Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет **Инфокоммуникационных технологий**

Образовательная программа **Мобильные и сетевые технологии**

Направление подготовки (специальность) **09.03.03 Прикладная информатика**

О Т Ч Е Т П О Л А Б О Р А Т О Р Н О Й Р А Б О Т Е

по дисциплине «Алгоритмы и структуры данных»

на тему: Жадные алгоритмы. Динамическое программирование

Выполнил:

Самаров Илья Игоревич,

Студент 1 курса гр. K3140

Преподаватель:

Харьковская Татьяна Александровна

Дата сдачи:

31.05.2022

Санкт-Петербург 2022

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

• **Ограничения на входные данные**. $1 \le n \le 103$, $0 \le W \le 2 \cdot 106$, $0 \le pi \le 2 \cdot 106$, $0 \le wi \le 2 \cdot 106$ 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.

```
import random
import time


def Knapsack(W, w):
    n = len(w)
    A = [0] * n
    V = 0
    for i in range(n):
        if W == 0:
            return V, A
        a = min(w[i][1], W)
        V = V + a * (w[i][0] / w[i][1])
        w[i][1] = w[i][1] - a
        A[i] = A[i] + a
        W = W - a
    return V, A


n, m = map(int, input().split())
arr = []
for i in range(n):
    arr.append(list(map(int, input().split())))
    if arr[i][1] == 0:
        arr[i].append(0)
    else:
        arr[i].append(arr[i][0] / arr[i][1])
time_start = time.perf_counter()
arr = sorted(arr, key=lambda a: a[2], reverse=True)
res = Knapsack(m, arr)[0]
print('{:.4f}'.format(res))
print(time.perf_counter() - time_start)
```

«Жадный» алгоритм:

• Пока рюкзак не полон

• Выберите элемент i с максимальным p/w

• Если элемент помещается в рюкзак, возьмите его целиком

• В противном случае возьмите столько, чтобы наполнить рюкзак

• Повторяем этот процесс, пока рюкзак не заполнится.

• В конце возвращаем общую ценность уложенных предметов и их количество.

2 задача. Заправки (0.5 балла)

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 stop1, stop2, ..., stopn 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 stop1, stop2, ..., stopn.

• **Ограничения на входные данные**. $1 \le d \le 10^5$, $1 \le m \le 400$, $1 \le n \le 300$, $1 < stop1 < stop2 < ... < stopn < d$

• **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 stop1, stop2, ..., stopn 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.

```
import time


def MinRefills(x, n, L):
    numRefills = 0
    currentRefill = 0
    while currentRefill <= n:
        lastRefill = currentRefill
        while currentRefill <= n and x[currentRefill + 1] - x[lastRefill] <=
L:
            currentRefill = currentRefill + 1
        if currentRefill == lastRefill:
            return -1
        if currentRefill <= n:
            numRefills = numRefills + 1
    return numRefills


d = int(input())
m = int(input())
n = int(input())
stops = [0] + list(map(int, input().split())) + [d]
time_start = time.perf_counter()
print(MinRefills(stops, n, m))
print(time.perf_counter() - time_start)
```

Пояснение.

Жадный алгоритм:

• Начинаем в A

• Заправиться на самой дальней из доступных заправок - G

• Делаем G - новым пунктом A

• Добраться из нового пункта A в B с минимальным количество дозаправок (решить подзадачу)

• Подзадача – это задача, аналогичная первоначальной, но меньшего размера.

3 задача. Максимальный доход от рекламы (0.5 балла)

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

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

• **Формат ввода / входного файла (input.txt).** В первой строке содержится целое число n, во второй - последовательность целых чисел a1, a2, ..., an, в третьей - последовательность целых чисел b1, b2, ..., bn.

• **Ограничения на входные данные.** $1 \le n \le 103$, $-105 \le ai, bi \le 105$, для всех $1 \le i \le n$.

• **Формат вывода / выходного файла (output.txt).** Выведите максимальное значение Pn i=1 aici, где c1, c2, ..., cn является перестановкой b1, b2, ..., bn.

