

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

Отчет по лабораторной работе №2.1
по курсу «Алгоритмы и структуры данных»
Тема: Жадные алгоритмы. Динамическое
программирование №2
Вариант 2

Выполнил:

Вилис З.М.

13.3

Проверила:

Артамонова В.Е.

Санкт-Петербург

2024 г.

Содержание отчета

Содержание отчета	2
Задачи по варианту	3
Задание №3. Максимальный доход от рекламы [N баллов]	3
Задание №5. Максимальное количество призов [N баллов]	6
Задание №10. Яблоки[N баллов]	9
Задание №14. Максимальное значение арифметического выражения [N баллов]	11
Задание №18. Кафе[N баллов]	11
Дополнительные задачи	15
Задание №17. Ход конем[2,5 баллов]	15
Задание №2. Заправки[0,5 баллов]	17
Задание №13. Сувениры [1,5 баллов]	22
Задание №4. Сбор подписей [0,5 баллов]	26
Вывод по лабораторной работе	27

Задачи по варианту

Задача №3. Максимальный доход от рекламы [N баллов]

Даны две последовательности a_1, a_2, \dots, a_n (a_i - прибыль за клик по i -му объявлению) и b_1, b_2, \dots, b_n (b_i - среднее количество кликов в день i -го слота), нужно разбить их на n пар (a_i, b_j) так, чтобы сумма их произведений была максимальной.

Листинг кода:

```
with open('input.txt', 'r') as input_file:
    n = int(input_file.readline().strip())
    profit = list(map(int, input_file.readline().split()))
    clicks = list(map(int, input_file.readline().split()))

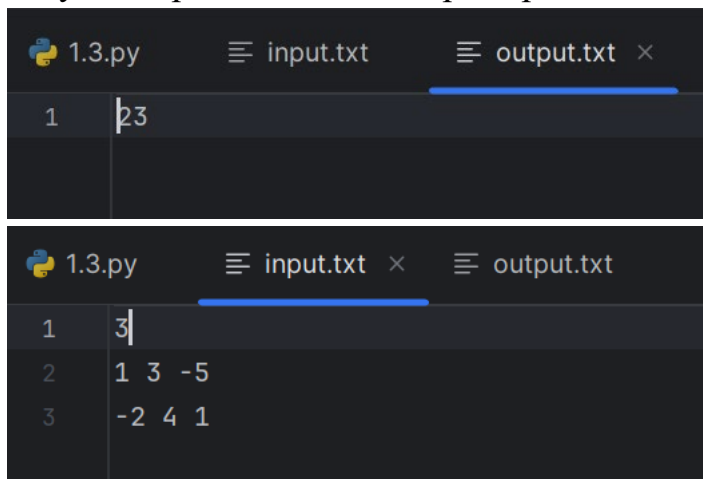
profit.sort(reverse=True)
clicks.sort(reverse=True)

max_profit = str(sum([profit[i] * clicks[i] for i in range(n)]))

with open('output.txt', 'w') as output_file:
    output_file.write(max_profit)
```

Текстовое объяснение решения: программа ищет способ максимизировать общую прибыль, сопоставляя прибыль и количество кликов для n элементов. Сначала она сортирует оба списка по убыванию, чтобы наибольшие прибыли были перемножены с наибольшим количеством кликов, что позволяет достичь максимального значения суммарной прибыли.

Результат работы кода на примерах из текста задачи:



```

1.3.py  input.txt  output.txt x
1 397

```

```

1.3.py  input.txt x  output.txt
1 1
2 23
3 39

```

Результат работы кода на максимальных и минимальных значениях:

```

1.3.py x  input.txt  output.txt x
1 1

```

```

1.3.py  input.txt x  output.txt
1 1
2 1
3 1

```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи(1 элемент)	0.000667810440063476 6	22.8
Пример из задачи	0.000486135482788085 94	23.2
Пример из задачи	0.000667810440063476 6	22.8

Вывод по задаче: использовала новые методы решения.

