**Практическое занятие 4 «Оценка сложности эвристических алгоритмов»**

Задачи практического занятия:

1. Ознакомление с основными понятиями и характеристиками эвристических алгоритмов.
2. Оценка временной и пространственной сложности эвристических алгоритмов.

Вариант 15:

Ход работы:

using System;

class Program

{

// Матрица расстояний между городами

static int[,] distances = { { 0, 10, 15, 20 }, // Расстояния от города 0 до остальных

{ 10, 0, 35, 25 }, // Расстояния от города 1 до остальных

{ 15, 35, 0, 30 }, // Расстояния от города 2 до остальных

{ 20, 25, 30, 0 } }; // Расстояния от города 3 до остальных

static int numCities = 4; // Количество городов

static int[] path = new int[numCities]; // Массив для хранения текущего пути

static bool[] visited = new bool[numCities]; // Массив для отслеживания посещенных городов

static int minLength = int.MaxValue; // Переменная для хранения длины минимального пути

static void Main(string[] args)

{

TSP(0, 1, 0); // Начинаем с города 0, глубина пути 1, общая длина пути 0

Console.WriteLine("Min Length: " + minLength); // Выводим минимальную длину пути

Console.WriteLine("Path: ");

foreach (var city in path)

{

Console.Write(city + " "); // Выводим города в найденном пути

}

}

// Рекурсивная функция для поиска кратчайшего пути

static void TSP(int currentCity, int depth, int totalLength)

{

if (depth == numCities) // Если достигнута глубина равная количеству городов

{

totalLength += distances[currentCity, 0]; // Добавляем расстояние до стартового города

if (totalLength < minLength) // Если длина текущего пути меньше минимальной

{

minLength = totalLength; // Обновляем минимальную длину

Array.Copy(path, 0, path, 0, numCities); // Копируем найденный путь

}

return;

}

// Перебираем все возможные следующие города

for (int nextCity = 0; nextCity < numCities; nextCity++)

{

if (!visited[nextCity]) // Если город еще не посещен

{

visited[nextCity] = true; // Отмечаем город как посещенный

path[depth] = nextCity; // Добавляем город в текущий путь

TSP(nextCity, depth + 1, totalLength + distances[currentCity, nextCity]); // Рекурсивно ищем путь дальше

visited[nextCity] = false; // Снимаем отметку о посещении для следующих итераций

}

}

}

}

Вывод: Я ознакомился с основными понятиями и характеристиками эвристических алгоритмов.

Оценил временную и пространственную сложности эвристических алгоритмов.