• Ограничение по времени. 2 сек.

```
import random
import time

n = int(input())
a = list(map(int, input().split()))
b = list(map(int, input().split()))
a.sort()
b.sort()
res = 0
for i in range(n):
    res += a[i] * b[i]
print(res)

# test
n = 10 ** 3
a = [random.randint(-10 ** 5, 10 ** 5) for i in range(n)]
b = [random.randint(-10 ** 5, 10 ** 5) for i in range(n)]
time_start = time.perf_counter()
a.sort()
b.sort()
res = 0
for i in range(n):
    res += a[i] * b[i]
print(res)
print(time.perf_counter() - time_start)
```

Пояснение. Отсортируем первый и второй массивы и будем перемножать элементы с одинаковыми индексами, чтобы максимизировать значение.

4 задача. Сбор подписей (0.5 балла)

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.

• **Постановка задачи.** Дан набор из n отрезков [a0, b0], [a1, b1], ..., [an−1, bn−1] с координатами на прямой, найдите минимальное количество m точек такое, чтобы каждый отрезок содержал хотя бы одну точку. То есть найдите набор целых чисел X минимального размера такой, чтобы для любого отрезка [ai , bi] существовала точка x ∈ X такая, что ai ≤ x ≤ bi .

• **Формат ввода / входного файла (input.txt).** Первая строка входных данных содержит количество отрезков n. Каждая из следующих n строк содержит два целых числа ai и bi (через пробел), определяющие координаты концов i-го отрезка.

• **Ограничения на входные данные**. 1 ≤ n ≤ 102, 0 ≤ ai , bi ≤ 109 - целые для всех 1 ≤ i ≤ n.

• **Формат вывода / выходного файла (output.txt).** Выведите минимальное количество m точек в первой строке и целочисленные координаты этих m точек (через пробел) во второй строке. Вывести точки можно в любом порядке. Если таких наборов точек несколько, можно вывести любой набор. (Нетрудно видеть, что всегда существует множество точек минимального размера, для которых все координаты точек - целые числа.)

• Ограничение по времени. 2 сек.

```
import time


def dots(segments):
    resulting_segments = []
    while len(segments) > 0:
        if len(segments) < 2:
            seg = segments.pop()
            resulting_segments.append(seg)
        else:
            a, b = segments[0], segments[1]
            if b[0] <= a[1]:
                left, right = b[0], b[1] if b[1] <= a[1] else a[1]
                overlapping = (left, right)
                segments = segments[2:]
                segments = [overlapping] + segments
            else:
                resulting_segments.append(segments[0])
                segments = segments[1:]
    return [x[1] for x in resulting_segments]


n = int(input())
segments_input = []
for i in range(n):
    segments_input.append(tuple(map(int, input().split())))
print(segments_input)
time_start = time.perf_counter()
```

```
segments_input.sort(key=lambda x: (x[0], x[1]))
result = dots(segments_input)
print(len(result))
print(' '.join(map(str, result)))
print(time.perf_counter() - time_start)
```

10000Пояснение. Сортируем по левой границе (потом по правой). Далее просматриваем входит ли интервал в последующий, если да то создаем новый, иначе добавляем в отчет точку.

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 a1 + a2 + ... + ak, где a1, ..., ak - натуральные числа и ai ≠ aj для всех $1 \le i < j \le k$.

• **Формат ввода / входного файла (input.txt).** Входные данные состоят из одного целого числа n.

• **Ограничения на входные данные**. $1 \le n \le 10^9$.

• **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

```
import time


def get_sum(num):
    res = 0
    i = 1
    while res + i <= num:
        res += i
        i += 1
    res_array = [j for j in range(1, i - 1)]
    res_array.append(i - 1 + num - res)
    return res_array


n = int(input())
time_start = time.perf_counter()
res_for_n = get_sum(n)
print(len(res_for_n))
print(*res_for_n)
print(time.perf_counter() - time_start)
```

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 a1, a2, ..., an.

• **Ограничения на входные данные.** $1 \le n \le 102$, $1 \le a_i \le 103$ для всех $1 \le i \le n$.