Задача №5. Максимальное количество призов [N баллов]

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 $a_1 + a_2 + \dots + a_k$, где a_1, \dots, a_k - натуральные числа и $a_i \neq a_j$ для всех $1 \leq i < j \leq k$.

Листинг кода:

```
with open('input.txt', 'r') as f:
    n = int(f.readline().strip())

def max_pairwise_sum(n):
    k = 1
    while k * (k + 1) <= 2 * n:
        k += 1
    k -= 1

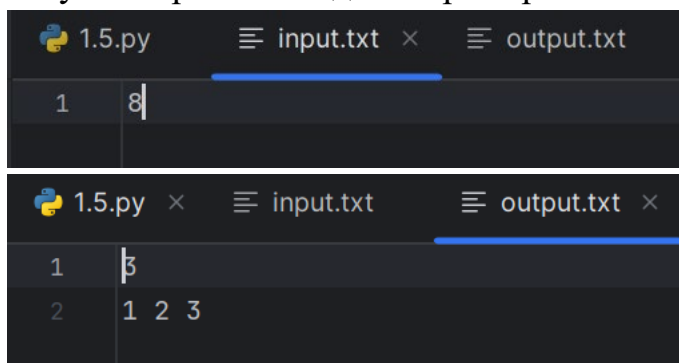
    result = []
    for i in range(1, k):
        result.append(i)
        n -= i
    result.append(n)

    return k, result

with open('output.txt', 'w') as f:
    k, result = max_pairwise_sum(n)
    f.write(str(k) + '\n')
    f.write(' '.join(map(str, result)))
```

Текстовое объяснение решения: Этот код позволяет вычислить максимальное количество различных натуральных чисел, сумма которых не превышает заданное значение n , а затем записывает результаты в файл.

Результат работы кода на примерах из текста задачи:



```
1.5.py  input.txt  output.txt
1      8

1.5.py  input.txt  output.txt
1      3
2      1 2 3
```

```
1.5.py input.txt output.txt
1 6
```

```
1.5.py input.txt output.txt 1.3.py
1 |
2 1 2 5
```

Результат работы кода на максимальных и минимальных значениях:

```
1.5.py input.txt output.txt
1 1
```

```
1.5.py input.txt output.txt
1 |
2 1
```

Проверка задачи на (асгр и тд при наличии в задаче):

	Время выполнения, с	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи	0.007384727467283623	22.8
Пример из задачи	0.007263723138237237	23.2
Пример из задачи	0.008932398239102339	23.2

Вывод по задаче: интересная математика.

Задача №10. Яблоки [N баллов]

Текст задачи. Алисе в стране чудес попались n волшебных яблок.

Про каждое яблоко известно, что после того, как его съешь, твой рост сначала уменьшится на a_i сантиметров, а потом увеличится на b_i сантиметров.

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

Листинг кода:

```
f = open('input.txt')
s = f.readline().split()
apples, rost = int(s[0]), int(s[1])
z_apples = []
for i in range(apples):
    s2 = f.readline().split()
    x = [int(i) for i in s2]
    z_apples.append(x)

def funny_apples(A, h):
    m = A.copy()
    rez = ''
    while A:
        for i in A:
            if h-i[0] > 0:
                h = h + i[1] - i[0]
                A.remove(i)
                r0 = m.index(i) + 1
                rez = rez + str(r0) + ' '
                break
        elif A.index(i) == len(A)-1:
            return '-1'
    return rez

z = open('output.txt', 'w')
z.write(funny_apples(z_apples, rost))
```

Текстовое объяснение решения: создается копия списка, содержащего характеристики каждого яблока. Пока в списке с яблоками имеются яблоки выполняется следующее: проверяется, можно ли взять яблоко и не уйдет ли рост Алисы в ноль или отрицательное значение. Если можно, к росту Алисы добавляется плюс к росту от яблока и вычитается минус к росту от яблока. Далее это яблоко удаляется из списка. Далее в результирующую строку записывается индекс данного яблока из скопированного списка.

Если же случается такая ситуация, что наше рассматриваемое яблоко последнее в списке и при этом не выполняется условие о росте, возвращается значение -1, дающее нам понять, что Алиса не может съесть все эти яблоки. Иначе, когда значения в списке A заканчиваются, мы возвращаем результирующую строку.

