|  |  |  |  |
| --- | --- | --- | --- |
| **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  **КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка**  **ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**  **Кафедра програмних систем і технологій**  **Дисципліна**  **«Кросплатформне програмування»**  **Лабораторна робота 3** | | | |
| **Виконав:** | Чушенко Ярослав Володимирович | **Перевірив**: | Петрівський В.Я |
| Група | ІПЗ-41 | Дата перевірки |  |
| Форма навчання | денна | Оцінка |  |
| Спеціальність | 121 |
| 2023 | | | |

**Варіант – 54**Репозиторій на Github: [Посилання](https://github.com/PingvinAustr/CrossplatformKNU)  
 **Умова завдання:**Для дослідження поверхні Марса вченим необхідно розробити систему оповіщення, яка б змогла передавати інформацію за ланцюговим принципом між містами, які планується там побудувати.

При цьому в кожному місті необхідно побудувати радіостанцію таким чином, щоб був зв'язок між усіма містами. При цьому всі такі станції повинні передавати сигнал один одному на рівній відстані R. Таким чином, буде можлива передача інформації з одного міста в інше лише тоді, коли відстань між ними не більше R.

За заданими координатами міст, з метою економії енергії радіостанцій, Вам слід визначити мінімальне значення R, за якого інформація зможе бути доставлена з будь-якого міста до всіх інших.

*Вхідні дані:*

У першому рядку вхідного файлу INPUT.TXT спочатку записано натуральне число N – кількість міст (N ≤ 1000). Далі йде N рядків, які містять речові координати (Xi, Yi) відповідного міста. (-10000 ≤ Xi, Yi ≤ 10000). Передбачається, що всі міста знаходяться на площині.

*Вихідні дані:*

У вихідний файл OUTPUT.TXT виведіть одне речове число – найменший радіус радіостанцій. Число слід вивести з двома знаками після коми, без лідируючих нулів, у форматі, наведеному в прикладах

*Приклади:*

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 4  0 0  2 0  0 2  2 3 | 2.24 |
| 2 | 3  2 0  0 2  4 2 | 2.83 |

**Виконання роботи:**

1. **Код:**
2. using System.Diagnostics;
3. class Program
4. {
5. static double CalculateDistance((double x, double y) point1, (double x, double y) point2)
6. {
7. double dx = point1.x - point2.x;
8. double dy = point1.y - point2.y;
9. return Math.Sqrt(dx \* dx + dy \* dy);
10. }
11. static double FindSmallestRadius(List<(double x, double y)> cities)
12. {
13. int n = cities.Count;
14. double[] minDistances = Enumerable.Repeat(double.MaxValue, n).ToArray();
15. bool[] connected = new bool[n];
16. double maxEdge = 0;
17. minDistances[0] = 0;
18. for (int count = 0; count < n; count++)
19. {
20. double minDistance = double.MaxValue;
21. int currentCity = -1;
22. for (int i = 0; i < n; i++)
23. {
24. if (!connected[i] && minDistances[i] < minDistance)
25. {
26. minDistance = minDistances[i];
27. currentCity = i;
28. }
29. }
30. if (currentCity == -1) break;
31. maxEdge = Math.Max(maxEdge, minDistance);
32. connected[currentCity] = true;
33. for (int i = 0; i < n; i++)
34. {
35. if (!connected[i])
36. {
37. double distance = CalculateDistance(cities[currentCity], cities[i]);
38. minDistances[i] = Math.Min(minDistances[i], distance);
39. }
40. }
41. }
42. return maxEdge;
43. }
44. static void Main()
45. {
46. var lines = File.ReadAllLines("..//..//..//INPUT.TXT");
47. int n = int.Parse(lines[0]);
48. var cities = new List<(double x, double y)>(n);
49. Console.WriteLine($"Received data from input.txt. Cities:{string.Join(",", cities)}\n\nCalculating...");
50. for (int i = 1; i <= n; i++)
51. {
52. var coordinates = lines[i].Split();
53. cities.Add((double.Parse(coordinates[0]), double.Parse(coordinates[1])));
54. }
55. Stopwatch stopwatch = new Stopwatch();
56. stopwatch.Start();
57. double smallestRadius = FindSmallestRadius(cities);
58. stopwatch.Stop();
59. File.WriteAllText("..//..//..//OUTPUT.TXT", smallestRadius.ToString("F2"));
60. Console.WriteLine($"Result: {smallestRadius.ToString("F2")}");
61. Console.WriteLine($"\nExecution Time: {stopwatch.Elapsed.TotalMilliseconds} ms");
62. }
63. }

**Опис алгоритму виконання:**

Поточну задачу можна змоделювати як пошук мінімального остовного/каркасного дерева графа, де вершинами є міста, а зваженими ребрами - відстані між ними. Вага максимального ребра в МКД буде мінімальним радіусом R, необхідним для радіостанцій.

Хід дії алгоритму:

- Ініціалізація:

Створюється масив minDistances для зберігання мінімальної відстані кожного міста від остовного дерева. Далі створюється логічний масив для відстеження того, чи входить місто до остовного дерева.

- Обробка кожного міста:

Для кожного міста обирається місто, яке ще не включено до МКД і яке має мінімальну відстань від остовного дерева.

Далі включаємо це місто до МКД й оновлюємо maxEdge, якщо відстань більша за поточну maxEdge.

Оновлюємо масив minDistances для всіх міст, які ще не включено до МКД. Нова відстань для міста - це мінімум поточної відстані та відстані до нового міста.

Після того, як всі міста будуть включені до MST, maxEdge буде мінімальним радіусом R, необхідним для радіостанцій.

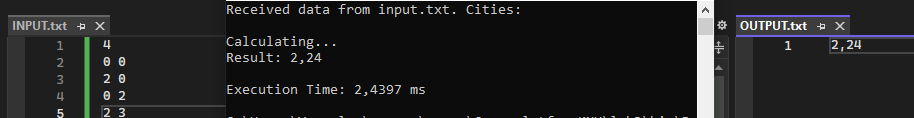
Як можна зрозуміти з вищенаведеного коду та пояснень – було використано алгоритм Пріма. Це не єдиний варіант вирішення поточної задачі. Хоча, наприклад, алгоритм Крускала також можна було б використати і він однаково добре підходить для знаходження МКД, детермінованість і простота алгоритму Прима роблять його дещо кращим вибором для цієї конкретної задачі. **Результати виконання програми:**

Для більшої наочності в програму було додано обчислення часу, що витрачається на виконання обрахунку.  
  
  
1. Input:  
4

0 0

2 0

0 2

2 3  
Output = 2.24  
Час виконання = 2.44 мс  
  
  
  
2. Input:  
3

2 0

0 2

4 2  
Output = 2.83  
Час виконання = 2,25 мс  