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

• Ограничение по времени. 2 сек.

```
import random
import time


def LargestNumber(Digits):
    answer = ''
    while Digits:
        max_digit = '.'
        for digit in Digits:
            i = 0
            min_len = min(len(digit), len(max_digit))
            while i < min_len-1 and digit[i] == max_digit[i]:
                i += 1
            if i == min_len-1:
                if len(digit) > len(max_digit):
```

```
                    max_digit = max_digit if digit[i] < max_digit[0] else
digit
                else:
                    max_digit = digit if max_digit[i] < digit[0] else
max_digit
            elif digit[i] > max_digit[i]:
                max_digit = digit
        answer += max_digit
        Digits.remove(max_digit)
    return answer


n = int(input())
a = list(map(str, input().split()))
time_start = time.perf_counter()
print(LargestNumber(a))
print(time.perf_counter() - time_start)

# test
n = 10 ** 2
a = list(map(str, [random.randint(1, 10 ** 3) for i in range(n)]))
print(n, a)
time_start = time.perf_counter()
print(LargestNumber(a))
print(time.perf_counter() - time_start)
```

Пояснение. Во-первых, числа теперь в строковом формате. Ищем минимальную длину между текущим числом и максимальным. Находим индекс, когда цифры в числах различаются. В зависимости кончается ли какое-то из чисел или нет выбираем новое максимальное.

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 t1, ...tn - количество минут, которые требуются, чтобы починить i-й сапог.

• **Ограничения на входные данные.** $1 \le K \le 1000$, $1 \le n \le 500$, $1 \le ti \le 100$ для всех $1 \le i \le n$

• **Формат вывода / выходного файла (output.txt).** Выведите одно число – максимальное количество сапог, которые можно починить за один рабочий день.

• Ограничение по времени. 2 сек.

```
import time


def get_res(m, n, a):
    a.sort()
    sum = 0
    i = 0
    while i < n and a[i] + sum <= m:
        sum += a[i]
        i += 1
    print(i)


m_in, n_in = map(int, input().split())
a_in = list(map(int, input().split()))

time_start = time.perf_counter()
get_res(m_in, n_in, a_in)
print(time.perf_counter() - time_start)
```

Пояснение. Сортируем числа. Суммируем пока сумма не превышает максимум. Считаем при этом количество и его выводим.

8 задача. Расписание лекций (1 балла)

• **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 [si , fi) - 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 si и fi для каждой лекции i. Гарантируется, что si < fi . Время начала и окончания лекции - натуральные числа, не превышают 1440 (в минутах с начала суток).

• **Ограничения на входные данные**. $1 \le N \le 1000$, $1 \le si , fi \le 1440$

• **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.

```
n = int(input())
arr = []
for i in range(n):
    arr.append(list(map(int, input().split())))
arr.sort(key=lambda x: x[1])
B = arr[0][1]
res = [arr[0]]
for i in range(1, n):
    if B <= arr[i][0]:
        res.append(arr[i])
        B = arr[i][1]
print(len(res))

arr.sort(key=lambda x: x[1])
B = arr[0][1]
res = [arr[0]]
for i in range(1, n):
    if B <= arr[i][0]:
        res.append(arr[i])
        B = arr[i][1]
print(len(res))
print(time.perf_counter() - time_start)
```

Пояснение. Сортируем по правой границе. Добавляем интервал если левая граница новой пары правее правой границы последней выбранной пары.

10 задача. Яблоки (1 балл)

• **Постановка задачи**. Алисе в стране чудес попались n волшебных яблок. Про каждое яблоко известно, что после того, как его съешь, твой рост сначала уменьшится на ai сантиметров, а потом увеличится на bi 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.

• **Формат ввода / входного файла (input.txt).** В первой строке вводятся натуральные числа n и s – число яблок и начальный рост Алисы. В следующих n строках вводятся пары натуральных чисел ai , bi через пробел.

• **Ограничения на входные данные**. $1 \le n \le 1000$, $1 \le s \le 1000$, $1 \le ai , bi \le 1000$.

• **Формат вывода / выходного файла (output.txt).** Если яблоки съесть нельзя, выведите число -1. Иначе выведите n чисел – номера яблок, в том порядке, в котором их нужно есть.

