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 21

Выполнила:
Савченко А.С.
К3141

Проверил:
Афанасьев А.В.

Санкт-Петербург
2024 г.

Содержание отчета

Задачи по варианту

Задача №1. Максимальная стоимость добычи [0.5 баллов]

Текст задачи:

1 задача. 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.** $1 \le n \le 10^3, 0 \le W \le 2 \cdot 10^6, 0 \le p_i \le 2 \cdot 10^6, 0 \le w_i \le 2 \cdot 10^6$ для всех $1 \le i \le n$. 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.
- Примеры:

input.txt	output.txt
3 50 60 20 100 50 120 30	180.0000

Чтобы получить значение 180, берем первый предмет и третий предмет в сумку.

input.txt	output.txt
1 10 500 30	166.6667

Здесь просто берем одну треть единственного доступного предмета.

Листинг кода:

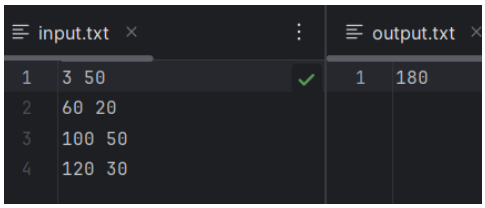
```
def unit_cost_sort(l):
        return   l.sort(key=lambda   x:   x[0]   /   x[1],
reverse=True)
```

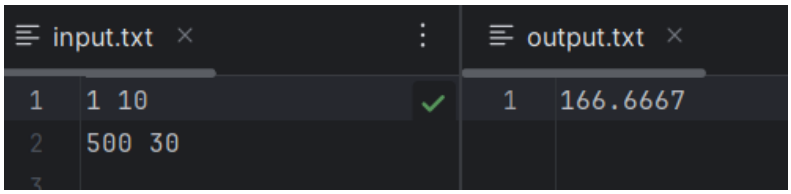
```
with open("input.txt", "r", encoding='utf-8') as
input_file, open("output.txt", "w", encoding='utf-8')
as output_file:
    n, w = map(int, input_file.readline().split())
            items = [[int(j) for j in
input_file.readline().split()] for _ in range(n)]

    unit_cost_sort(items)

    total = 0
    i = 0
    while w > 0:
        if i == n:
            break
        elif items[i][1] <= w:
            w -= items[i][1]
            total += items[i][0]
            i += 1
        else:
            total += (items[i][0] / items[i][1]) * w
            w = 0

    output_file.write(str(round(total, 4)))
```

Текстовое объяснение решения:

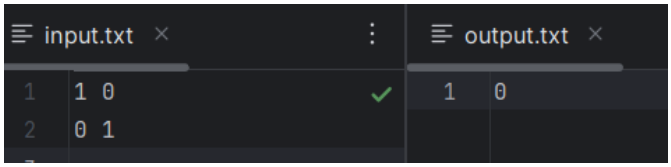
Считываем данные из файла. Сортируем предметы в порядке убывания их удельной стоимости. В цикле пока в рюкзаке есть место, проверяем остались ли предметы для сбора (если нет выходим из цикла), проверяем можем ли положить предмет целиком, если да - уменьшаем оставшееся место в рюкзаке, увеличиваем итоговую стоимость добычи на стоимость предмета, переходим к следующему предмету. Если положить целиком не удается, то кладем часть предмета, увеличиваем итоговую стоимость добычи на удельную стоимость предмета умноженную на сколько кг можно запихнуть в рюкзак, места не осталось = 0. Выводим итоговую стоимость добычи с округление до 4-го знака после запятой.

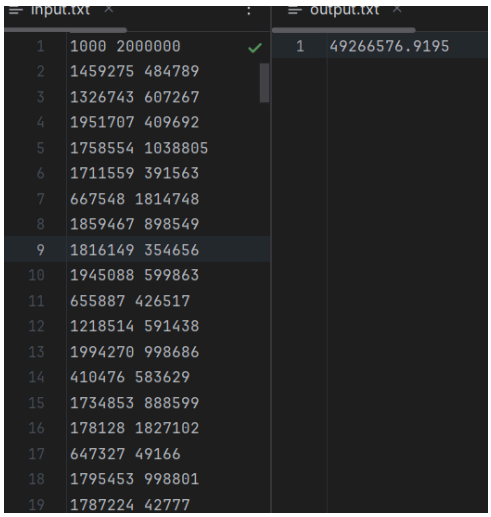
Результат работы кода на примерах из текста задачи:(скрины input output файлов)





Результат работы кода на минимальных и максимальных значениях:(скрины input output файлов)





	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0004339	14.68359375
Пример из задачи	0.0004453	14.953125
Пример из задачи	0.0004535	14.67578125

Верхняя граница диапазона значений входных данных из текста задачи	0.0012823	15.1640625

Вывод по задаче:

Я реализовала алгоритм для решения задачи о дробном рюкзаке, который,представляет из себя сортировку + жадный алгоритм.

Задача №7. Проблема сапожника[0.5 баллов]

Текст задачи:

7 задача. Проблема сапожника (0.5 балла)

- **Постановка задачи.** 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.
- **Формат ввода / входного файла (input.txt).** В первой строке вводятся натуральные числа K и n. Затем во второй строке идет n натуральных чисел $t_1, ...t_n$ - количество минут, которые требуются, чтобы починить i-й сапог.
- **Ограничения на входные данные.** $1 \leq K \leq 1000$, $1 \leq n \leq 500$, $1 \leq t_i \leq 100$ для всех $1 \leq i \leq n$
- **Формат вывода / выходного файла (output.txt).** Выведите одно число – максимальное количество сапог, которые можно починить за один рабочий день.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Примеры:

input.txt	output.txt
10 3 6 2 8	2

input.txt	output.txt
3 2 10 20	0

Листинг кода. (именно листинг, а не скрины)

```
def partition(arr, low, high):
```

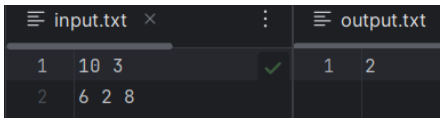
```
    pivot = arr[high]
    i = low - 1
    for j in range(low, high):
        if arr[j] <= pivot:
            i += 1
            arr[i], arr[j] = arr[j], arr[i]
    arr[i + 1], arr[high] = arr[high], arr[i + 1]
    return i + 1


def quick_sort(arr, low, high):
    if low < high:
        pivot = partition(arr, low, high)
        quick_sort(arr, low, pivot - 1)
        quick_sort(arr, pivot + 1, high)


with open("input.txt", "r", encoding='utf-8') as
input_file, open("output.txt", "w", encoding='utf-8')
as output_file:
    k, n = map(int, input_file.readline().split())
    t = [int(i) for i in input_file.readline().split()]

    quick_sort(t, 0, n - 1)

    count = 0
    i = 0

    while i < n and k >= t[i]:
        k -= t[i]
        count += 1
        i += 1

    output_file.write(str(count))
```

Текстовое объяснение решения.

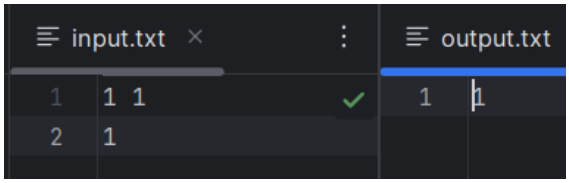
Наиболее выгодно будет делать сапоги с минимальным временем починки. Используем стандартные quick_sort and partition для быстрой сортировки сапогов в порядке возрастания времени.

