Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Алгоритмы и структуры данных

Лабораторная работа № 2.1

Выполнил:

Кононов С.В

Группа:

К3140.

Преподаватель:

Харьковская Т.А.

Санкт-Петербург,

14 марта 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 оцени- ли ожидаемое количество кликов в день для каждого слота. Теперь ваша цель - распределить рекламу по слотам, чтобы максимизировать общий доход.

Решение:

Жадный алгоритм. Стараемся перемножить максимальное с максимальным и наоборот.

```
n = input()
a_arr = list(map(int, input()))
b_arr = list(map(int, input()))

a_arr.sort()
b_arr.sort()

ans = 0

for i in range(len(a_arr)):
   ans += a_arr[i] * b_arr[i]

print(ans)
```

Вывод :

Задача решается простым жадным алгоритмом, при этом учитываются и отрицательные числа.

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 = []

cur_sum = 0

i = 1

while cur_sum + i <= n:

   ans.append(i)

   cur_sum += i

   i+=1

if cur_sum != n:

   ans[-1] += n - cur_sum

print(len(ans))

print(*ans)
```

Вывод :

Задача решается разбиением на слагаемые (начиная с 0), до выхода за допустимую сумму. Остаток закидываем в последнее число.

Задача 6:

Составить наибольшее число из набора целых чисел.

Решение:

```
def largest_number(numbers):
    answer = ''
    boster = len(str(max(map(int, numbers))))

    while numbers:
        max numbes = numbers[0]
        for number in numbers:
            if preparete_number(number, boster) >
preparete number(max numbes, boster):
                max numbes = number
        answer += max numbes
        numbers.remove(max_numbes)
    return answer


def preparete_number(num, boster):
    num += '9' * (boster - len(num))
    return int(num)


n = input()

arr = list(input().split())

print(arr)
```

Вывод:

Для решения будем использовать старый алгоритм, но будем преобразовывать числа, для сравнения, добивая более короткое девятками.

Задача 7

В некоей воинской части есть сапожник. Рабочий день сапожника длится K минут. Заведующий складом оценивает работу са- пожника 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
cur_sum = 0

while i < len(A) and cur_sum + A[i] <= S:
    cur_sum += A[i]
    i += 1

print(i)
```

Вывод:

Задача решается жадным алгоритмом. Мы стараемся выбрать самое минимальное время.

Задача 10

Алисе в стране чудес попались n волшебных яблок. Про каждое яблоко известно, что после того, как его съешь, твой рост сна- чала 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 со- всем. 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())

data = []

for i in range(n):
   data.append(list(map(int,
input().split())))
n, s = 3, 5
data = [[2, 3], [10, 5], [5, 10]]

ans = []
delete counter = 0
while delete_counter < len(data):
   cur appel = 0
   decrease min = 10 ** 10
   increase_max = -10 ** 10

   for i in range(len(data)):
       if data[i] is not None:
           if data[i][0] <= decrease_min:
               decrease_min = data[i][0]

   for i in range(len(data)):
       if data[i] is not None:
           if data[i][1] >= increase_max and
data[i][0] == decrease min:
               increase_max = data[i][1]
               cur_appel = i

   s -= data[cur_appel][0]
   if s <= 0:
       print(-1)
       break
   s += data[cur_appel][1]

   ans.append(cur appel + 1)
   data[cur_appel] = None
   delete_counter += 1
else:
   print(*ans)
```

Вывод

Среди всех яблок мы пытаемся первыми съесть те, которые минимальны уменьшают и максимально увеличивают рост. Если во время этого наш рост остается больше 0, то у нас получится съесть все яблоки.

Задача 11

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()))

F = [1] + [0] * S
F_new = F[:]

for j in range(len(A)):
    for i in range(A[j], S + 1):
        if F[i - A[j]] == 1:
            F new[i] = 1
    F = F_new[:]

i = S
while F[i] == 0:
    i -= 1
print(i)
```

Вывод

Задача решается динамически. Мы поддерживаем массив весов которые мы можем взять ровно. Собственно последние значение которое мы можем взять и будет ответом.

