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 23

Выполнил:
Тарасов А.Н.
К3244

Проверил:
Афанасьев А.В.

Санкт-Петербург
2024 г.

Содержание отчета

Лабораторная работа №1.

Задача №2. Заправки

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?

```
import time
nach = time.time()

def min_ref(x, y, z, n):
    x = [0] + x + [y]

    refuel = 0
    curr = 0
    while curr <= n:
        last_refill = curr

        while curr <= n and x[curr + 1] - x[last_refill] <= z:
            curr += 1

        if last_refill == curr:
            return -1

        if curr <= n:
            refuel += 1

    return refuel


if __name__ == "__main__":
    with open('input.txt', 'r', encoding='utf-8') as input_file,
open('output.txt', 'w',

encoding='utf-8') as output_file:
        y, z, 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_ref(stop, y, z, n)))

kon = time.time()
c = kon - nach
print('Время :', c)

import os, psutil; print(psutil.Process(os.getpid()).memory_info().rss /
1024 ** 2)
```

Изначальное количество заправок равно 0, текущая заправка также равна 0. Мы добавляем в список заправок начальную точку (0) и конечную точку (y), так что x = [0] + x + [y].

Пока не достигнута последняя заправка, проверяем, насколько далеко можно доехать от текущей позиции. Если нет доступных заправок для продолжения пути, возвращаем -1. В противном случае, перемещаемся на самую дальнюю доступную заправку (которая в пределах досягаемости) и увеличиваем счетчик заправок на 1.

Задача №5. Максимальное количество призов

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.

```
import time
nach = time.time()

def main():

    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)}')

kon = time.time()
c = kon - nach
print('Время :', c)
```

```
import os, psutil; print(psutil.Process(os.getpid()).memory_info().rss /
1024 ** 2)
```

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

Задача №6. Максимальная зарплата

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.
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.

```
import time
nach = time.time()

def main():

    def max_num(n, arr):
        for i in range(n - 1):
            for j in range(n - i - 1):
                v1 = arr[j] + arr[j + 1]
                v2 = arr[j + 1] + arr[j]
                if v1 < v2:
                    arr[j], arr[j + 1] = arr[j + 1], arr[j]
        return "".join(arr)
    with open("input.txt", "r") as file:
        n = int(file.readline())
        arr = list(file.readline().split())
    with open("output.txt", "w") as file:
        file.write(str(max_num(n, arr)))

kon = time.time()
```

```
c = kon - nach
print('Время :', c)

import os, psutil; print(psutil.Process(os.getpid()).memory_info().rss /
1024 ** 2)
```

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

Задача №13. Сувениры

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

```
import time
nach = time.time()

def main():

    with open("input.txt", "r") as file:
        n = int(file.readline())
        values = list(map(int, file.readline().split()))

    total = sum(values)

    if total % 3 != 0:
        result = 0
    else:
        target = total // 3
        matr = [[0 for _ in range(target + 1)] for _ in range(n + 1)]

        for i in range(n + 1):
            matr[i][0] = 1
        for i in range(1, n + 1):
            for j in range(1, target + 1):
                if values[i - 1] <= j:
                    matr[i][j] = matr[i - 1][j] or matr[i - 1][j - values[i
-1]]
                else:
                    matr[i][j] = matr[i - 1][j]
        result = 1 if matr[n][target] else 0
    with open("output.txt", "w") as file:
        file.write(str(result))

kon = time.time()
c = kon - nach
print('Время :', c)

import os, psutil; print(psutil.Process(os.getpid()).memory_info().rss /
1024 ** 2)
```