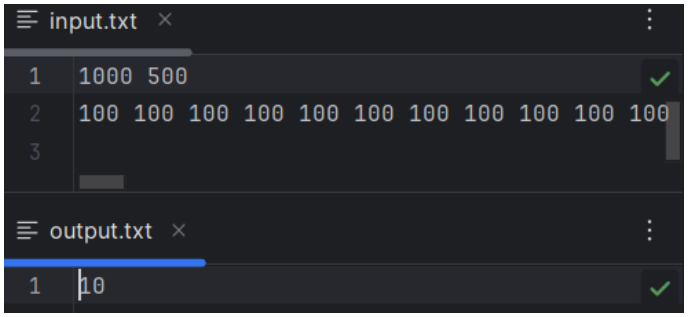
Пока еще есть сапоги, которые нужно починить и оставшееся время работы сапожника меньше или равно времени необходимому на починку еще одного сапога, чиним сапог уменьшить рабочий день сапожника на время, которое понадобилось для починки сапога, переходим к следующему сапогу.

Результат работы кода на примерах из текста задачи:(скрины input output файлов)



Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)





	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0005413	14.73828125
Пример из задачи	0.0004644	14.75390625
Пример из задачи	0.0004688	14.75

Верхняя граница диапазона значений входных данных из текста задачи	0.0080843	14.96171875

Вывод по задаче:

Для этой задачи я воспользовалась быстрой сортировкой для простого жадного алгоритм, работающего на максимальное количество сапог.

Задача №9. Распечатка [1 баллов]

Текст задачи:

9 задача. Распечатка (1 балл)

- **Постановка задачи.** Диссертация дело сложное, особенно когда нужно ее печатать. При этом вам нужно распечатать не только текст самой диссертации, так и другие материалы (задание, рецензии, отзывы, автореферат для защиты и т.п.). Вы оценили объём печати в N листов. Фирма, готовая размножить печатные материалы, предлагает следующие финансовые условия. Один лист она печатает за A_1 рублей, 10 листов - за A_2 рублей, 100 листов - за A_3 рублей, 1000 листов - за A_4 рублей, 10000 листов - за A_5 рублей, 100000 листов - за A_6 рублей, 1000000 листов - за A_7 рублей. При этом не гарантируется, что один лист в более крупном заказе обойдется дешевле, чем в более мелком. И даже может оказаться, что для любой партии будет выгодно воспользоваться тарифом для одного листа. Печать конкретного заказа производится или путем комбинации нескольких тарифов, или путем заказа более крупной партии. Например, 980 листов можно распечатать, заказав печать 9 партий по 100 листов плюс 8 партий по 10 листов, сделав 98 заказов по 10 листов, 980 заказов по 1 листу или заказав печать 1000 (или даже 10000 и более) листов, если это окажется выгоднее. Требуется по заданному объему заказа в листах N определить минимальную сумму денег в рублях, которой будет достаточно для выполнения заказа.
- **Формат ввода / входного файла (input.txt).** На вход программе сначала подается число N – количество листов в заказе. В следующих 7 строках ввода находятся натуральные числа $A_1, A_2, A_3, A_4, A_5, A_6, A_7$ соответственно.
- **Ограничения на входные данные.** $1 \le N \le 2 \times 10^9, 1 \le A_i \le 10^6$
- **Формат вывода / выходного файла (output.txt).** Выведите одно число – минимальную сумму денег в рублях, которая нужна для выполнения заказа. Гарантируется, что правильный ответ не будет превышать 2×10^9.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Примеры:

input.txt	output.txt
980 1 9 90 900 1000 10000 10000	882

input.txt	output.txt
980 1 10 100 1000 900 10000 10000	900

Листинг кода. (именно листинг, а не скрины)

```
with open('input.txt', 'r', encoding='utf-8') as
input_file, open('output.txt', 'w', encoding='utf-8')
as output_file:
    n = int(input_file.readline())
    a = sorted(map(lambda x: [x[0], x[1], x[1] / x[0]],
```

```
                                                [[10  **  i,
int(input_file.readline())]  for  i  in  range(7)]),
key=lambda x: x[2])
   tariff_one, tariff_parts = 2 * 10 ** 9 + 1, 0

   parts_n = n
   for i in a:
       if i[0] > n and i[1] < tariff_one:
           tariff_one = i[1]
           continue
       while parts_n >= i[0]:
           parts_n -= i[0]
           tariff_parts += i[1]
                 output_file.write(str(min(tariff_one,
tariff_parts)))
```

Текстовое объяснение решения.

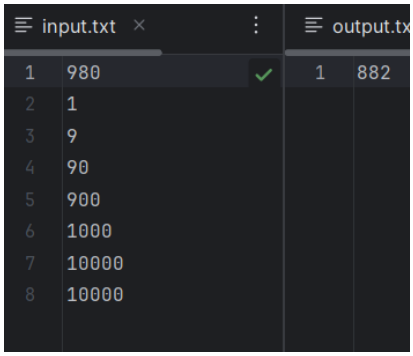
Считываем данные и на их основе создаем список тарифов включающий кол-во листов, стоимоть печать такого кол-ва, удельную стоимость за один лист. Сортируем в порядке возрастания удельной стоимости.

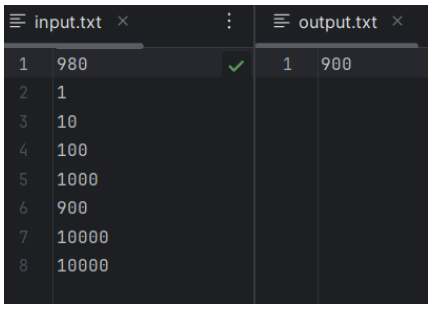
Наиболее выгодными могут оказаться 2 сценария: 1) - один крупный заказ, 2) заказ частями(комбинация тарифов)

Инициализируем начальные значения “тарифа разом” и “тарифа частями”

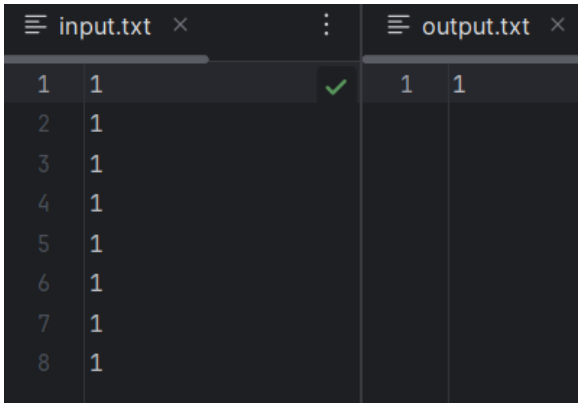
Пробегаем по всем тарифам. Если кол-во листов в тарифе больше необходимого и стоимость меньше стоимости “тарифа разом”: обновляем “тариф разом”. Продолжаем(переходим к вычислению 2 сценария). Пока кол-во листов, которые необходимо распечатать больше чем в тарифе: вычитаем из необходимого кол-ва листы из тарифа и прибавляем стоимость в “тариф частями”. Выбираем наименьшую стоимость печати из “тарифа разом” и “тарифа частями”

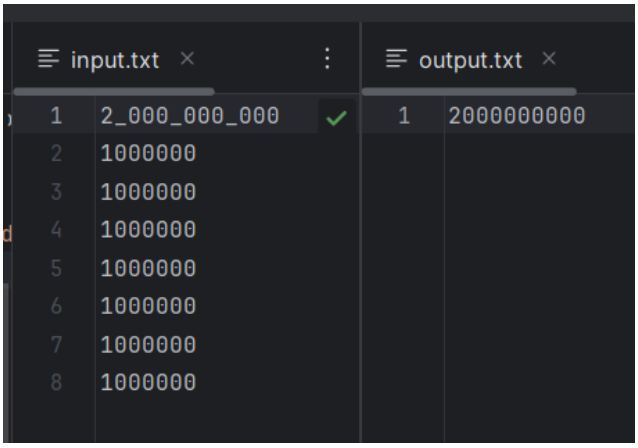
Результат работы кода на примерах из текста задачи:(скрины input output файлов)





Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)





	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0005452	14.94140625
Пример из задачи	0.0004456	14.84375
Пример из задачи	0.0005537	14.6953125

Верхняя граница диапазона значений входных данных из текста задачи	0.00055	14.68359375

Вывод по задаче:

Интересная задача на поиск лучшей стратегии. Отлично подходит для отработки жадных алгоритмов. Я реализовала жадный алгоритм, который выбирает, что более выгодно использовать какой-то один тариф или попытаться собрать комбинацию из предложенных. Здесь снова пригодилась идея сортировки по удельной стоимости. Также мне понравилось, что тесты на максимальных значения очень простые, даже не пришлось генерировать их. :)

Задача №17. Ход конем [2.5 баллов]

Текст задачи:

17 задача. Ход конем (2.5 балла)

- **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.