Результат работы кода на примерах из текста задачи:

3 5	1 3 2
2 3	
10 5	
5 10	

3 5	-1
2 3	
10 5	
5 6	

Результат работы кода на максимальных и минимальных значениях:

1 1	-1
1 1	

1000 1000	1 2 3 4 5 8 9 6 7
101 585	
397 339	
275 614	

Проверка задачи на (астр и тд при наличии в задаче):

	Время выполнения, с	Затраты памяти, Мб
--	---------------------	--------------------

Нижняя граница диапазона значений входных данных из текста задачи(1 элемент)	0.0007779598236083984	22.0
Пример из задачи	0.0008001327514648438	22.1
Пример из задачи	0.0008869171142578125	22.3
Верхняя граница диапазона значений входных данных из текста задачи(100 элементов)	0.012340307235717773	22.3

Вывод по задаче: люблю есть яблоки

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

Текст задачи. В этой задаче ваша цель - добавить скобки к заданному арифметическому

выражению, чтобы максимизировать его значение.

$\max(5 - 8 + 7 \times 4 - 8 + 9) = ?$ Найдите максимальное значение арифметического выражения, указав порядок применения его арифметических операций с помощью дополнительных скобок.

Листинг кода:

```
f = open('input.txt')
s = f.readline()
n = []
operations = []
for i in range(len(s)):
    if i % 2 == 0:
        n.append(int(s[i]))
    else:
        operations.append(s[i])

def Maxx(op, m1, m2):
```

```

if op == '-':
    return m1-m2
elif op == '+':
    return m1 + m2
elif op == '*':
    return m1 * m2

def max_itog(op, ch, nt):
    Max = [[0]*(nt) for i in range(nt)]
    Min = [[0]*(nt) for i in range(nt)]
    for i in range(nt):
        Max[i][i] = ch[i]
        Min[i][i] = ch[i]
    for i in range(nt-1):
        for s in range(nt-1):
            b = []
            for k in range(s, i+1+s):
                b.append(Maxx(op[k], Max[s][k], Max[k + 1][s + 1 + i]))
                b.append(Minx(op[k], Min[s][k], Min[k + 1][s + 1 + i]))
            Min[s][i + 1 + s] = min(b)
            Max[s][i + 1 + s] = max(b)
    nt-=1
    return Max[0][len(Max)-1]

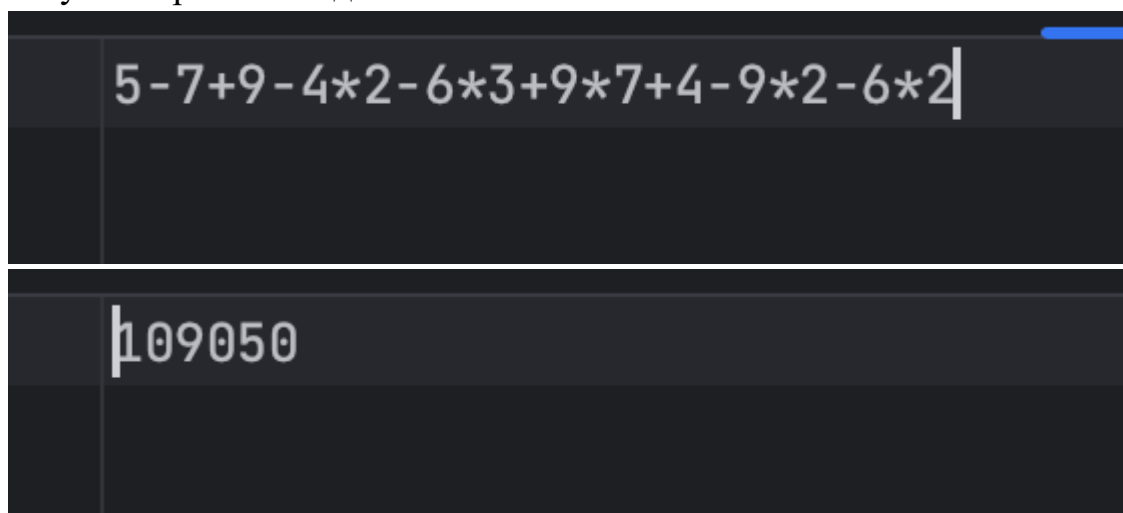
```