Задача №14. 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) =?

```
import time
nach = time.time()

def main():
    def maxExpressionValue(expression):
        def findMinAndMax(start, end, maxValues, minValues, operators):
            current_min = float("inf")
            current_max = float("-inf")
            for index in range(start, end):
                a = eval(f"{maxValues[start][index]} {operators[index]}
{maxValues[index + 1][end]}")
                b = eval(f"{maxValues[start][index]} {operators[index]}
{minValues[index + 1][end]}")
                c = eval(f"{minValues[start][index]} {operators[index]}
{maxValues[index + 1][end]}")
                d = eval(f"{minValues[start][index]} {operators[index]}
{minValues[index + 1][end]}")
                current_min = min(current_min, a, b, c, d)
                current_max = max(current_max, a, b, c, d)
            return current_min, current_max

        digits = list(map(int, expression[::2]))
        operators = list(expression[1::2])
        length = len(digits)
        minValues = [[0] * length for _ in range(length)]
        maxValues = [[0] * length for _ in range(length)]

        for i in range(length):
            minValues[i][i] = digits[i]
            maxValues[i][i] = digits[i]

        for s in range(1, length):
            for start in range(length - s):
                end = start + s
                minValues[start][end], maxValues[start][end] =
findMinAndMax(start, end, maxValues, minValues, operators)

        return maxValues[0][length - 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()
        res = maxExpressionValue(s)
        output_file.write(str(res))

kon = time.time()
c = kon - nach
```

```
print('Время :', c)

import os, psutil; print(psutil.Process(os.getpid()).memory_info().rss /
1024 ** 2)
```

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 findMinAndMax: 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 maxExpressionValue: 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

Задача № 17. Ход конем

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. Напишите программу, определяющую количество телефонных номеров длины N, набираемых ходом коня. Поскольку таких номеров может быть очень много, выведите ответ по модулю 109

```
import time
nach = time.time()

def main():
```

```
    def dark_knight(n):
        motion = [0, 1, 1, 1, 1, 1, 1, 1, 0, 1]
        mod = 10 ** 9
        for i in range(n - 1):
            motion = [
                (motion[4] + motion[6]) % mod,
                (motion[6] + motion[8]) % mod,
                (motion[7] + motion[9]) % mod,
                (motion[4] + motion[8]) % mod,
                (motion[0] + motion[3] + motion[9]) % mod,
                0,
                (motion[0] + motion[1] + motion[7]) % mod,
                (motion[2] + motion[6]) % mod,
                (motion[1] + motion[3]) % mod,
                (motion[2] + motion[4]) % mod
            ]
        return sum(motion) % 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())
        result = dark_knight(n)
        output_file.write(str(result))

kon = time.time()
c = kon - nach
print('Время :', c)

import os, psutil; print(psutil.Process(os.getpid()).memory_info().rss /
1024 ** 2)
```

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.

Задача №20. Почти палиндром

Слово называется палиндромом, если его первая буква совпадает с последней, вторая – с предпоследней и т.д. Например: «abba», «madam», «x». Для заданного числа K слово называется почти палиндромом, если в нем можно изменить не более K любых букв так, чтобы получился

палиндром. Например, при K = 2 слова «reactor», «kolobok», «madam» являются почти палиндромами (подчеркнуты буквы, заменой которых можно получить палиндром). Подсловом данного слова являются все слова, получающиеся путем вычеркивания из данного нескольких (возможно, одной или нуля) первых букв и нескольких последних. Например, подсловами слова «cat» являются слова «c», «a», «t», «ca», «at» и само слово «cat» (а «ct» подсловом слова «cat» не является). Требуется для данного числа K определить, сколько подслов данного слова S являются почти палиндромами

```
import time
nach = time.time()

def main():

    def count_palindromic_substrings(length, max_changes, string):
        total_palindromes = 0

        def check_palindromes(center, even_length=False):
            nonlocal total_palindromes
            changes_made = 0
            for offset in range(1, length):
                if even_length:
                    left_idx = center - offset + 1
                    right_idx = center + offset
                else:
                    left_idx = center - offset
                    right idx = center + offset

                if left_idx < 0 or right_idx >= length:
                    break
                if string[left_idx] != string[right_idx]:
                    if changes_made != max_changes:
                        total_palindromes += 1
                        changes_made += 1
                    else:
                        break
                else:
                    total_palindromes += 1

        for center in range(length):
            total_palindromes += 1
            check_palindromes(center)
            check_palindromes(center, True)

        return total_palindromes


    with open("input.txt", "r") as file:
        length, max_changes = map(int, file.readline().split(" "))
        string = file.readline().strip()

    res = count_palindromic_substrings(length, max_changes, string)

    with open('output.txt', 'w') as file:
```

```
        file.write(str(res))

kon = time.time()
c = kon - nach
print('Время :', c)

import os, psutil; print(psutil.Process(os.getpid()).memory_info().rss /
1024 ** 2)
```

Принцип простой: находим у слова середину, затем постепенно “наращиваем” его, определяя, является ли полученное палиндромом.

Вывод

В ходе лабораторной работы я научился использовать жадный алгоритм.