- Примеры:

input.txt	output.txt
1	8

input.txt	output.txt
2	16

Листинг кода:

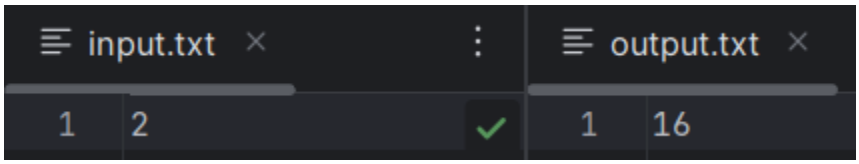
```
def knight_move_number(n):
    mod = 10 ** 9
    move = [0, 1, 1, 1, 1, 1, 1, 1, 0, 1]
    for i in range(n - 1):
        move = [
            (move[4] + move[6]) % mod,
            (move[6] + move[8]) % mod,
            (move[7] + move[9]) % mod,
            (move[4] + move[8]) % mod,
            (move[0] + move[3] + move[9]) % mod,
            0,
            (move[0] + move[1] + move[7]) % mod,
            (move[2] + move[6]) % mod,
            (move[1] + move[3]) % mod,
            (move[2] + move[4]) % mod
```

```
        ]
    return sum(move) % mod


with open("input.txt", "r", encoding='utf-8') as
input_file, open("output.txt", "w", encoding='utf-8')
as output_file:
    n = int(input_file.readline())
    output_file.write(str(knight_move_number(n)))
```

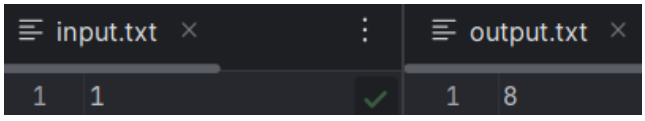
Текстовое объяснение решения:

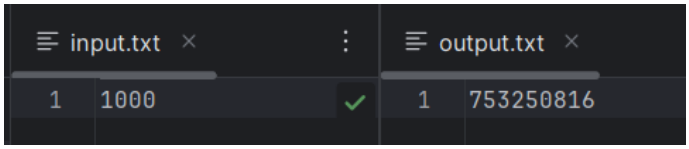
Для решения используем одномерный динамический массив. Инициализируем его с учет того что по условию номер не может начинаться ни с цифры 0, ни с цифры 8. Элементы этого массива показывают сколько телефонных номеров заканчиваются на i где i - индекс элемента в массиве. В цикле для всех длин от 2 до n обновляем массив в зависимость от как ходит конь(в 0 можно попасть из 4 и 6, в 1 из [6, 8],2: [7, 9],3: [4, 8],4: [3, 9, 0],5: [] нельзя попасть ,6: [1, 7, 0],7: [2, 6],8: [1, 3],9: [2, 4]). Суммируя элементы массива мы получаем кол-во возможных номеров.

Результат работы кода на примерах из текста задачи:(скрины input output файлов)



Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)





	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0004055	14.68359375
Пример из задачи	0.0004037	14.81640625
Верхняя граница диапазона значений входных данных из текста задачи	0.0011018	14.703125

Вывод по задаче:

Не особо люблю задачи про шахматы, так что ,увидев название, взгрустнула, но, к счастью, это оказалось не тем, чего я ожидала по названию).

Для этой задачи я использовала разбиение задачи на подзадачи (используя результаты предыдущих вычислений) т.е. динамическое программирование.

Задача №21. Игра в дурака[3 баллов]

Текст задачи:

21 задача. 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.

- Примеры:

input.txt	output.txt
6 2 C KD KC AD 7C AH 9C 6D 6C	YES

input.txt	output.txt
4 1 D 9S KC AH 7D 8D	NO

- Проверить можно по ссылке.

Листинг кода:

```
def function(my_cards, opp_cards, r):
    rank = ['6', '7', '8', '9', 'T', 'J', 'Q', 'K',
'A']
   suit = ['S', 'C', 'D', 'H']

   for card in opp_cards[r]:
       for my_card in my_cards[r]:
           if rank.index(my_card) > rank.index(card):
```

```
                opp_cards[r].remove(card)
                my_cards[r].remove(my_card)
                break
        if opp_cards[r]:
            return False
    suit.remove(r)
    for char in suit:
        for card in opp_cards[char]:
            for my_card in my_cards[char]:
                                if  rank.index(my_card)  >
rank.index(card):
                    opp_cards[char].remove(card)
                    my_cards[char].remove(my_card)
                    break
            if opp_cards[char]:
                for card in opp_cards[char]:
                    if my_cards[r]:
                        opp_cards[char].remove(card)
                        my_cards[r] = my_cards[r][1:]
                    else:
                        return False
            if opp_cards[char]:
                return False

    return True


with   open("INPUT.TXT",   "r",   encoding='utf-8')   as
input_file,  open("OUTPUT.TXT",  "w",  encoding='utf-8')
as output_file:
    my_cards = {'S': [], 'C': [], 'D': [], 'H': []}
    opp_cards = {'S': [], 'C': [], 'D': [], 'H': []}
            n,    m,    r    =    [i    for    i    in
input_file.readline().split()]
```

```
    for my_card_m_r in input_file.readline().split():
        my_cards[my_card_m_r[1]].append(my_card_m_r[0])

    for oppo_card_m_r in input_file.readline().split():

opp_cards[oppo_card_m_r[1]].append(oppo_card_m_r[0])

    output_file.write(('NO', 'YES')[function(my_cards,
opp_cards, r)])
```

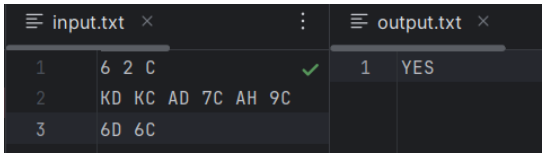
Текстовое объяснение решения:

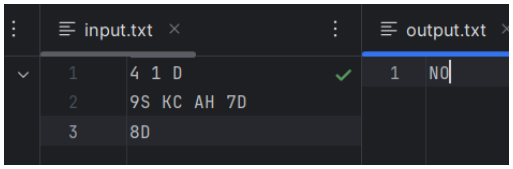
Считываем словарями карты игрока и карты которым оппонент сделал под него ход. Считываем козырь.

Проверяем можно ли отбить карту: первоначально проходимся по козырным картам. Если ранг карты противника меньше нашей - то карта побита, удаляем их из словарей. Если у противника еще остались козырные карты, а у игрока их больше нет - выходим из цикла,возвращая 0, так как отбиться невозможно. Если козырей больше нет у обоих, алалогичным образом проходимся по карта других мастей. Если все карты побиты возвращаем 1.