Текстовое объяснение решения: сначала из файла считываются данные в два отдельных списка, в один числа, а в другой знаки. Далее в функции max_itog создаются два массива(Max и Min) размером количества чисел. Диагонали каждого массива заполняются числами. Далее рассматриваются ячейки в верхнем треугольнике матрицы. В пустой список добавляются все вариации расстановки скобок с текущими числами. Далее в список Min на место этой ячейки ставится минимальный элемент из списка, а в список Max максимальный.

Результат работы кода на примерах из текста задачи:

1+5	6
5-8+7*4-8+9	200

Результат работы кода на максимальных и минимальных значениях:



Проверка задачи на (астр и тд при наличии в задаче):

	Время выполнения, с	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи(1 элемент)	0.0007779598236083984	22.0
Пример из задачи	0.0008001327514648438	22.1
Пример из задачи	0.0008869171142578125	22.3
Верхняя граница диапазона значений входных данных из текста задачи(100 элементов)	0.0007307529449462891	22.3

Вывод по задаче: поработала с функциями.

Задача №18. Кафе [N баллов]

Текст задачи. Около университета недавно открылось новое кафе, в котором действует следующая система скидок: при каждой покупке бо-

лее чем на 100 рублей покупатель получает купон, дающий право на один бесплатный обед (при покупке на сумму 100 рублей и меньше такой купон покупатель не получает). Однажды вам на глаза попался прејскурант на ближайшие n дней. Внимательно его изучив, вы решили, что будете обедать в этом кафе все n дней, причем каждый день вы будете покупать в кафе ровно один обед. Однако стипендия у вас небольшая, и поэтому вы хотите по максимуму использовать предоставляемую систему скидок так, чтобы ваши суммарные затраты были минимальны. Требуется найти минимально возможную суммарную стоимость

Листинг кода:

```
f = open('input.txt')
days = int(f.readline())
lunches = []
for i in range(days):
    a = int(f.readline())
    lunches.append(a)

def min_cost(l, d):
    dp = [[500]*(d+1) for _ in range(d+1)]
    dp[0][0] = 0
    for i in range(1, d+1):
        for j in range(i):
            dp[i][j] = min(dp[i][j], dp[i-1][j] + l[i-1])
            if l[i-1] > 100:
                dp[i][j+1] = min(dp[i][j+1], dp[i-1][j] + l[i-1])
            if j >= 1:
                dp[i][j-1] = min(dp[i][j-1], dp[i-1][j])
    m = min(dp[d])
    mm = m
    not_used = dp[d].index(m)
    kyponi = []
    for i in range(d-1, 0, -1):
        for j in range(d):
            if m == (dp[i][j] + l[i]):
                m = dp[i][j]
                break
            elif m == dp[i][j]:
                kyponi.append(i+1)
                break
    print(dp)
    print(l)
    return (mm, kyponi, not_used)

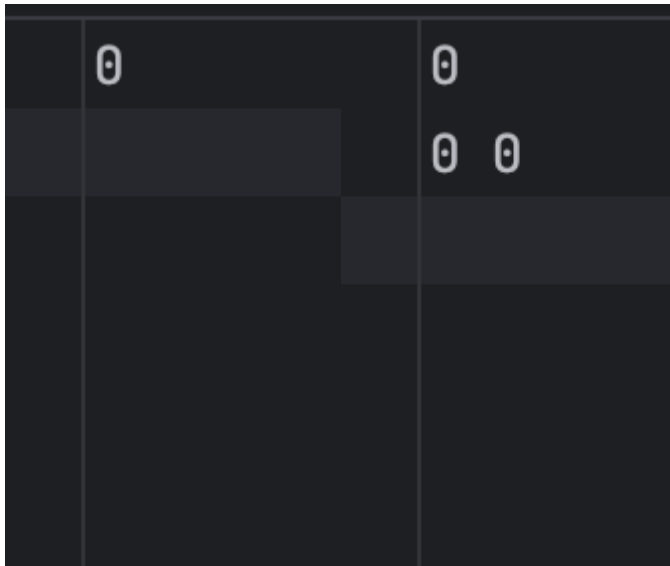
p = min_cost(lunches, days)
z = open('output.txt', 'w')
z.write(str(p[0]) + '\n')
z.write(str(p[2]) + ' ' + str(len(p[1])) + '\n')
p[1].reverse()
for i in p[1]:
    z.write(str(i) + '\n')
```

