

Лабораторная работа №4

Оценка сложности эвристических алгоритмов

1 Цель

1.1 Научиться реализовывать, оценивать и применять эвристические алгоритмы.

2 Литература

2.1 Фленов, М.Е. Библия С#. – 3 изд. – Санкт-Петербург: БХВ-Петербург, 2016. – URL: <https://ibooks.ru/bookshelf/353561/reading>. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный. – гл.8.

3 Подготовка к работе

3.1 Повторить теоретический материал (см. п.2).

3.2 Изучить описание лабораторной работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

5.1 Алгоритм сдачи монет

Реализовать и протестировать алгоритм сдачи монет (жадный алгоритм), для того, чтобы найти требуемые монеты для сдачи. Оценить сложность алгоритма.

Эвристика: на каждом шаге требуется выбрать монету с наибольшим номиналом, не превышающим оставшуюся сумму.

Особенность: алгоритм работает оптимально, только если номиналы монет составляют «каноническую систему» (например, {10, 5, 2, 1}). Для других наборов (например, {1, 3, 4}) он может не найти минимальное количество монет.

Пример проблемы: для суммы 6 и монет {1,3,4} алгоритм выдаст {4,1,1}, хотя оптимальное решение {3,3}.

5.2 Алгоритм поиска локального максимума

Реализовать и протестировать алгоритм нахождения локального максимума (элемент, больший своих соседей) среди элементов массива. Оценить сложность алгоритма.

Эвристика: требуется найти первый локальный максимум, сравнивая элемент только с его соседями (для первого и последнего – только с одним рядом стоящим соседом).

Особенность: локальный максимум не обязательно является глобальным максимумом массива. Выбирается первый «локально хороший» результат, даже если существует лучшее решение.

Пример проблемы: в массиве {1, 3, 7, 5, 8} алгоритм остановится на 7, хотя глобальный максимум 8.

5.3 Алгоритм ближайшего соседа

Проанализировать и протестировать имеющийся код, реализующий алгоритм ближайшего соседа. Спроектировать для него схему алгоритма. Оценить сложность алгоритма.

Алгоритм ближайшего соседа решает задачу коммивояжера (обхода всех городов за минимальное расстояние) и позволяет найти приближённый путь. Это классический эвристический подход, так как он работает быстро, но не гарантирует идеального результата (пути минимальной длины).

Эвристика: на каждом шаге требуется выбрать ближайший город, который ещё не посещен.

Особенность: алгоритм не учитывает последствия текущего выбора и не проверяет глобальную оптимальность маршрута. Он строит путь жадно, основываясь только на ближайшей точке.

Пример проблемы: в графах с «ловушками» (например, ближайший город ведет к длинным обходам) маршрут будет неоптимальным.

```
// Двумерный массив отображает в ячейках расстояние между городами.
// По диагонали – расстояние от города до себя, в ячейке на пересечении строк и столбцов –
// расстояние от города с номером строки до города с номером столбца
int[,] distances = {
    { 0, 10, 15, 20 },
    { 10, 0, 35, 25 },
    { 15, 35, 0, 30 },
    { 20, 25, 30, 0 }
};

int n = distances.GetLength(0);
int[] visited = new int[n]; // Помечаем посещённые города
int[] path = new int[n]; // Храним маршрут
int currentCity = 0; // Начинаем с города 0
visited[currentCity] = 1; // Отмечаем как посещённый
path[0] = currentCity;

for (int step = 1; step < n; step++)
{
    int nearestCity = -1;
    int minDistance = int.MaxValue;

    for (int i = 0; i < n; i++)
    {
        if (visited[i] == 0 && distances[currentCity, i] < minDistance)
        {
            minDistance = distances[currentCity, i];
            nearestCity = i;
        }
    }

    visited[nearestCity] = 1; // Помечаем город как посещённый
    path[step] = nearestCity; // Добавляем в маршрут
    currentCity = nearestCity; // Переходим в новый город
}

// Вывод результата
Console.WriteLine("Приближённый маршрут:");
for (int i = 0; i < n; i++)
{
    Console.Write(path[i] + (i < n - 1 ? "->" : ""));
}
```

6 Порядок выполнения работы

6.1 Запустить MS Visual Studio и создать консольное приложение C# с названием LabWork4.

6.2 Выполнить все задания из п.5 в проекте LabWork4. При выполнении заданий использовать минимально возможное количество команд и переменных и выполнять форматирование и рефакторинг кода.

6.3 Ответить на контрольные вопросы.

7 Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Ответы на контрольные вопросы

7.4 Вывод

8 Контрольные вопросы

8.1 Что такое «эвристика»?

8.2 Зачем используются эвристические алгоритмы?

8.3 Какие алгоритмы относятся к эвристическим?