Задача 15

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 l in range(n)] for k in range(n)]
ep = [[0 for l 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:
            # База динамики
            dp[left][right] = 1
        else:
            min = math.inf
            mink = -1
            if s[left] == '(' and s[right] ==
')' \
                    or s[left] == '[' and
s[right] == ']' \
                    or s[left] == '{' and
s[right] == '}':
                # Случай соответствующих скобок
                min = dp[left + 1][right - 1]

            # Общий случай правила перехода
динамики
            for k in range(left, right):
                if min > dp[left][k] + dp[k +
1][right]:
                    min = dp[left][k] + dp[k +
1][right]
                    # Поиск оптимального
разбиения строки
                    mink = k
            dp[left][right] = min
            ep[left][right] = mink


# Восстановление ответа
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:
            # Если подстрока имеет в начале и конце
соответствующего типа правильные скобки,
            # то печатаем левую скобку
            print(s[left], end="")
            # Вызов рекурсию вложенной подстроки
            restoring_response(left + 1, right - 1)
            # Печатаем правую скобку
            print(s[right], end="")
            return
        # Вызов рекурсии от левой и правой
подстроки соответсвенно
        restoring_response(left, ep[left][right])
        restoring_response(ep[left][right] + 1,
right)


    restoring_response(0, n - 1)
```

Вывод

Задача решается динамически Чтобы вычислить d[l][r] – наименьшее количество скобок, которое нужно удалить из подстроки s[l..r] разобьем всевозможными способами строку s[l..r] на две подстроки s[l,k] и s[k+1,r], где k=l…r-1. Если подстроки s[l,k] и s[k+1,r] сделать правильными скобочными последовательностями, удалив лишние строки в них, то и строка s[l..r] станет правильной скобочной последовательностью. Поэтому остается найти минимальное значение суммарного количества удаляемых скобок для всевозможных разбиений строки на две подстроки. Это значение и будет давать наименьшее количество удаляемых скобок в строке s[l..r].

Задача 17

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 f(n):
        array = [[0 for j in range(n+1)] for i in
range(10)]
        mod = 10**90000
        for i in range(10):
            array[i][1] = 1

        for num in range(2, n + 1):
            for k in range(10):
                match k:
                    case 0:
                        array[0][num]                =
```

```
(array[4][num - 1] + array[6][num - 1]) % mod
                    case 1:
                        array[1][num]                =
(array[6][num - 1] + array[8][num - 1]) % mod

                    case 2:
                        array[2][num]                =
(array[9][num - 1] + array[7][num - 1]) % mod

                    case 3:
                        array[3][num]                =
(array[8][num - 1] + array[4][num - 1]) % mod

                    case 4:
                        array[4][num]                =
(array[0][num - 1] + array[3][num - 1] +
array[9][num - 1]) % mod

                    case 6:
                        array[6][num]                =
(array[0][num - 1] + array[1][num - 1] +
array[7][num - 1]) % mod

                    case 7:
                        array[7][num]                =
(array[6][num - 1] + array[2][num - 1]) % mod

                    case 8:
                        array[8][num]                =
(array[1][num - 1] + array[3][num - 1]) % mod

                    case 9:
                        array[9][num]                =
(array[2][num - 1] + array[4][num - 1]) % mod
        sum = 0
        for i in range(1, 10):
            if i != 8:
                sum = (sum + array[i][n]) % mod
        return sum
    print(f(100))
```

Вывод

Пусть F(k, d) означает количество набираемых ходом коня последовательностей цифр длины k, первая из которых равна d. Выпишим рекуррентные соотношения для чисел F(k, d). Например, F(k+1, 6) = F(k, 0) + F(k, 1) + F(k, 7). Далее превратим решение из рекуррентного в динамическое .Ответом на задачу будет являться сумма чисел F(N, d), взятая по всем цифрам d, отличным от 0 и 8.