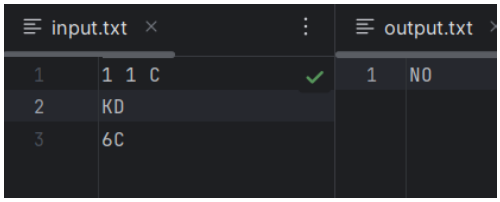
output_file.write(('NO', 'YES')[function(my_cards, opp_cards, r)]) - выводим no или yes в зависимость от того что вернула функция логический 0 или 1.

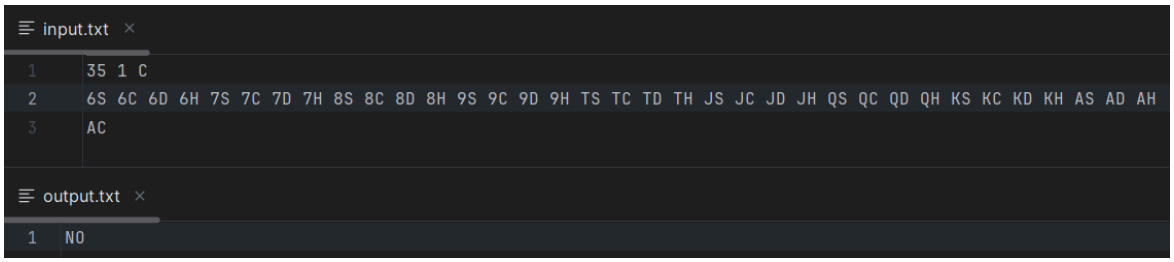
Результат работы кода на примерах из текста задачи:(скрины input output файлов)





Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)





Проверка задачи на (openedu, acmp и тд при наличии в задаче). (скрин)

ID	Дата	Автор	Задача	Язык	Результат	Тест	Время	Память
21848619	29.08.2024 19:44:12	Савченко Анастасія Сергеевна	0698	Python	Accepted		0,046	506 Кб

	Время выполнения,с	Затраты памяти,Мб
Нижняя граница диапазона значений входных данных из текста задачи	0.0004203	15.078125
Пример из задачи	0.000442	14.882812
Пример из задачи	0.0004629	14.8828125
Верхняя граница диапазона значений входных данных из текста задачи	0.0006484	15.0078125

Вывод по задаче:

Для задачи был реализован жадный алгоритм для перебора карт, которые смогут побить карты соперника. Сначала я организовала перебор карт по порядку, но по итогу легче сначала поработать с козырными картами т.к. если у противника их больше победа игрока невозможна.

Дополнительные задачи

Задача №2. Заправки [0.5 баллов]

Текст задачи:

Вы собираетесь поехать в другой город, расположенный в d км от вашего родного города. Ваш автомобиль может проехать не более m км на полном баке, и вы начинаете с полным баком. По пути есть заправочные станции на расстояниях $stop_1, stop_2, ..., stop_n$ из вашего родного города. Какое минимальное количество заправок необходимо?

- **Формат ввода / входного файла (input.txt).** В первой строке содержится d - целое число. Во втройо строке - целое число m. В третьей находится количество заправок на пути - n. И, наконец, в последней строке - целые числа через пробел - остановки $stop_1, stop_2, ..., stop_n$.
- **Ограничения на входные данные.** $1 \leq d \leq 10^5$, $1 \leq m \leq 400$, $1 \leq n \leq 300$, $1 < stop_1 < stop_2 < ... < stop_n < d$
- **Формат вывода / выходного файла (output.txt).** Предполагая, что расстояние между городами составляет d км, автомобиль может проехать не более m км на полном баке, а заправки есть на расстояниях $stop_1, stop_2, ..., stop_n$ по пути, *выведите минимально необходимое количество заправок*. Предположим, что машина начинает ехать с полным баком. Если до места назначения добраться невозможно, выведите -1.
- Ограничение по времени. 2 сек.
- Примеры:

input.txt	output.txt
950 400 4 200 375 550 750	2

В первом примере расстояние между городами 950 км, на полном баке машина может проехать максимум 400 км. Достаточно сделать две заправки: в точках 375 и 750. Это минимальное количество заправок, так как при одной заправке можно проехать не более 800 км.

input.txt	output.txt	input.txt	output.txt
10 3 4 1 2 5 9	-1	200 250 2 100 150	0

Во втором примере до заправки в точке 9 добраться нельзя, так как предыдущая заправка слишком далеко.

В последнем примере нет необходимости заправлять бак, так как автомобиль стартует с полным баком и может проехать 250 км, а расстояние до пункта назначения составляет 200 км.

Листинг кода:

```
def min_refills(x, d, m, n):
    x = [0] + x + [d]
```

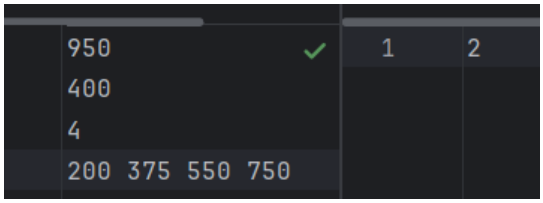
```
    num_refills = 0
    current_refill = 0
    while current_refill <= n:
        last_refill = current_refill

         while current_refill <= n and x[current_refill
+ 1] - x[last_refill] <= m:
            current_refill += 1

        if last_refill == current_refill:
            return -1

        if current_refill <= n:
            num_refills += 1

    return num_refills


with   open('input.txt',   'r',   encoding='utf-8')   as
input_file,  open('output.txt',  'w',  encoding='utf-8')
as output_file:
          d,   m,   n   =   int(input_file.readline()),
int(input_file.readline()), int(input_file.readline())
              stop     =     [int(i)     for     i     in
input_file.readline().split()]
    output_file.write(str(min_refills(stop, d, m, n)))
```

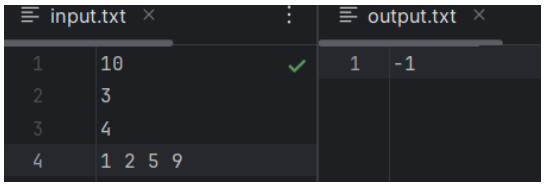
Текстовое объяснение решения.

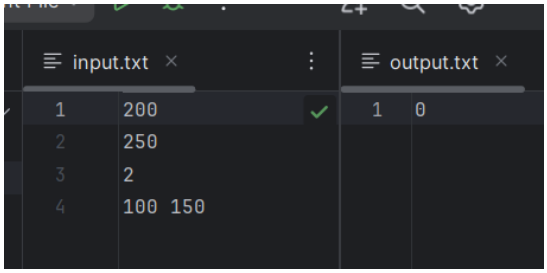
Кол-во заправок, изначально равно 0, текущая заправка, изначально равно 0. x = [0] + x + [d] - добавляем начальную точку 0 и конечную точку d к списку заправок

Пока мы не достигли последней заправки, смотрим до какой наиболее удаленной можем доехать, если нет таких возвращаем -1, иначе едем на самую дальнюю (из возможных - безопасных ход) и увеличиваем счетчик на 1.