• Ограничение по времени. 2 сек.

• Ограничение по памяти. 256 мб.

```
import random
import time


def apples(h, x):
    res = []
    while len(x) > 0:
        for i in range(len(x)):
            if h - x[i][0] > 0:
                h += x[i][1]
                res.append(x[i][2])
                del x[i]
                break
            elif i == len(x) - 1:
                return [-1]
    return res


n, s = map(int, input().split())
apples_arr = []
for i in range(n):
    apples_arr.append(list(map(int, input().split())))
    apples_arr[i][1] = apples_arr[i][1] - apples_arr[i][0]
    apples_arr[i].append(i + 1)
apples_arr = sorted(apples_arr, key=lambda apple: apple[1])
apples_arr.reverse()

time_start = time.perf_counter()
print(*apples(s, apples_arr))
print(time.perf_counter() - time_start)
```

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

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, афторефераты для защиты и т.п.). Вы оценили объём печати в N листов. Фирма, готовая размножить печатные материалы, предлагает следующие финансовые условия.

```
with open("input.txt") as file: #
    a = []
    ceni = []
    zakaz = int(file.readline())
    for i in range(7):
        a.append(file.readline().split("\n"))
        ceni = []
        for i in range(len(a)):
            ceni.append(int(a[i][0]))
b = {}
def slovar(a):
    c = 1
    for i in range(7):
        b[c] = a[i]
        c *= 10

slovar(ceni)
def schet(b,zakaz):
    vigodnoe = 1000000000000000000000
    itogo = 0
    for i in b.keys():
        if zakaz // i == 0:
            if vigodnoe > b[i]:
                vigodnoe = b[i]
        else:
            itogo = (zakaz // i)*b[i]
            mod = zakaz % i
            itogo += schet(b,mod)
    if vigodnoe < itogo:
        return vigodnoe
    else:
        return itogo
print(ceni)
print(b)
print(schet(b,zakaz))
```

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 слеующей строке записано n целых чисел w0, w1, ..., wn−1, определяющие вес золотых слитков.

• Ограничения на входные данные. $1 \le W \le 104$, $1 \le n \le 300$, $0 \le w0, ..., wn-1 \le 105$

w0, ..., wn−1 ≤ 105

• Формат вывода / выходного файла (output.txt). Выведите максимальный вес золота, поместится в сумку вместимости W.

• Ограничение по времени. 5 сек.

```
import random
import time

K, n = map(int, input().split())
A = list(map(int, input().split()))

F = [0] * (K + 1)
F[0] = 1
Prev = [0] * (K + 1)
for i in range(len(A)):
    for k in range(K, A[i] - 1, -1):
        if F[k - A[i]] == 1:
            F[k] = 1
            Prev[k] = A[i]

i = K
while F[i] == 0:
    i -= 1
Ans = 0
while i > 0:
    Ans += Prev[i]
    i -= Prev[i]
print(Ans)

# test
K, n = 10 ** 4, 300
A = [random.randint(0, 10 ** 5) for i in range(n)]
time_start = time.perf_counter()
```

```
A = [random.randint(0, 10 ** 5) for i in range(n)]
time_start = time.perf_counter()

F = [0] * (K + 1)
F[0] = 1
Prev = [0] * (K + 1)
for i in range(len(A)):
    for k in range(K, A[i] - 1, -1):
        if F[k - A[i]] == 1:
            F[k] = 1
            Prev[k] = A[i]
i = K
while F[i] == 0:
    i -= 1
Ans = []
k = i
while k > 0:
    Ans.append(Prev[k])
    k -= Prev[k]

print(sum(Ans))
print(time.perf_counter() - time_start)

```

13 задача. Сувениры (1.5 балла)

