УНИВЕРСИТЕТ ИТМО

Факультет инфокоммуникационных технологий

Мобильные и сетевые технологии

Лабораторная работа №2.1

по дисциплине

«Алгоритмы и структуры данных»

Выполнила:

Юсипов Артем Александрович

К3140

Преподаватель:

Харьковская Т. А.

Санкт-Петербург

2022

Задача №3

Описание задания:

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.

Решение:

Классический жадный алгоритм. Перемножаем максимальные числа с максимальными и наоборот.

```
num = input()
first_arr = list(map(int, input().split()))
second_arr = list(map(int, input().split()))
first_arr.sort()
second_arr.sort()
ans = 0
for i in range(len(first_arr)):
    ans += first_arr[i] * second_arr[i]
print(ans)
```

Вывод:

Задача решается классическим жадным алгоритмом. Также учитываются и отрицательные числа.

Задача №6

Описание задания:

Составить наибольшее число из набора целых чисел.

Решение:

```
def largest_number(numbers):
    answer = ''
    arr_0 = len(str(max(map(int, numbers))))
    while numbers:
        max_numbers = numbers[0]
        for x in numbers:
            if preparation(x, arr_0) > preparation(max_numbers, arr_0):
                max_numbers = x
        answer += max_numbers
        numbers.remove(max_numbers)
    return answer


def preparation(num, arr_0):
    num += '9' * (arr_0 - len(num))
    return int(num)


n = input()
arr = list(input().split())
print(largest_number(arr))
```

Вывод:

Преобразуем числа, добавляя к недостаточно длинным ‘9’

Задача №10

Описание задания:

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.

Решение:

```
n, s = map(int, input().split())
arr = []
for i in range(n):
    arr.append(list(map(int, input().split())))
ans = []
delete_counter = 0
while delete_counter < len(arr):
    cur_apple = 0
    decrease = 10 ** 10
    increase = -10 ** 10
    for i in range(len(arr)):
        if arr[i] is not None:
            if arr[i][0] <= decrease:
                decrease = arr[i][0]
    for i in range(len(arr)):
        if arr[i] is not None:
            if arr[i][1] >= increase and arr[i][0] == decrease:
                increase = arr[i][1]
                cur_apple = i
    s -= arr[cur_apple][0]
    if s <= 0:
        print(-1)
        break
    s += arr[cur_apple][1]
    ans.append(cur_apple + 1)
    arr[cur_apple] = None
    delete_counter += 1
else:
    print(*ans)
```

Вывод:

Среди всех имеющихся яблок мы сначала едим те, которые больше всего уменьшают и увеличивают рост. Если после этого рост останется больше 0, то у нас получится съесть все яблоки.

Задача №15

Описание задания:

Дана строка, составленная из круглых, квадратных и фигурных скобок. Определите, какое наименьшее количество символов необходимо удалить из этой строки, чтобы оставшиеся символы образовывали правильную скобочную последовательность.

Решение:

```
import math

s = input()
n = len(s)
dp = [[0 for i in range(n)] for j in range(n)]
ep = [[0 for h in range(n)] for k in range(n)]
for i in range(n):
    for j in range(n):
        if i == j:
            dp[i][j] = 1
for right in range(n):
    for left in range(right, -1, -1):
        if left == right:
            dp[left][right] = 1
        else:
            minimum = math.inf
            mink = -1
            if s[left] == '(' and s[right] == ')' \
                    or s[left] == '[' and s[right] == ']' \
                    or s[left] == '{' and s[right] == '}':
                minimum = dp[left + 1][right - 1]
            for k in range(left, right):
                if minimum > dp[left][k] + dp[k + 1][right]:
                    minimum = dp[left][k] + dp[k + 1][right]
                    mink = k
            dp[left][right] = minimum
            ep[left][right] = mink

def restoring_response(left, right):
    temp = right - left + 1
    if dp[left][right] == temp:
        return
    if dp[left][right] == 0:
        print(s[left:right + 1], end="")
        return
    if ep[left][right] == -1:
        print(s[left], end="")
        restoring_response(left + 1, right - 1)
        print(s[right], end="")
        return
    restoring_response(left, ep[left][right])
    restoring_response(ep[left][right] + 1, right)


restoring_response(0, n - 1)
```

Вывод:

Задача решается с помощью динамического программирования.

Задача №17

Описание задания:

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.

Решение:

```
def horse(n):
    arr = [[0 for _ in range(n + 1)] for _ in range(10)]
    mod = 10 ** 9
    for i in range(10):
        arr[i][1] = 1
    for num in range(2, n + 1):
        for k in range(10):
            match k:
                case 0:
                    arr[0][num] = (arr[4][num - 1] + arr[6][num - 1]) % mod
                case 1:
                    arr[1][num] = (arr[6][num - 1] + arr[8][num - 1]) % mod
                case 2:
                    arr[2][num] = (arr[9][num - 1] + arr[7][num - 1]) % mod
                case 3:
                    arr[3][num] = (arr[8][num - 1] + arr[4][num - 1]) % mod
                case 4:
                    arr[4][num] = (arr[0][num - 1] + arr[3][num - 1] + arr[9][num - 1]) % mod
                case 6:
                    arr[6][num] = (arr[0][num - 1] + arr[1][num - 1] + arr[7][num - 1]) % mod
                case 7:
                    arr[7][num] = (arr[6][num - 1] + arr[2][num - 1]) % mod
                case 8:
                    arr[8][num] = (arr[1][num - 1] + arr[3][num - 1]) % mod
                case 9:
                    arr[9][num] = (arr[2][num - 1] + arr[4][num - 1]) % mod
    sum = 0
    for j in range(1, 10):
        if j != 8:
            sum = (sum + arr[j][n]) % mod
    return sum
```

Вывод:

Используем рекуррентное соотношение и новый инструмент python 3.10

Задача №11

Описание задания:

Даны n золотых слитков, найдите максимальный вес золота, который поместится в сумку вместимостью W.

Решение:

```
s, n = map(int, input().split())
a = list(map(int, input().split()))
k = [1] + [0] * s
new_k = k[:]
for j in range(len(a)):
    for i in range(a[j], s + 1):
        if k[i - a[j]] == 1:
            new_k[i] = 1
    k = new_k[:]
i = s
while k[i] == 0:
    i -= 1
print(i)
```

Вывод:

Задача решается с помощью динамического программирования. Используем массив весов. Последнее значение – ответ.

Задача № 5

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 j ≤ k.

Решение:

```
n = int(input())
ans = []
current_sum = 0
i = 1
while current_sum + i <= n:
    ans.append(i)
    current_sum += i
    i += 1
if current_sum != n:
    ans[-1] += n - current_sum
print(len(ans))
print(*ans)
```

Вывод:

Задача решается разбиением на слагаемые до преодоления за допустимой сумму. Остаток записываем в последнее число.

Задача №7

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.

Решение:

```
s, n = map(int, input().split())
a = list(map(int, input().split()))
a = sorted(a)
i = 0
current_sum = 0
while i < len(a) and current_sum + a[i] <= s:
    current_sum += a[i]
    i += 1
print(i)
```

Вывод:

Задача решается жадным алгоритмом. Мы ищем самое минимальное время