Текстовое объяснение решения: Цены считываются в список lunches. Функция min_cost выполняет следующее: массив dp заполняется большими числами по количеству дней, которые студент будет есть. Нулевой элемент приравнивается нулю. Запускается цикл. В нем i обозначает день, а j обозначает количество купонов. На ячейку с индексами i и j записывается минимальное значение из текущего в ячейке и предыдущего в ячейке + цена обеда. Если цена больше 100, то в ячейку с индексами i и j+1 записывается минимальное значение из текущего и предыдущего + цена обеда. Если количество купонов больше или равно 1, то в ячейку с индексами i и j-1 записывается минимальное из текущего и предыдущего. Далее возвращается последовательность выборов обедов.

Результат работы кода на примерах из текста задачи:

5	260
110	0 2
40	3
120	5
110	
60	
3	220
110	0 1
110	2
110	

Результат работы кода на максимальных и минимальных значениях:



	Время выполнения, с	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи(1 элемент)	0.000384893747398473	22.2
Пример из задачи	0.0003264634786187324	22.3
Пример из задачи	0.000327467634784435	22.4
Верхняя граница диапазона значений входных данных из текста задачи(100 элементов)	0.000324423737467434	22.4

Вывод по задаче: произвольный вывод количества оставшихся.

Дополнительные задачи

Задание №17. Ход конем[2,5 баллов]

Условие 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.

Листинг кода:

```
f = open('input.txt')
n = int(f.readline())
a = [[1, 2, 3], [4, 5, 6], [7, 8, 9], ['.', 0, '.']]
x = [[]*i for i in range(10)]
b = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
for i in range(len(a)):
    for j in range(len(a[i])):
        if a[i][j] != '.':
            if i >= 2:
                if j > 0:
                    if a[i-2][j-1] != '.':
                        x[a[i][j]].append(a[i-2][j-1])
                if j != 2:
                    if a[i-2][j+1] != '.':
                        x[a[i][j]].append(a[i-2][j+1])
            if i > 0:
                if j == 2:
                    if a[i-1][j-2] != '.':
                        x[a[i][j]].append(a[i-1][j-2])
                if j == 0:
                    if a[i-1][j+2] != '.':
                        x[a[i][j]].append(a[i-1][j+2])
            if i != 3:
                if j == 2:
                    if a[i+1][j-2] != '.':
                        x[a[i][j]].append(a[i+1][j-2])
                if j == 0:
                    if a[i+1][j+2] != '.':
                        x[a[i][j]].append(a[i+1][j+2])
            if i <= 1:
                if j > 0:
                    if a[i+2][j-1] != '.':
                        x[a[i][j]].append(a[i+2][j-1])
                if j != 2:
```

```

        if a[i + 2][j + 1] != '.':
            x[a[i][j]].append(a[i + 2][j + 1])

def hod_konem(s, l, o):
    if s == 1:
        return 1
    else:
        summ = 0
        for i in range(len(l[o])):
            summ += hod_konem(s-1, l, i)
        return summ

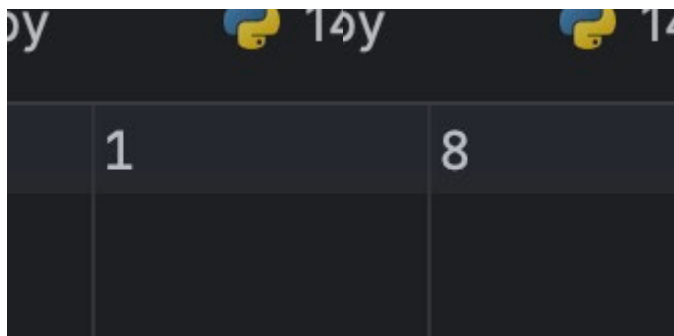
print(x)
summ = 0
for i in b:
    if i != 0 and i != 8:
        summ += hod_konem(n, x, i)
print(summ)

