Санкт-Петербургский Национальный Исследовательский Университет
Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий и систем связи

Лабораторная работа №4

Вариант №2

Выполнил(и:)

Алексеев Т.

Бабаев Р.

Белисов Г.

Проверил

Мусаев А.А.

Санкт-Петербург,

2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы являлось знакомство с динамическим программирование, его общим принципом и видами.

Для достижения данной цели необходимо было выполнить следующие задания:

1. Задание 1: Вор пробрался в музей и хочет украсть N экспонатов. У каждого экспоната есть свой вес и цена. Вор может сделать M заходов, каждый раз уносят K кг веса. Определить, что должен унести вор, чтобы сумма украденного была максимальной.
2. Задание 2: Дана последовательность матриц A, B, C, …, Z таким образом, что с ними можно выполнить ассоциативные операции. Используя динамическое программирование, минимизируйте количество скалярных операций для нахождения их произведения.
3. Задание 3: Дан массив N, состоящий из n случайных целых чисел, находящихся в диапазоне от -100 до 100. Найти наибольшую непрерывную возрастающую последовательность из чисел внутри массива.

Решения данных задач будут находиться на GitHub по ссылке: https://github.com/NorthPole0499/Algoritms_task_9 .

1. ЗАДАНИЕ 1

Ранее мы решали данную задачу с помощью «жадных» алгоритмов. Теперь решение первой задачи будет выполнено с помощью динамического программирования. В данном случае, восходящего.

Первым делом, после ввода всех данных, сортируется список экспонатов: в начало ставится тот экспонат, у которого соотношение цена/вес наибольшее. Уже потом идёт сортировка по весу в случае совпадения первого параметра. Делается это с помощью многоуровневой сортировки itemgetter из встроенной библиотеки operator.

Далее запускаем цикл for с m итерациями, где m – это количество раз, которые вор может зайти. Внутри него находится сам алгоритм. Мы объявляем переменную current_weight, которая обозначает текущий доступный вес у вора и по умолчанию равна допустимому количество кг, которое может унести вор. И мы проходим по всем элементам нашего списка циклом for. Если экспонат помещается в сумку к вору, то добавляем цену в ответ и вес вычитаем из переменной current_weight.

В конце каждой итерации мы удаляем из основного списка экспонаты, которые вор забрал с собой. Делается это не сразу во избежание появления ошибок и путаницы с индексами в основном списке.

```
Введите количество экспонатов: 4
Введите вес экспоната и его цену: 4 100
Введите вес экспоната и его цену: 3 120
Введите вес экспоната и его цену: 1 10
Введите вес экспоната и его цену: 5 200
Сколько заходов может сделать вор: 1
Сколько кг может унести вор за раз: 7
Максимальная сумма украденного: 220
```

Приложение 1 – Пример вывода программы для Задания 1

2. ЗАДАНИЕ 2

Вторая задача решена также с помощью динамического программирования. Грубо говоря, в этой задаче нам нужно расставить скобки так, чтобы количество скалярных операций было наименьшим.

В ней для начала мы вводим размерность n матриц и составляем матрицу dp размерностью n-1, заполненную нулями. Далее мы заходим в два цикла for, которые работают с определёнными тройками матриц (зависящими от определённой точки k) и заполняют матрицу dp. В каждую ячейку записывается минимум из двух значений: текущего значения ячейки и суммы двух предшествующих подзадач и произведения размерностей трёх матриц. Таким образом, у нас получается верхнетреугольная матрица dp, в которой минимальное количество скалярных операций будет находиться в её верхнем правом углу.

```
Введите размерности матриц через пробел: 10 8 2 4 8
Минимальное число скалрных операций: 384
```

Приложение 2 – Пример вывода программы для Задания 2

3. ЗАДАНИЕ 3

В третьей задаче было применено восходящее динамической программирование.

В самом начале мы заполняем массив data случайными числами с помощью функции randint из модуля random. Далее мы создаём список index_max, который содержит в себе единицу и n – 1 нулей. А в его ячейках будут хранится длины непрерывных восходящих последовательностей.

Таким образом, мы идём циклом for по данному массиву. И каждую итерацию проверяем условие. Если data[i - 1] < data[i], то в i-ый элемент массива index_max записываем index_max[i - 1] + 1. Если условие неверное, то записываем единицу, так как последовательность оборвалась и началась новая.

В итоге максимальный элемент из списка index_max и будет являться длиной самой длинной непрерывной возрастающей последовательности. Найдя это, с помощью срезов и работы с индексами мы без труда восстанавливаем искомую последовательность из списка data.

```
Введите длину массива: 10
Исходный массив: [-45, -29, 92, -3, 1, -57, -36, 99, -63, 42]
Наибольшая непрерывная возрастающая последовательность: [-45, -29, 92]
Длина данной последовательности: 3
```

Приложение 3 – Пример вывода программы для Задания 3

ЗАКЛЮЧЕНИЕ

Таким образом, в ходе выполнения данной лабораторной работы были достигнуты все поставленные цели. Произошло знакомство с динамическим программированием, их принципом и видами, и применение их на практике.

СПИСОК ЛИТЕРАТУРЫ

1) Динамическое программирование – Tproger // URL: https://tproger.ru/experts/what-is-dynamic-programming/ (дата обращения: 30.04.2023)
2) Всё о сортировке в Python: исчерпывающий гайд – Tproger // URL: https://tproger.ru/translations/python-sorting/ (дата обращения: 30.04.2023)