоритма, а 0.5 в замерах появляется, т.к. одновременно идёт замер памяти и времени. Из-за чего показания не точные.

**Задача №6. Максимальная зарплата [0.5 баллов]**

• Постановка задачи. Составить наибольшее число из набора целых чисел.

• Формат ввода / входного файла (input.txt). Первая строка входных данных содержит целое число n. Во второй строке даны целые числа a1, a2, ..., an.

• Ограничения на входные данные. 1 ≤ n ≤ 102 , 1 ≤ ai ≤ 103 для всех 1 ≤ i ≤ n.

• Формат вывода / выходного файла (output.txt). Выведите наибольшее число, которое можно составить из a1, a2, ..., an.

• Ограничение по времени. 2 сек.).

Листинг кода

def func(n, arr):

    for i in range(n - 1):

        for j in range(n - i - 1):

            if arr[j] + arr[j + 1] < arr[j + 1] + arr[j]:

                arr[j], arr[j + 1] = arr[j + 1], arr[j]

    return "".join(arr)

def main():

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

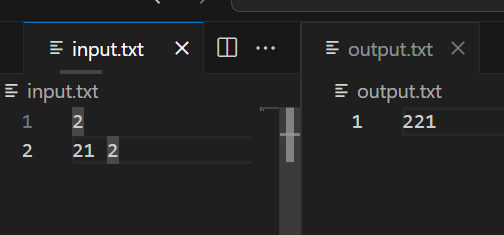
        n = int(file.readline())

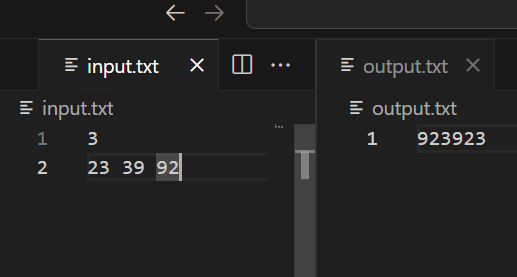
        arr = list(file.readline().split())

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

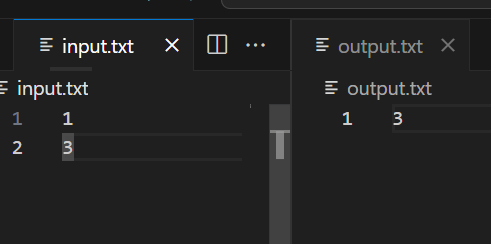
        file.write(str(func(n, arr)))

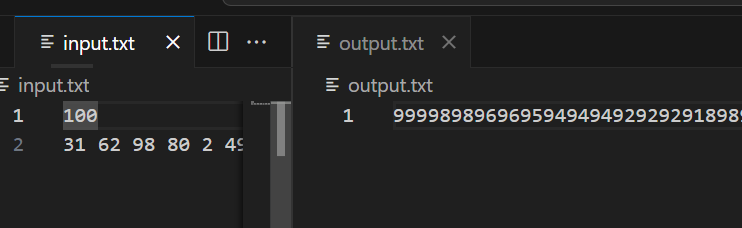
С помощью пузырьковой сортировки и встроенным в питон методом сравнением строк сортируем массив из чисел по уменьшению и получаем наибольшее число, которое мы можем составить.





Результат работы кода на максимальных и минимальных значениях:





|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.5439 seconds | 23.4883 MiB |
| Пример из задачи | 0.5410 seconds | 23.3945 MiB |
| Пример из задачи | 0.5954 seconds | 23.4258 MiB |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.5350 seconds | 23.5000 MiB |

Вывод по задаче: Вспомнили пузырьковую сортировку.

**Задача №13. Сувениры [1.5 баллов]**

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 v1, v2, ..., vn, разделенные пробелами.

• Ограничения на входные данные. 1 ≤ n ≤ 20, 1 ≤ vi ≤ 30 для всех i.

• Формат вывода / выходного файла (output.txt). Выведите 1, если можно разбить v1, v2, ..., vn на три подмножества с одинаковыми суммами и 0 в противном случае.

• Ограничение по времени. 5 сек.