Задача 21

Пете необходимо решить следующую задачу: сможет ли игрок, обладая набором из N карт, отбить M карт, которыми под него сделан ход?

Решение

```
import sys

sys.stdin = open('INPUT.TXT')
sys.stdout = open('OUTPUT1.TXT', 'w')
n, m, trump = input().split()
card_values = {'T': 10, 'J': 11, 'Q': 12, 'K': 13, 'A':
14}
all_close = True

def scan_input():
   cads = input().split()
   temp_dict = dict()
   for card in cads:
       value = int(convert_value(card[0]))
       suit = card[1]
       if suit not in temp_dict:
           temp_dict[suit] = [value]
       else:
           temp_dict[suit].append(value)
   return temp_dict


def convert_value(val):
   if val in card_values:
       return card_values[val]
   return int(val)


def sort_cards(array):
   for elem in array:
       array[elem] = sorted(array[elem])


player_hand = scan_input()
opponent_hand = scan_input()
import sys

sys.stdin = open('INPUT.TXT')
sys.stdout = open('OUTPUT.TXT', 'w')
n, m, trump = input().split()
card_values = {'T': 10, 'J': 11, 'Q': 12, 'K': 13, 'A':
14}
all_close = True

def scan_input():
   cads = input().split()
   temp_dict = dict()
   for card in cads:
       value = int(convert_value(card[0]))
       suit = card[1]
       if suit not in temp_dict:
           temp_dict[suit] = [value]
       else:
           temp_dict[suit].append(value)
```

```
        return temp_dict


    def convert_value(val):
        if val in card_values:
            return card_values[val]
        return int(val)


    def sort_cards(array):
        for elem in array:
            array[elem] = sorted(array[elem])


    player hand = scan input()
    opponent_hand = scan_input()

    sort_cards(player_hand)
    sort_cards(opponent_hand)

    for cur card suit in opponent hand:
        if cur_card_suit not in player_hand:
            if trump in player hand:
                while     opponent_hand[cur_card_suit]     and
player_hand[trump]:
                    opponent hand[cur card suit].pop(0)
                    player_hand[trump].pop(0)
            continue
        op index = 0
        while op_index < len(opponent_hand[cur_card_suit]) and
player hand[cur card suit] and cur card suit != trump:
            op card value = opponent_hand[cur_card_suit][0]
            index = 0
            while index < (len(player_hand[cur_card_suit])-1)
and player_hand[cur_card_suit][index] <= op_card_value:
                index += 1
            if       player_hand[cur_card_suit][index]       >
op_card value:
                player hand[cur card suit].pop(index)
                opponent hand[cur_card_suit].pop(0)
                op index = 0
            elif trump in player_hand and player_hand[trump]:
                player hand[trump].pop(0)
                opponent hand[cur_card_suit].pop(0)
                op index = 0
            op_index += 1

        op index = 0
        while  cur_card_suit  ==  trump  and       op_index  <
len(opponent_hand[trump])  and  trump  in  player_hand  and
len(player hand[trump]) > 0:
            op card value = opponent_hand[trump][0]
            index = 0
            while index < (len(player_hand[trump]) - 1) and
player hand[trump][index] <= op_card_value:
                index += 1
            if player hand[trump][index] > op_card_value:
                player hand[trump].pop(index)
                opponent hand[trump].pop(0)
                op_index = 0
```

```
            op_index += 1

    flag = True
    for cur_card_suit in opponent_hand:
        if len(opponent_hand[cur_card_suit]) > 0:
            flag = False
            break
    if flag:
        print("YES")
    else:
        print("NO")
    sys.stdout.close()
```

Вывод

Для начала будем считать что валет, дама… и тд тоже имеют численное значение 10, 11… соответственно. Далее сортируем карты. Пытаемся покрыть обычную карту, картой той же масти, но большей по номиналу. Если мы не можем это сделать или у нас нет карт нужной масти, то кроем козырем (если он есть). Отдельно крем козырные карты козырным