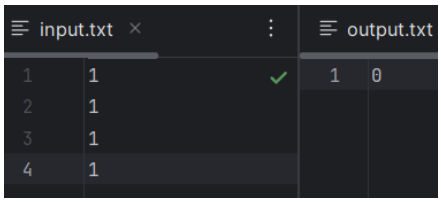
Результат работы кода на примерах из текста задачи:(скрины input output файлов)

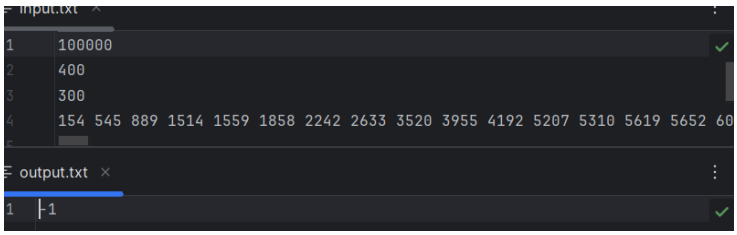






Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)





	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0005169	14.92578125

Пример из задачи	0.0007441	14.9257812
Пример из задачи	0.0005906	14.77734375
Пример из задачи	0.0005291	14.91796875
Верхняя граница диапазона значений входных данных из текста задачи	0.0037579	14.9375

Вывод по задаче:

Для этой задачи я написала простейший жадный алгоритм, с которого начинались наши лекции, он ищет минимальное кол-во заправок, которые может сделать автомобиль и при этом успешно доехать до пункта назначения(безопасных ход).

Задача №5. Максимальное количество призов [0.5 баллов]

Текст задачи.

5 задача. Максимальное количество призов (0.5 балла)

Вы организуете веселый конкурс для детей. В качестве призового фонда у вас есть n конфет. Вы хотели бы использовать эти конфеты для раздачи k лучшим местам в конкурсе с естественным ограничением, заключающимся в том, что чем выше место, тем больше конфет. Чтобы осчастливить как можно больше детей, вам нужно найти наибольшее значение k, для которого это возможно.

- **Постановка задачи.** Необходимо представить заданное натуральное число n в виде суммы как можно большего числа попарно различных натуральных чисел. То есть найти максимальное k такое, что n можно записать как $a_1 + a_2 + ... + a_k$, где $a_1, ..., a_k$ - натуральные числа и $a_i \neq a_j$ для всех $1 \leq i < j \leq k$.
- **Формат ввода / входного файла (input.txt).** Входные данные состоят из одного целого числа n.
- **Ограничения на входные данные.** $1 \leq n \leq 10^9$.
- **Формат вывода / выходного файла (output.txt).** В первой строке выведите максимальное число k такое, что n можно представить в виде суммы k попарно различных натуральных чисел. Во второй строке выведите эти k попарно различных натуральных чисел, которые в сумме дают n (если таких представлений много, выведите любое из них).
- Ограничение по времени. 2 сек.
- Примеры:

№	input.txt	output.txt
1	6	3 1 2 3

№	input.txt	output.txt
2	8	3 1 2 5

№	input.txt	output.txt
3	2	1 2

Листинг кода. (именно листинг, а не скрины)

```
def max_prize(n):
    n = n - 1
    sweets = [1]
    winner = 2
    while n > 0:
        if winner > n:
            sweets[-1] += n
            break
        else:
            sweets.append(winner)
            n -= winner
```

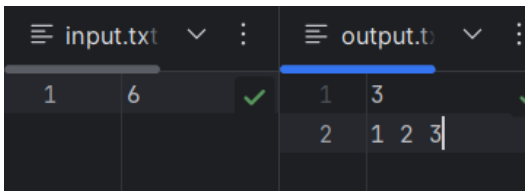
```
            winner += 1
    return sweets


with open('input.txt', 'r', encoding='utf-8') as
input_file, open('output.txt', 'w', encoding='utf-8')
as output_file:
    n = int(input_file.readline())
    ans = [str(i) for i in max_prize(n)]
    output_file.write(f'{len(ans)}\n{" ".join(ans)}')
```

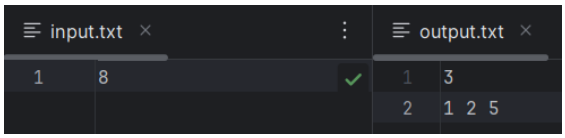
Текстовое объяснение решения:

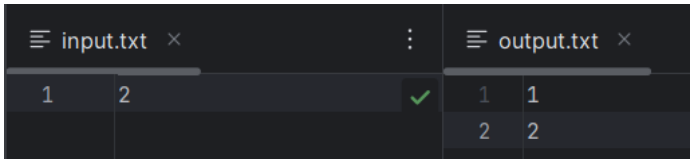
Создаем пустой список sweets, где храним количество конфет для каждого призового места.

Устанавливаем начальное значение на 1 (последнее место получает 1 конфету).В цикле Пока n>0(пока конфеты не кончились), проверяем, можно ли вычесть текущее значение конфет для нового победителя из n и оставить достаточно для следующего места. Если да - добавляем текущее значение в список sweets, уменьшаем n на текущее значение и текущее значение+= 1, если нет, добавляем оставшиеся n конфет в список и выходим из цикла.

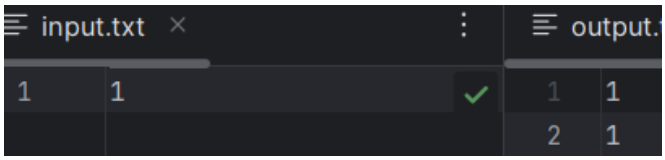
Результат работы кода на примерах из текста задачи:(скрины input output файлов)

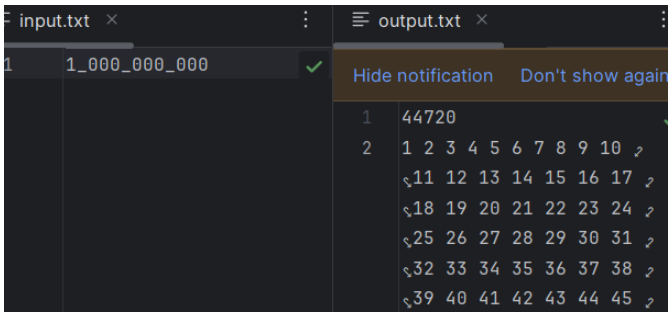






Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)





	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0004049	14.73828125
Пример из задачи	0.0005629	14.9023437
Пример из задачи	0.0004166	14.81640625
Пример из задачи	0.0004748	14.8828125
Верхняя граница диапазона значений входных данных из текста задачи	0.0078725	16.32421875

Вывод по задаче:

Если приглядеться, можно увидеть, что эта задача решает не только проблему детских утренников, но определяет на какое максимальное кол-во слагаемых можно разложить число и как это будет выглядеть.

Задача №6. Максимальная зарплата[0.5 баллов]

Текст задачи.

6 задача. Максимальная зарплата (0.5 балла)

В качестве последнего вопроса успешного собеседования ваш начальник дает 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?

На лекциях мы рассмотрели следующий алгоритм составления наибольшего числа из заданных *однозначных* чисел.

```
def LargestNumber(Digits):
    answer = ''
    while Digits:
        maxDigit = float('-inf')
        for digit in Digits:
            if digit >= maxDigit:
                maxDigit = digit
        answer += str(maxDigit)
        Digits.remove(maxDigit)
    return answer
```

К сожалению, этот алгоритм работает только в том случае, если вход состоит из однозначных чисел. Например, для ввода, состоящего из двух целых чисел 23 и 3 (23 не однозначное число!) возвращается 233, в то время как наибольшее число на самом деле равно 323. Другими словами, использование наибольшего числа из входных данных в качестве первого числа *не является безопасным ходом*.

Ваша цель в этой задаче – настроить описанный выше алгоритм так, чтобы он работал не только с однозначными числами, но и с произвольными положительными целыми числами.

- **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 $a_1, a_2, ..., a_n$.
- **Ограничения на входные данные.** $1 \leq n \leq 10^2$, $1 \leq a_i \leq 10^3$ для всех $1 \leq i \leq n$.
- **Формат вывода / выходного файла (output.txt).** Выведите наибольшее число, которое можно составить из $a_1, a_2, ..., a_n$.
- Ограничение по времени. 2 сек.
- Пример:

input.txt	output.txt
2 21 2	221

input.txt	output.txt
3 23 39 92	923923

Листинг кода.