Листинг кода

list1 = []

data = {}

def func(sum1, sum2, sum3, i):

    if sum([sum1, sum2, sum3]) == sum(list1):

        if sum1 == sum2 == sum3:

            return 1

        return 0

    b = (sum1, sum2, sum3)

    if b in data:

        return data[b]

    result = max(

        func(sum1 + list1[i], sum2, sum3, i+1),

        func(sum1, sum2 + list1[i], sum3, i+1),

        func(sum1, sum2, sum3 + list1[i], i+1)

    )

    data[b] = result

    return result

def main():

    global list1, data

    with open('input.txt') as file:

        n = int(file.readline())

        list1 = list(map(int, file.readline().split()))

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

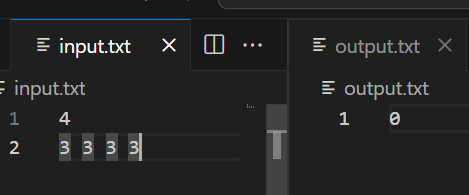
        if sum(list1) % 3 == 0:

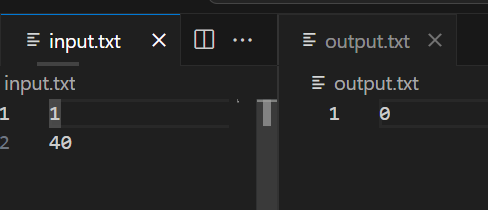
            file.write(str(func(0, 0, 0, 0)))

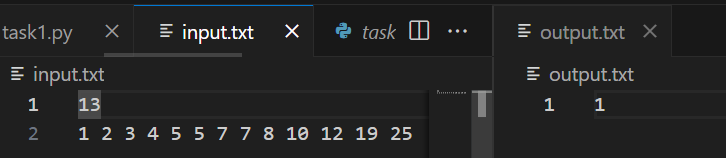
        else:

            file.write('0')

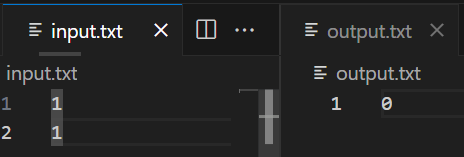
Для начала создаем 2 переменные list1 и data, которые будут глобальными. Чтоб каждый раз не передавать их в функцию, тем самым, не занимая лишнее место. Далее обрабатываем информацию из input.txt и создаем массив. Также создаем функцию func, которая рекурсивно вызывает сама себя, перебирая возможные варианты разделения сувениров между друзей. Когда мы находим ситуацию, в которой сувениры поделены поровну то возвращаем 1. Во всех остальных случаях – 0.

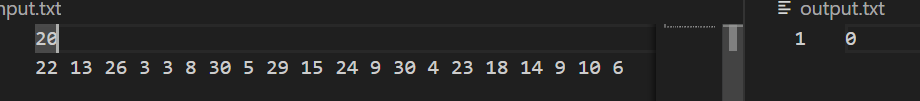






Результат работы кода на максимальных и минимальных значениях:





|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.5436 seconds | 23.4283 MiB |
| Пример из задачи | 0.5433 seconds | 23.3203 MiB |
| Пример из задачи | 0.5639 seconds | 23.3867 MiB |
| Пример из задачи | 0.5627 seconds | 24.4336 MiB |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.5648 seconds | 23.4727 MiB |

Вывод по задаче: вспомнили как решать задачи на динамическое программирование через рекурсивные функции.

**Задача №14. Максимальное значение арифметического выражения** **[2 балла] \*[Замена 15-го задания]**

В этой задаче ваша цель - добавить скобки к заданному арифметическому выражению, чтобы максимизировать его значение. max(5 − 8 + 7 × 4 − 8 + 9) =?

• Постановка задачи. Найдите максимальное значение арифметического выражения, указав порядок применения его арифметических операций с помощью дополнительных скобок.

• Формат ввода / входного файла (input.txt). Единственная строка входных данных содержит строку s длины 2n + 1 для некоторого n с символами s0, s1, ..., s2n. Каждый символ в четной позиции s является цифрой (то есть целым числом от 0 до 9), а каждый символ в нечетной позиции является одной из трех операций из +, −, ∗

• Ограничения на входные данные. 0 ≤ n ≤ 14 (следовательно, строка содержит не более 29 символов).

• Формат вывода / выходного файла (output.txt). Выведите максимально возможное значение заданного арифметического выражения среди различных порядков применения арифметических операций.

• Ограничение по времени: 5 сек.