```

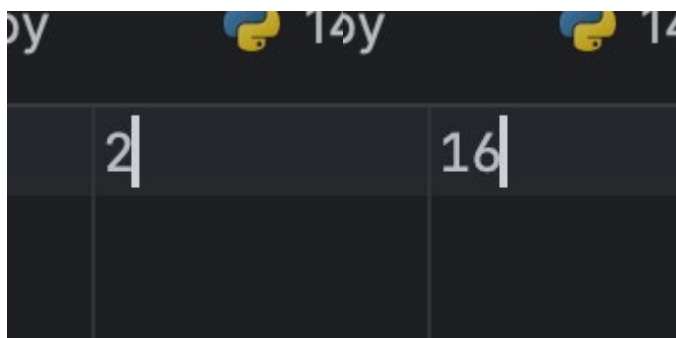
Текстовое объяснение решения:

С помощью цикла вычисляются доступные ходом коня для каждого числа цифры. Далее запускается цикл. В нем, если цифра не равна 0 или 8, то к количеству номеров определенной длины прибавляется количество номеров этой длины, начинающихся с этой цифры, вычисляемая в функции `hod_konem`. В этой функции если требуемая длина равна 1, возвращается 1. Иначе к сумме добавляется длина на 1 меньше.

Результат работы кода на примерах из задачи:



1	8



	Время выполнения, с	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи(1 элемент)	0.00655467668754467	22.2
Пример из задачи	0.00655456487658764	22.3
Пример из задачи	0.00789688544875769	22.4
Верхняя граница диапазона значений входных данных из текста задачи(100 элементов)	-	-

Вывод по задаче: Задача не прошла тест на больших числах, так как была задействована рекурсия.

Задание №2. Заправки[0,5 баллов]

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 $stop_1, stop_2, \dots, stop_n$ из вашего родного города. Какое минимальное количество заправок необходимо?

Код:

```
f = open("input.txt")
rast_B = int(f.readline())
max_km = int(f.readline())
c_stop = int(f.readline())
a = f.readline().split()
stop = [0] + [int(x) for x in a] + [rast_B]
```

```

def min_stops(m, A, r):
    num, cr = 0, 0
    while cr <= r:
        lr = cr
        while cr <= r and A[cr+1] - A[lr] < m:
            cr += 1
        if lr == cr:
            return -1
        if cr <= r:
            num += 1
    return num

p = min_stops(max_km, stop, c_stop)
s = open("output.txt", "w")
s.write(str(p))
s.close()

```

Текстовое объяснение решения:

Функция `min_stops` работает, пока текущая заправка меньше или равна последней остановке. Последняя становится текущей. Пока текущая меньше последней и расстояние между следующей после текущей и последней меньше расстояния без бензина. Текущий становится на 1 больше. Если последний равен текущему, то возвращается -1. Если текущий меньше последнего, то к количеству остановок прибавляется 1.

Результат работы кода на примерах из задачи:

	950
	400
	4
	200 375 550 750
	2
	10
	3
	4
	1 2 5 9
	-1

```
200
250
2
100 150|
0
```

Результат работы кода на минимальных значениях:

```
1
1
1
1|
-1
```

Результат работы кода на максимальных значениях:

100000
400
300
324
463
-1

	Время выполнения, с	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи(1 элемент)	0.004727838723982173	22.1
Пример из задачи	0.00467367362738237	22.2
Пример из задачи	0.00789688544875769	22.2
Верхняя граница диапазона значений входных данных из текста задачи(100 элементов)	0.00767362873682732	22.4

Вывод по задаче: Задача интересная

Задание №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 накопили.