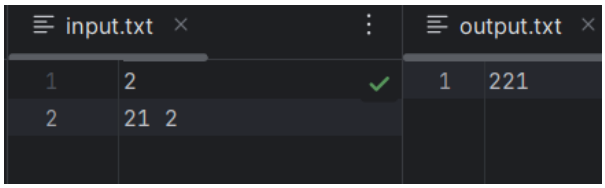
```
def max_salary(x1, x2):
    if int(f'{x1}{x2}') > int(f'{x2}{x1}'):
        return True
    return False
```

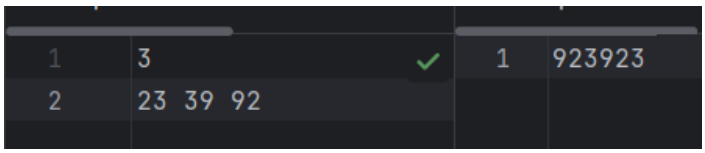
```
def f_sort(l, n):
    for i in range(n - 1):
        for j in range(n - i - 1):
            if max_salary(l[j + 1], l[j]):
                l[j + 1], l[j] = l[j], l[j + 1]
    return l


with open('input.txt', 'r', encoding='utf-8') as
input_file, open('output.txt', 'w',

encoding='utf-8') as output_file:
    n = int(input_file.readline())
    l = f_sort(input_file.readline().split(), n)
    ans = ''.join(l)
    output_file.write(ans)
```

Текстовое объяснение решения.

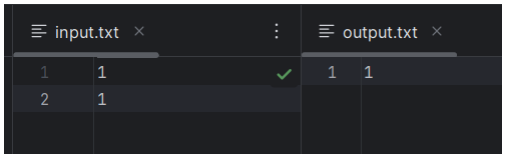
Считываем числа. Сортируем числа для создания наибольшего возможного числа из их комбинаций. Используем пузырьковую сортировку для упорядочивания списка l.

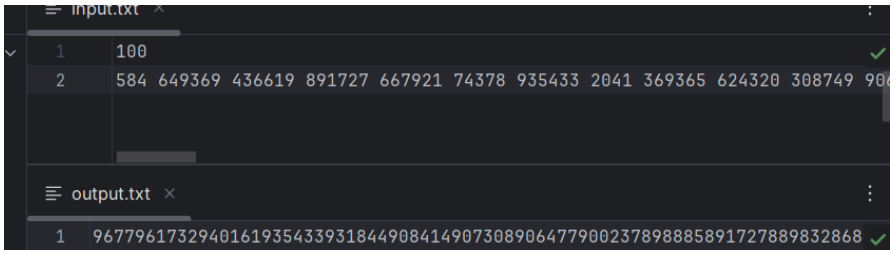
Результат работы кода на примерах из текста задачи:(скрины input output файлов)





Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)





	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0029878	14.65234375
Пример из задачи	0.0004361	14.68359375
Пример из задачи	0.0042229	14.73828125
Верхняя граница диапазона значений входных данных из текста задачи	0.0056099	14.84765625

Вывод по задаче:

Я решила задачу о максимизации числа из набора чисел. Для 100 штук заданных чисел алгоритм весьма простой. Задача является улучшенной версией, той о которой говорили на лекции, т.к. вместо цифр подаются числа

Задача №8. Расписание лекций[1 баллов]

Текст задачи.

8 задача. Расписание лекций (1 балл)

- **Постановка задачи.** Вы наверно знаете, что в ИТМО лекции читают одни из лучших преподаватели мира. К сожалению, лекционных аудиторий у нас не так уж и много, особенно на Биржевой, поэтому каждый преподаватель составил список лекций, которые он хочет прочитать студентам. Чтобы студенты, в начале февраля, увидели расписание лекций, необходимо его составить прямо сейчас. И без вас нам здесь не справиться. У нас есть список заявок от преподавателей на лекции для одной из аудиторий. Каждая заявка представлена в виде временного интервала $[\ s_i, f_i\ )$ - время начала и конца лекции. Лекция считается открытым интервалом, то есть какая-то лекция может начаться в момент окончания другой, без перерыва. Необходимо выбрать из этих заявок такое подмножество, чтобы суммарно выполнить максимальное количество заявок. Учтите, что одновременно в лекционной аудитории, конечно же, может читаться лишь одна лекция.
- **Формат ввода / входного файла (input.txt).** В первой строке вводится натуральное число N - общее количество заявок на лекции. Затем вводится N строк с описаниями заявок - по два числа в каждом s_i и f_i для каждой лекции i. Гарантируется, что $s_i < f_i$. Время начала и окончания лекции - натуральные числа, не превышают 1440 (в минутах с начала суток).
- **Ограничения на входные данные.** $1 \le N \le 1000, 1 \le s_i, f_i \le 1440$
- **Формат вывода / выходного файла (output.txt).** Выведите одно число – максимальное количество заявок на проведение лекций, которые можно выполнить.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Примеры:

input.txt	output.txt
1 5 10	1

input.txt	output.txt
3 1 5 2 3 3 4	2

Листинг кода. (именно листинг, а не скрины)

```
with open('input.txt', 'r', encoding='utf-8') as
input_file, open('output.txt', 'w', encoding='utf-8')
as output_file:
    n = int(input_file.readline())
        lects = sorted([[int(i) for i in
input_file.readline().split()] for _ in range(n)],
key=lambda x: x[1])
    # print(lects) -- сортируем по времени окончания
    counter = 1
```

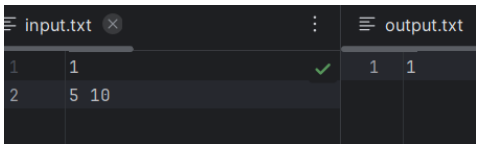
```
    current_lec = lects[0]
    for i in lects[1:]:
        if i[0] >= current_lec[1]:
            current_lec = i
            counter += 1
    output_file.write(str(counter))
```

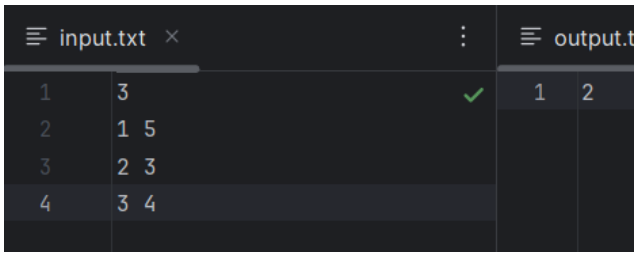
Текстовое объяснение решения.

Считываем время начала и окончания лекций и сортируем по времени окончания в порядке возрастания.

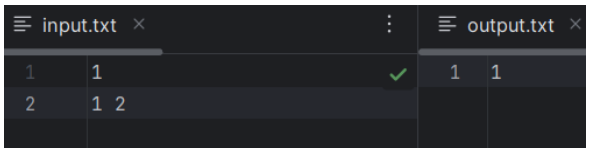
В цикле идем по отсортированному массиву лекций для каждой из них проверяем может ли она начаться после текущей. Обновляем текущую лекцию и прибавляем 1 к итоговому счетчику.