Листинг кода

def func(expr):

    n = (len(expr) // 2) + 1

    MVal = [[-float('inf')] \* n for \_ in range(n)]

    mVal = [[float('inf')] \* n for \_ in range(n)]

    for i in range(n):

        MVal[i][i] = mVal[i][i] = int(expr[2 \* i])

    for l in range(2, n + 1):

        for i in range(n - l + 1):

            j = i + l - 1

            for k in range(i, j):

                op = expr[2 \* k + 1]

                if op == '-':

                    MVal[i][j] = max(MVal[i][j], eval(f"MVal[i][k] {op} mVal[k + 1][j]"))

                    mVal[i][j] = min(mVal[i][j], eval(f"mVal[i][k] {op} MVal[k + 1][j]"))

                else:

                    MVal[i][j] = max(MVal[i][j], eval(f"MVal[i][k] {op} MVal[k + 1][j]"))

                    mVal[i][j] = min(mVal[i][j], eval(f"mVal[i][k] {op} mVal[k + 1][j]"))

    return MVal[0][n - 1]

def main():

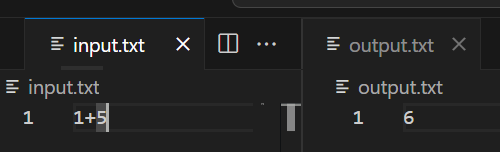
    with open('input.txt') as file:

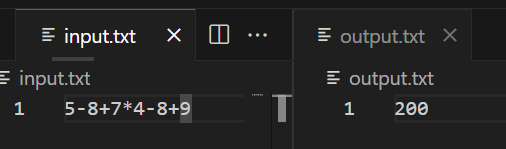
        expr = file.readline()

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

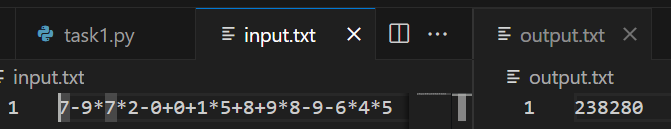
        file.write(str(func(expr)))

Считываем строку из входного файла. Потом создаем 2 матрицы, одна отвечает за максимальные значения, другая – за минимальные. Заполняем матрицы по диагонали значениями из входного файла. Далее циклом проходимся по операциям, данным нам, заполняя матрицы следующим образом – для вычисления значений матриц максимальных значений используем для операций “+” и “\*” только матрицу максимальных значений, т.к. эти операции дадут наибольшую сумму. Для операции “-“ используем за основу матрицу максимальных значений и вычитаем из матрицы минимальных значений. Все эти же операции для матрицы минимальных значений используем в обратную сторону. Далее выводим





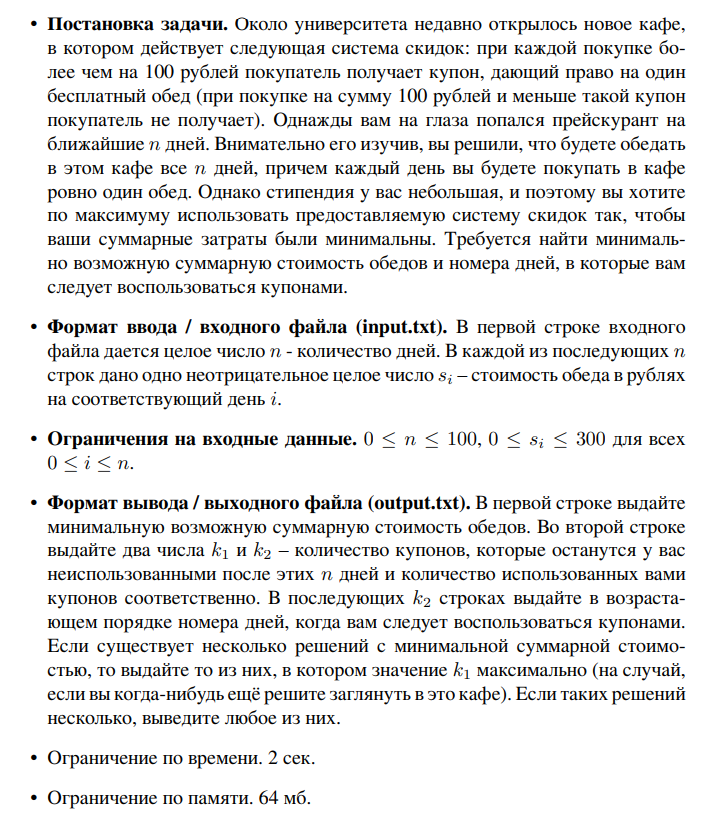
Результат работы кода на максимальных и минимальных значениях:



|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.5266 seconds | 23.5625 MiB |
| Пример из задачи | 0.5590 seconds | 23.3203 MiB |
| Пример из задачи | 0.6064 seconds | 23.3594 MiB |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.6264 seconds | 23.4648 MiB |

Вывод по задаче: вспомнили как решать задачи с помощью матриц.