Код:

```
f = open("input.txt")
n = int(f.readline())
s = f.readline().split()
u = [int(i) for i in s]

def check_split_array(x):
    if sum(x) % 3 == 0:
        fl = 1
        tmp = sum(x) / 3
        x.sort()
        x.reverse()
        for _ in range(3):
            y = []
            j = 0
            while sum(y) != tmp and j < len(x):
                if (sum(y) + x[j]) <= tmp:
                    y.append(x[j])
                    del (x[j])
                else:
                    j += 1
            if sum(y) != tmp:
                fl = 0
                return fl
        return fl
    else:
        fl = 0
        return fl

z = open("output.txt", "w")
z.write(str(check_split_array(u)))
```

Текстовое объяснение решения:

Функция проверяет, делится ли сумма всех чисел на три. Если да, то в tmp записывается сумма деленная на три. Запускается цикл на три итерации. Пока сумма не равна tmp и индекс меньше длины списка. Если сумма

текущей рассматриваемой суммы и элемента списка меньше чем tmp, то в у записывается элемент списка и удаляется из самого списка. Иначе происходит переход к следующему элементу в списке.

Результат работы кода на примерах из задачи:

13
1 2 3 4 5 5 7 7 8 10 12 19 25

1

4	0
3 3 3 3	

1	0
40	

11
17 59 34 57 17 23 67 1 18 2 59

1

Результат работы кода на минимальных значениях:

```

1
1
0

```

Результат работы кода на максимальных значениях:

```

20
1 2 3 4 5 6 7 8 9 13 15 17 18 19
0

```

	Время выполнения, с	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи(1 элемент)	0.005467368762387537	22.1
Пример из задачи	0.005632536725636723	22.3
Пример из задачи	0.005736273627372832	22.3
Верхняя граница диапазона значений входных данных из текста задачи(100 элеметов)	0.008273673628738328	22.4

Вывод по задаче: Задача позволяет узнать можно ли разделить на три подмассива с одинаковыми суммами.

Задание №4. Сбор подписей [0,5 баллов]

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

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

Код:

```
f = open('input.txt')
c_otrezki = int(f.readline())
otrezki = []
for i in range(c_otrezki):
    k = f.readline()
    s = [int(x) for x in k.split()]
    otrezki.append(s)

def min_tochki(A, l):
    A.sort()
    c = 0
    for j in range(l-1):
        if A[j][1] >= A[j+1][0]:
            A[j+1] = [max(A[j][0], A[j+1][0]), min(A[j][1], A[j+1][1])]
            A[j] = '*'
    return A

rez = min_tochki(otrezki, c_otrezki)
g = open('output.txt', 'w')
g.write(str(len(rez) - rez.count('*')) + '\n')
for i in rez:
    if i != '*':
        g.write(str(i[0]) + ' ')
```

Текстовое объяснение решения:

Список сортируется по возрастанию. Запускается цикл, работающий столько, сколько отрезков. Если конец отрезка больше или равен началу

следующего, то в следующий отрезок записывается пересечение этих отрезков.

Результат работы кода на примерах из задачи:

3		1	
1	3	3	
2	5		
3	6		

4		2	
4	7	2	5
1	3		
2	5		
5	6		

Результат работы кода на минимальных значениях:

1		1	
0	5	0	

Результат работы кода на максимальных значениях:

100	1
195 38317	487
317 55441	
368 98264	
228 34963	

	Время выполнения, с	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи(1 элемент)	0.004763478362786338	22.1
Пример из задачи	0.004762357263872632	22.3
Пример из задачи	0.004632567356127327	22.3
Верхняя граница диапазона значений входных данных из текста задачи(100 элементов)	0.007728368723872832	22.4

Вывод по задаче: Задача позволяет найти как можно меньше точек, чтобы были задействованы все отрезки.

Вывод по лабораторной работе

Лабораторная работа посвящена динамическому программированию и жадным алгоритмам. В ее ходе я научилась применению различных алгоритмов.