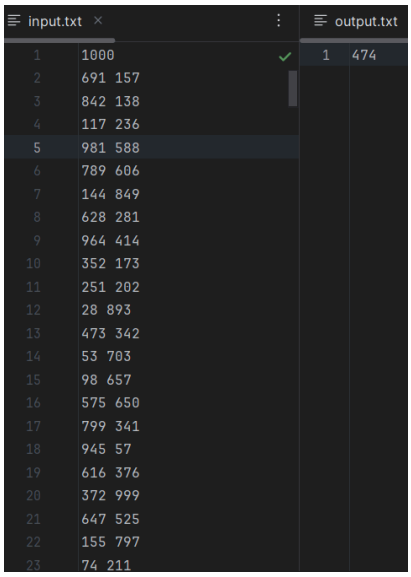
Результат работы кода на примерах из текста задачи:(скрины input output файлов)





Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)





	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0003849	14.890625
Пример из задачи	0.0008905	14.8984375
Пример из задачи	0.0004372	14.6640625
Верхняя граница диапазона значений входных данных из текста задачи	0.0092157	15.03125

Вывод по задаче:

Задача легко решается, когда понимаешь идею про временные отрезки.

Задача №11. Максимальное количество золота [1 баллов]

Текст задачи:

11 задача. Максимальное количество золота (1 балл)

Вам дается набор золотых слитков, и ваша цель - набрать как можно больше золота в свою сумку. Существует только одна копия каждого слитка, и для каждого слитка вы можете либо взять его, либо нет (т.е. вы не можете взять часть слитка).

- **Постановка задачи.** Даны n золотых слитков, найдите максимальный вес золота, который поместится в сумку вместимостью W.
- **Формат ввода / входного файла (input.txt).** Первая строка входных данных содержит вместимость W сумки и количество n золотых слитков. В следующей строке записано n целых чисел $w_0, w_1, ..., w_{n-1}$, определяющие вес золотых слитков.
- **Ограничения на входные данные.** $1 \leq W \leq 10^4$, $1 \leq n \leq 300$, $0 \leq w_0, ..., w_{n-1} \leq 10^5$
- **Формат вывода / выходного файла (output.txt).** Выведите максимальный вес золота, который поместится в сумку вместимости W.
- Ограничение по времени. 5 сек.
- Пример:

input.txt	output.txt
10 3 1 4 8	9

Здесь сумма весов первого и последнего слитка равна 9.

- Обратите внимание, что в этой задаче все предметы имеют одинаковую стоимость на единицу веса по простой причине: все они сделаны из золота.

Листинг кода. (именно листинг, а не скрины):

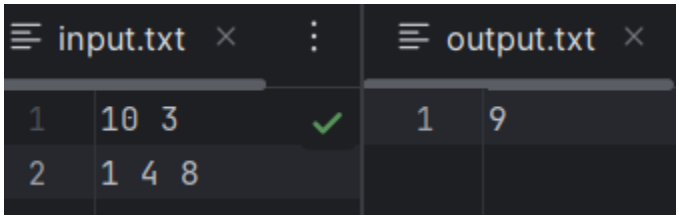
```
def knapsack(w, wi, n):
    value = [[0] * (w + 1) for _ in range(n + 1)]
    for i in range(1, n + 1):
        for j in range(1, w + 1):
            if j < wi[i - 1]:
                value[i][j] = value[i - 1][j]
            else:
                    value[i][j] = max(value[i - 1][j],
value[i - 1][j - wi[i - 1]] + wi[i - 1])
    return value[-1][-1]
```

```
with open('input.txt', 'r', encoding='utf-8') as
input_file, open('output.txt', 'w', encoding='utf-8')
as output_file:
    w, n = map(int, input_file.readline().split())
                wi      =      [int(i)      for      i      in
input_file.readline().split()]
    output_file.write(str(knapsack(w, wi, n)))
```

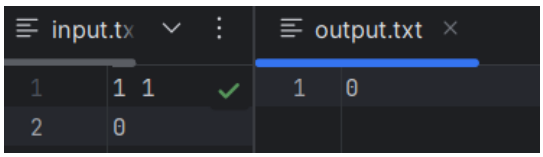
Текстовое объяснение решения:

Создаем таблицу value, заполняя ее нулями. Во внешнем цикле идем по всем элементам, во внутреннем - по всем весом рюкзака. Если текущий вес рюкзака меньше веса предмета - взять его мы не можем, поэтому записываем в таблицу значение из строки выше(такое же как и для предыдущего предмета). Если же предмет мы взять можем, нужно выбрать максимум из двух вариантов не брать предмет, взять предмет. Таким образом заполнив таблицу последний элемент будет самым большим, возвращаем его.

Результат работы кода на примерах из текста задачи:(скрины input output файлов):



Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений	0.0004283	14.8515625

входных данных из текста задачи		
Пример из задачи	0.000547	14.8828125
Верхняя граница диапазона значений входных данных из текста задачи	0.238841	16.5625

Вывод по задаче:

Задача про рюкзак без повторений, где вес предмета равняется его стоимости. Я реализовала алгоритм на основе таблицы, что делает его быстрее итеративного алгоритма с одномерным массивом.

Задача №22. Симпатичные узоры[4 баллов]

Текст задачи:

22 задача. 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.

- Примеры:

input.txt	output.txt	input.txt	output.txt
2 2	14	3 3	322

- Проверить можно по ссылке.

Листинг кода:

```
def checker(x, y, n):
    cell = [0] * 5

    for i in range(n - 1):
        cell[1] = 1 if (x // (2 ** i)) % 2 != 0 else 0
```

```
        cell[2] = 1 if (x // (2 ** (i + 1))) % 2 != 0
else 0
        cell[3] = 1 if (y // (2 ** i)) % 2 != 0 else 0
        cell[4] = 1 if (y // (2 ** (i + 1))) % 2 != 0
else 0
        print(cell[1])
        if cell[1] == cell[2] == cell[3] == cell[4]:
            return False

    return True

def counter(n, m):
    res = 0
    length = 2 ** n  # количество всех возможных чисел
длины n бит
    matrix = [[0] * length for _ in range(length)]
    s_dp = [[0] * length for _ in range(m)]

    for i in range(length):
        for j in range(length):
            if checker(i, j, n):
                matrix[i][j] = 1
            else:
                matrix[i][j] = 0

    for i in range(length):
        s_dp[0][i] = 1

    for k in range(1, m):
        for i in range(length):
            for j in range(length):
                        s_dp[k][i] += s_dp[k - 1][j] *
matrix[j][i]

    for i in range(length):
```

```
            res += s_dp[m - 1][i]

    return res


with open("INPUT.TXT", "r", encoding='utf-8') as
input_file, open("OUTPUT.TXT", "w", encoding='utf-8')
as output_file:
    n, m = sorted([int(i) for i in
                   input_file.readline().split()])  #
чтоб точно знать какая на каком месте, из-за прохода
циклов помогает уменьшить затраты памяти
    output_file.write(str(counter(n, m)))
```

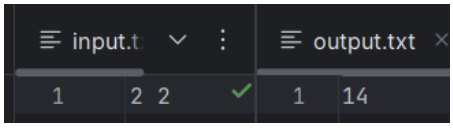
Текстовое объяснение решения:

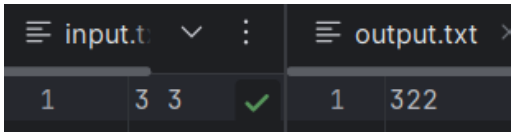
В функции checker проверяем является ли узор симпатичным, , то есть не содержит ли он квадрата 2×2. В цикле определяем какой окрас у каждой из клеток. Если у них одинаковый окрас(cell[1] == cell[2] == cell[3] == cell[4]) - возвращаем False узор не симпатичный, иначе True, что значит узор симпатичный.

Будем использовать checker в функции def counter для проверки всех возможных комбинаций узоров и подсчета количества симпатичных.