**Задача №18. Кафе** **[2.5 балла]**

Листинг кода

from copy import deepcopy

n = 0

min\_price = float('inf')

data = [0, 0]

history = []

def func(full\_price, tickets, i, all\_tickets, hs=[]):

    if i == n:

        global min\_price, data, history

        if full\_price < min\_price or (full\_price == min\_price and tickets > data[0]):

            min\_price = full\_price

            data = [tickets, all\_tickets]

            history = hs

        return full\_price

    k = float('inf')

    if list1[i] > 100:

        g = func(full\_price+list1[i], tickets+1, i+1, all\_tickets+1, deepcopy(hs))

    else:

        g = func(full\_price+list1[i], tickets, i+1, all\_tickets, deepcopy(hs))

    if tickets > 0:

        hs.append(i+1)

        k = func(full\_price, tickets-1, i+1, all\_tickets, deepcopy(hs))

    return min([k, g])

def main():

    global list1, n, min\_price, data

    with open('input.txt') as file:

        n = int(file.readline())

        list1 = list(map(int, file.readlines()))

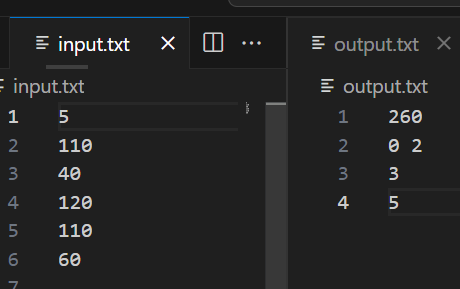
    with open("output.txt", "w") as file:

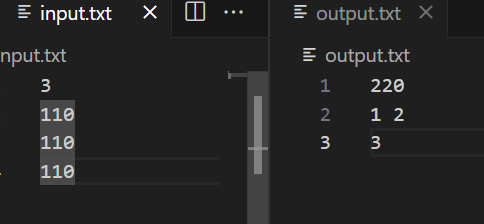
        file.write(str(func(0, 0, 0, 0))+'\n')

        file.write(' '.join(map(str, data))+'\n')

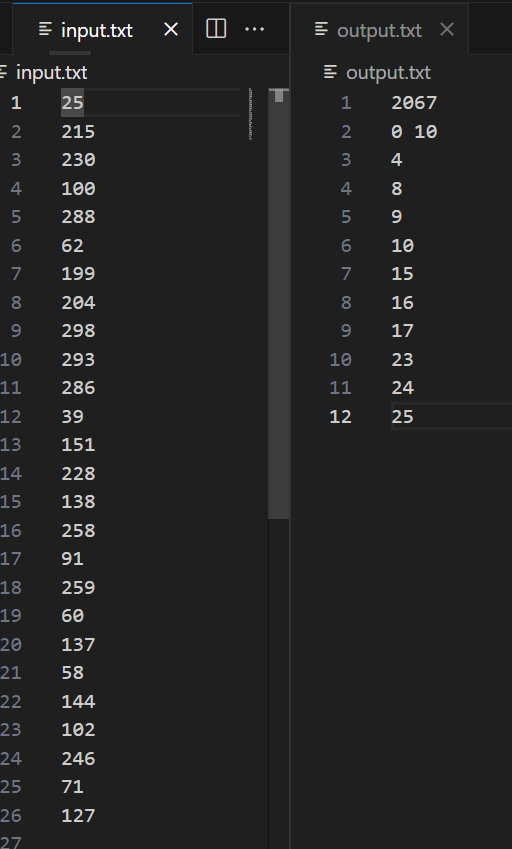
        file.write('\n'.join(map(str, history)))

Считываем строку из входного файла. Потом создаем 4 глобальные переменные, которые будут хранить конечный ответ. С помощью рекурсивного алгоритма перебираем возможные варианты - проверяем стоимость обеда в кафе, если больше 100 рублей - то запускаем рекурсивную функцию с добавлением 1 билета на бесплатный обед, запоминая общую стоимость за все дни. Если стоимость 100 или меньше – то запускаем рекурсию без учета нового билета. Также, запускаем рекурсию с учетом того, что мы пообедали за счет бесплатного билета. Таким образом, перебираем все возможные варианты развития событий. Когда дни заканчиваются – проверяем общую стоимость. Если она ниже прошлой – запоминаем её, также запоминаем количество оставшихся билетов и дни, в которые билеты нужно использовать.





Результат работы кода на максимальных и минимальных значениях:



|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.5527 seconds | 23.4375 MiB |
| Пример из задачи | 0.5996 seconds | 23.4766 MiB |
| Пример из задачи | 0.5555 seconds | 23.4219 MiB |
| Верхняя граница диапазона значений входных данных из текста задачи | 14.0440 seconds | 23.4648 MiB |

Вывод по задаче: научились использовать динамическое программирование на более сложных задачах.

**Вывод**

Данная лабораторная работа позволила вспомнить основы сортировок а также закрепить на практике решение задач с помощью динамического программирования.