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. $1 \le n \le 20$, $1 \le vi \le 30$ 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.

```
import random
import time


def three_partition_bu(arr):
    n = len(arr)
    s = sum(arr)

    if s % 3 != 0:
        return False

    D = [[[None for l in range(s + 1)] for j in range(s + 1)] for i in
range(n + 1)]
    for i in range(n + 1):
        D[i][0][0] = True

    for s1 in range(s + 1):
        for s2 in range(s + 1):
            D[0][s1][s2] = (s1 == 0) and (s2 == 0)

    for i in range(1, n + 1):
        for s1 in range(0, s + 1):
            for s2 in range(0, s + 1):
                D[i][s1][s2] = D[i - 1][s1][s2] or D[i - 1][s1 - arr[i -
1]][s2] or D[i - 1][s1][s2 - arr[i - 1]]

    return D[n][s // 3][s // 3]


n = int(input())
arr = [int(i) for i in input().split()]
result = three_partition_bu(arr)
print(1 if result else 0)
```

14 задача. Максимальное значение арифметического выражения.

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 секунд.

```
import random


def getVal(v1, v2, op):
    if op == '+':
        return v1 + v2
    if op == '-':
        return v1 - v2
    if op == '*':
        return v1 * v2


def MinAndMax(i, j, m, M, op):
    minimum = float("+inf")
    maximum = float("-inf")
    for k in range(i, j):
        a = getVal(M[i][k], M[k + 1][j], op[k])
        b = getVal(M[i][k], m[k + 1][j], op[k])
        c = getVal(m[i][k], M[k + 1][j], op[k])
        d = getVal(m[i][k], m[k + 1][j], op[k])
        minimum = min(minimum, a, b, c, d)
        maximum = max(maximum, a, b, c, d)
    return minimum, maximum


def maxValue(d, op):
    n = len(d)
    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]


dd = input()
n = len(dd) // 2 + 1
m = [[None] * n for j in range(n)]
```

```
M = [[None] * n for o in range(n)]
op = [None] * (n - 1)
z = 0
df = []
opf = []
while z < len(dd):
    if z % 2 == 0:
        df.append(int(dd[z]))
    else:
        opf.append(dd[z])
    z += 1
print(maxValue(df, opf))

# test
strr = '+-*'
dd = [random.randint(0, 9) if i % 2 == 0 else strr[random.randint(0, 2)] for
i in range(29)]
print(*dd)
```

Пояснение: Заполняем матрицы M и m в соответствии материалам лекции получаем соответствующий результат.

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 109.

• 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.

• Ограничение по времени. 1 сек.

• Ограничение по памяти. 256 мб.

```
def hodnanari(x,lvl):
    global n
```

```
    if lvl == n-1:
        if (x == 4 or x == 6):
            return 3
        else:
            return 2
    if x == 1:
        return hodnanari(8,lvl+1)+hodnanari(6,lvl+1)
    if x == 2:
        return hodnanari(7,lvl+1)+hodnanari(9,lvl+1)
    if x == 3:
        return hodnanari(4,lvl+1)+hodnanari(8,lvl+1)
    if x == 4:
        return hodnanari(3,lvl+1)+hodnanari(9,lvl+1)+hodnanari(0,lvl+1)
    if x == 6:
        return hodnanari(1,lvl+1)+hodnanari(7,lvl+1)+hodnanari(0,lvl+1)
    if x == 7:
        return hodnanari(2,lvl+1)+hodnanari(6,lvl+1)
    if x == 8:
        return hodnanari(1,lvl+1)+hodnanari(3,lvl+1)
    if x == 9:
        return hodnanari(4,lvl+1)+hodnanari(2,lvl+1)
    if x == 0:
        return hodnanari(4,lvl+1)+hodnanari(6,lvl+1)
a = [1,2,3,4,6,7,9]
k=0
n = int(input(f"Введите n"))
if n == 1:
    print(f"8 номеров можно составить")
else:
    for i in a:
        k += hodnanari(i, 1)
    print(f"{k} номеров можно составить")
```

Пояснение: создана рекурсивная функция, которая отслеживает уровень, на котором находится в номере и текущую цифру таким образом осуществляется подсчет.

Вывод:

В данной работе изучили жадные алгоритмы и продолжили изучать динамическое программирование. Узнали, что такое безопасный ход. Познакомились с задачей о рюкзаке. Заполоняли точки минимальным количеством отрезков и наоборот. Решали задачи разбивая их на подзадачи.