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

Факультет инфокоммуникационных технологий и систем связи

Лабораторная работа №3

Вариант №2

Выполнил(и:)

Алексеев Т.

Бабаев Р.

Белисов Г.

Проверил

Мусаев А.А.

Санкт-Петербург,

2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы являлось знакомство с «жадными» алгоритмами: их общим принципом и алгоритмом Дейкстры.

Для достижения данной цели необходимо было выполнить следующие задания:

1. Задание 1: Пользователю необходимо дать сдачу N рублей. У него имеется M1 монет номиналом S1, M2 монет номиналом S2, M3 монет номиналом S3 и M4 монет номиналом S4. Необходимо найти наименьшую комбинацию из заданных монет, которые позволят получить в сумме N.
2. Задание 2: 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.
3. Задание 3: На основе 1 и 2 задания сделайте выводы о применении «жадных алгоритмов». Всегда ли наилучшие решения на каждом шаге приводят к наилучшему конечному результату?
4. Задание 4: Реализовать алгоритм Дейкстры на основе реальных данных по теме «Улицы Москвы»

Решения данных задач будут находиться на GitHub по ссылке: https://github.com/NorthPole0499/Algoritms_task_8

1. ЗАДАНИЕ 1

В этом задании необходимо было решить классическую задачу про сдачу с определёнными монетами с помощью «жадного алгоритма».

После считывания всех данных от пользователя происходит сортировка списка со входными данными по номиналу монет и по убыванию. После этого запускается цикл for с 4 итерациями, каждая для своего номинала, начиная с большего.

Пока номинал текущей монеты меньше, чем общая сдача, мы отдаём эту монету. Как только это условие не выполняется или монеты данного номинала заканчиваются, мы переходим к следующему максимальному номиналу монет и алгоритм повторяется. Так идёт до тех пор, пока не будет выдана вся сдача.

```
Введите сдачу, которую нужно выдать: 77
Введите номинал первой монеты: 50
Введите количество первых монет: 2
Введите номинал второй монеты: 10
Введите количество вторых монет: 3
Введите номинал третьей монеты: 5
Введите количество третьих монет: 2
Введите номинал четвёртой монеты: 1
Введите количество четвёртых монет: 5

Номинал монеты: 50 Количество: 1
Номинал монеты: 10 Количество: 2
Номинал монеты: 5 Количество: 1
Номинал монеты: 1 Количество: 2
```

Приложение 1 – Пример вывода программы для Задания 1

2. ЗАДАНИЕ 2

Вторая задача также заключается в применении «жадного» алгоритма для решения практической задачи про экспонаты и вора.

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: 3
Введите вес экспоната и его цену: 10 60
Введите вес экспоната и его цену: 20 100
Введите вес экспоната и его цену: 30 120
Сколько заходов может сделать вор: 1
Сколько кг может унести вор за раз: 50
Максимальная сумма украденного: 160
```

Приложение 2 – Пример вывода программы для Задания 2

3. ЗАДАНИЕ 3

Таким образом, можно сделать вывод, что «жадные» алгоритмы очень полезны и удобны. Их очевидным плюсом является удобство использования, ведь их можно применить во множестве сфер. Единственное, что нужно сделать, это правильно структурировать и отсортировать данные. Также, очевидным достоинством является простота и понятность алгоритма, он интуитивно понятен.

Но, в то же время, у «жадных» алгоритмов имеется и большой недостаток. Не во всех ситуациях их применение рационально и приводит к правильному результату. Например, как это было в Задании 2, где на некоторых наборах данных получаются неверные ответы. Да, конечно, с помощью математических средств можно узнать, будет ли корректно работать «жадина» или нет. Но для этого нужно заранее знать область применения алгоритма.

В итоге можно сказать, что «жадные» алгоритмы являются прекрасным средством для решения разнообразных задач, но ситуативным. Нужно думать и понимать, будет ли здесь всегда правильно выполняться алгоритм или нет.

4. ЗАДАНИЕ 4

В четвёртой задаче было необходимо реализовать алгоритм Дейкстры для нахождения кратчайшего пути от заданной улицы Москвы до всех остальных. Данный алгоритм состоит из нескольких шагов:

1) Строим матрицу расстояний (a) на основании данного графа;

2) Создаём одномерный список (d) расстояний от данной улицы до всех остальных, при этом определим, что расстояние данной улицы до самой себя равно нулю;

3) Создаём словарь "дерево", в котором ключи – индексы улиц, а значения – список индексов улиц-потомков;

4) Далее на первом шаге определяем минимальные расстояния для каждой улицы, не включенной в итоговое дерево, по следующей формуле: d[x] = min(d[x], d[y] + a[y][x]), где x – индекс текущей улицы, для которой мы считаем минимальное расстояние, y – индекс последней улицы, включенной в итоговое дерево (для первого шага – y=0);

5) Индекс, соответствующий минимальному расстоянию из всех, что были получены на прошлом шаге, добавляется в итоговое дерево в качестве потомка той улицы, от которой было просчитано расстояние;

6) Повторяем п.4 и п.5 до тех пор, пока не добавим все улицы в итоговое дерево;

7) Выводим пути от данной улицы до всех остальных на основании полученного дерева.

Итак, в нашей программе изначальный граф расстояний задаётся словарём, где ключи – названия улиц, а значения – тоже словари типа "Улица-Расстояние до нее". На первом шаге пользователь вводит название улицы Москвы до тех пор, пока не найдем его в нашем графе.

Далее формируем матрицу расстояний, где на первую строчку поместим улицу, выбранную пользователем, а на остальные – оставшиеся. Последующие шаги соответствуют вышеописанному алгоритму за тем лишь исключением, что мы создадим дополнительный список, в котором будем хранить индексы последней улицы, от которой было посчитано минимальное расстояние до данной.

Вывод осуществляется следующим образом:

Выбранная улица -> ... -> Улица №1

Выбранная улица -> ... -> Улица №2

И т.д.

```
Введите название московской улицы: улица Арбат
улица Арбат -> Чкаловская улица -> Пушкинская площадь -> Тверская улица
улица Арбат -> Чкаловская улица -> Пушкинская площадь -> Страстной бульвар
улица Арбат -> Чкаловская улица -> Пушкинская площадь
улица Арбат -> Чкаловская улица
улица Арбат -> Смоленская площадь
улица Арбат -> Смоленская площадь -> Кутузовский проспект
улица Арбат -> Смоленская площадь -> Кутузовский проспект -> улица Новый Арбат
улица Арбат -> Чкаловская улица -> Пушкинская площадь -> Тверская улица -> Охотный ряд
```

Приложение 3 – Пример вывода программы для Задания 4

ЗАКЛЮЧЕНИЕ

Таким образом, в ходе выполнения данной лабораторной работы были достигнуты все поставленные цели. Произошло знакомство с «жадными» алгоритмами, в частности, с алгоритмом Дейкстры, и применение их на практике.

СПИСОК ЛИТЕРАТУРЫ

1) Жадные алгоритмы – ХАБР // URL: https://habr.com/ru/post/120343/ (дата обращения: 17.03.2023)
2) Всё о сортировке в Python: исчерпывающий гайд – Tproger // URL: https://tproger.ru/translations/python-sorting/ (дата обращения: 16.03.2023)