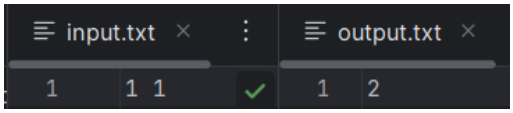
def counter: заводим res = 0 для подсчета результата. Заполняем матрицу: для всех пар чисел (i, j) проверяется checker является ли узор симпатичным. На первом шаге инициализируем все значения динам. теблицы s_dp[0][i] = 1 т.к. на первом шаге все узоры считаются симпатичными. Для каждой строки k от 1 до m-1 и для каждой пары чисел (i, j) обновляется значение s_dp[k][i]. Суммируем все значения последней строки таблицы s_dp[m-1][i], чтобы получить итоговое количество симпатичных узоров.

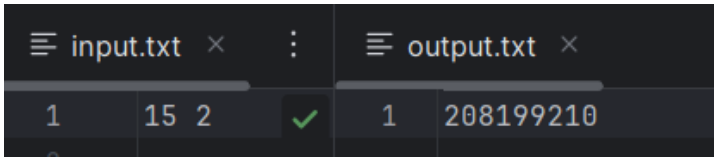
Результат работы кода на примерах из текста задачи:(скрины input output файлов)





Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)





Проверка задачи на (openedu, acmp и тд при наличии в задаче). (скрин)

ID	Дата	Автор	Задача	Язык	Результат	Тест	Время	Память
21848751	29.08.2024 20:41:30	Савченко Анастасия Сергеевна	0083	Python	Accepted		0.046	498 Кб

	Время выполнения, с	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи	0.0033866	15.078125
Пример из задачи	0.0004642	15.0546875
Пример из задачи	0.0010916	14.83984375
Верхняя граница диапазона значений входных данных из текста задачи	0.000446	15.109375

Вывод по задаче:

Был реализован в какой-то степени переборный алгоритм проверки всех возможных комбинаций узоров и подсчета количества симпатичных из них.

Задача №14. Максимальное значение арифметического выражения [2 баллов]

Текст задачи:

14 задача. Максимальное значение арифметического выражения (2 балла)

В этой задаче ваша цель - добавить скобки к заданному арифметическому выражению, чтобы максимизировать его значение.

$$max(5 - 8 + 7 \times 4 - 8 + 9) = ?$$

- **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 $s_0, s_1, ..., s_{2n}$. 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.** $0 \leq n \leq 14$ (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.
- Пример:

input.txt	output.txt	input.txt	output.txt
1+5	6	5-8+7*4-8+9	200

Здесь $200 = (5 - ((8 + 7) * (4 - (8 + 9))))$.

Листинг кода:

```
def MinAndMax(i, j, M, m, op):
    minimum = float("inf")
    maximum = float("-inf")
    for k in range(i, j):
        a = eval(f"{M[i][k]} {op[k]} {M[k + 1][j]}")
            # {op[k]} - на место {op[k]} встает один из
знаков например - или *,etcли op[k] равно +, то код
выполнит сложение
            # динамически создает и выполняет выражение,
исп. элементы матрицы и оператор из списка op
        b = eval(f"{M[i][k]} {op[k]} {m[k + 1][j]}")
```

```
        c = eval(f"{m[i][k]} {op[k]} {M[k + 1][j]}")
        d = eval(f"{m[i][k]} {op[k]} {m[k + 1][j]}")
        minimum = min(minimum, a, b, c, d)
        maximum = max(maximum, a, b, c, d)
    return minimum, maximum


def maxValue(d, op):
    n = len(d)
    m = [[0] * n for _ in range(n)]
    M = [[0] * n for _ in range(n)]

    # проходим по диагонали
    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]


with open("input.txt", "r", encoding='utf-8') as
input_file, open("output.txt", "w", encoding='utf-8')
as output_file:
    s = input_file.read().rstrip()
    digits = list(map(int, s[::2]))
    operators = list(s[1::2])

    output_file.write(str(maxValue(digits, operators)))
```

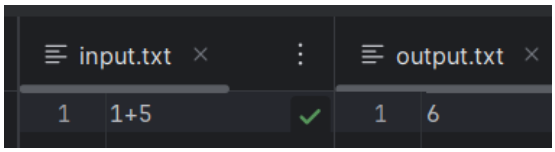
Текстовое объяснение решения:

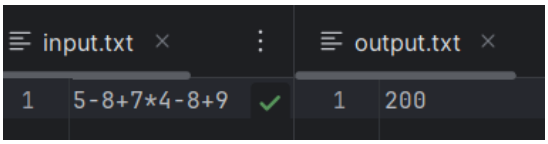
В отдельные списки считываем цифры и матем. Операторы (- + *).

По идее динам. Прог. Будем разбивать задачу на подзадачи. Так, например, предположив что последняя операция *, для вычисления выражения (5-8+7)*(4-8+9), нужно рассчитать оптимальные значения для 5-8+7 и 4-8+9. Будем отслеживать как минимальные так и макс. Знач. подвыражений(чтобы учесть случаи с отрицательными числами вель - * - = +).

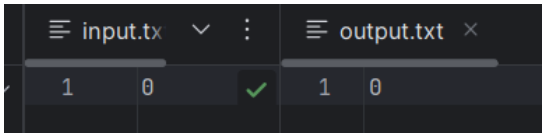
def MinAndMax: Функция из лекции, чтобы найти оптимальное значения подвыражений нужно рассмотреть все варианты мин-макс, макс-макс,мин-мин ,макс-мин и так для каждого оператора (- + *). Максимальные и минимальные значения подвыражений будем записывать в матрицы.

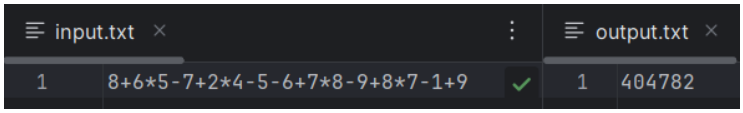
def maxValue: По диагоналям заполняем матрицы m(мин.) и M(макс.) выше главной диагонали. Рассм. а примере 5-8+7*4-8+9. На глав диагонали стоят сами числа из выражения, элемент (1, 1)= 5 , соответствует скобкам расставленным (5) -8+7*4-8+9, далее заполняем остальные элементы, элемент (1,2) = -3 соотв. (5-8)+7*4-8+9, элемент (2,3) = 15 соот. 5-(8+7)*4-8+9. Ячейка (1,3) можно получить 2 способами (1,1) - (2,3)=-10 или (1,2)+(3,3)=4, -10 запишется в матрицу m, а 4 в M. так идем до “углового элемента”. Так как в матрице сохраняются вычисленные значения их мы используем для вычисления следующих значений дабы не пересчитывать лишний раз. Это уменьшает сложность алгоритма функции maxValue с O(n!) до ~ O(n^2).
Общая сложность алгоритма ~ O(n^3).
Результат работы кода на примерах из текста задачи:





Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)





	Время выполнения,с	Затраты памяти,Мб
Нижняя граница диапазона значений входных данных из текста задачи	0.00039419998	14.984375
Пример из задачи	0.0007590000	14.8398437
Пример из задачи	0.001261100	14.984375
Верхняя граница диапазона значений входных данных из текста задачи	0.0122546999	15.0351562

Вывод по задаче:

Для этой задачи я реализовала жадный алгоритм для максимизации значения выражения, заданного без скобок, путем расставления скобок в лучших местах.

Вывод

В ходе лабораторной работы я научилась решать разные виды задачи о рюкзаке. Так, для задача о дробном рюкзаке, необходимо использовать сортировку и жадный алгоритм, а для классического рюкзака лучше всего подходит динамическое программирование. Я также узнала, что такое безопасный ход и динамические таблицы, а также развила навык решения задач через разбиение на подзадачи.

Все написанные алгоритмы были успешно протестированы как на примерах из задач, так и на максимальных значениях. То, что алгоритм успешно справляется с большими входными данными, доказывает, что он действительно эффективен для решения поставленной задачи.