Nama : Andyan Yogawardhana

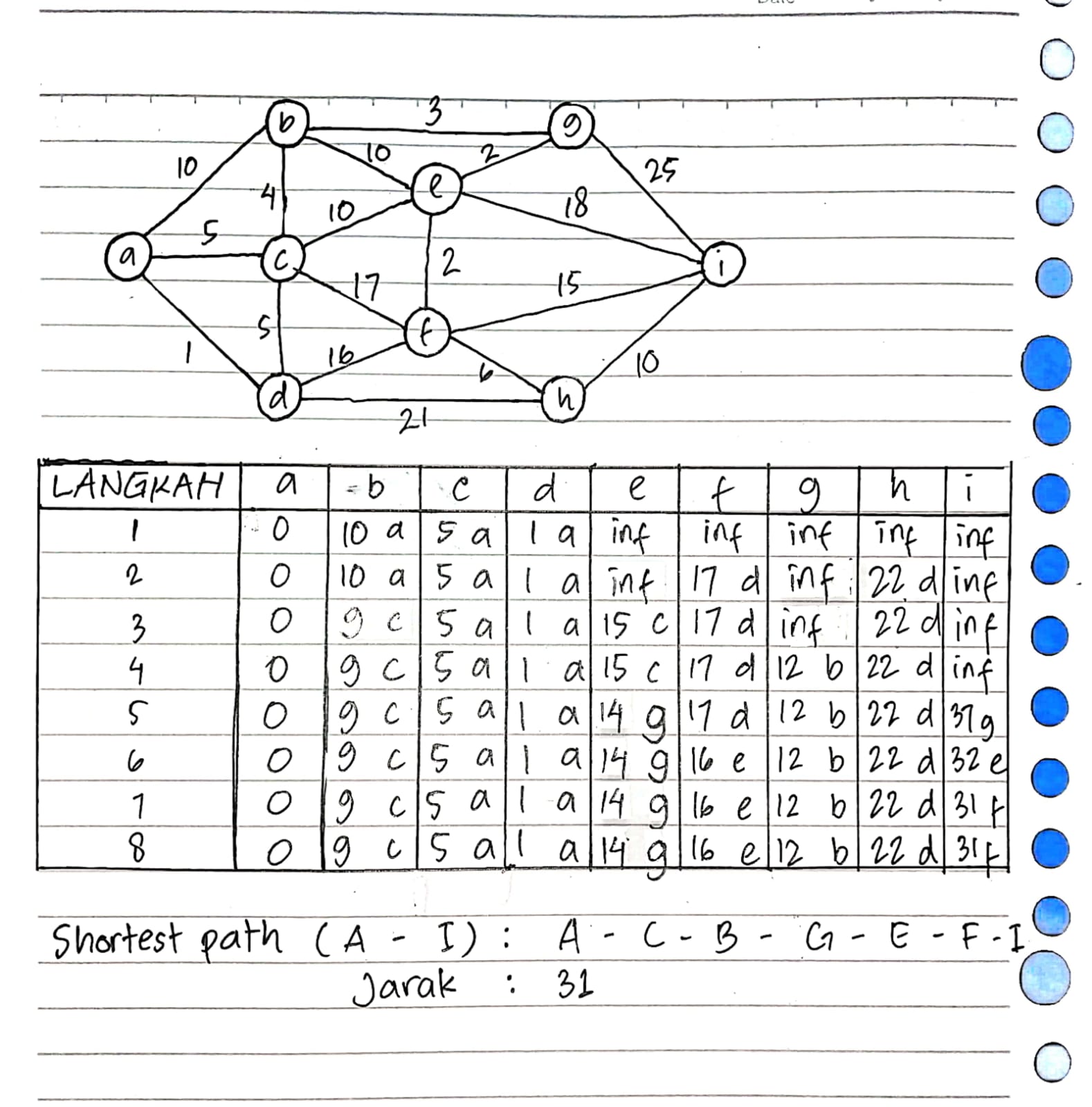
NIM : 21/482180/PA/21030

Kelas : KOMB1

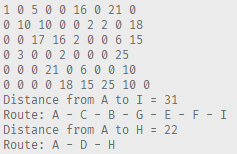
Tugas 7 – Dijkstra

Source Code

1. class package Adjacency;
2. class Graph {
3. private int numVertex;
4. private int[][] adjMatrix;
5. private char charSrc, charDst; // variabel untuk menyimpan index char vertex yang akan diubah dari integer
6. public Graph(int numVertex) {
7. this.numVertex = numVertex;
8. adjMatrix = new int[numVertex][numVertex];
9. }
10. public void addEdge(int from, int to, int weight) {
11. adjMatrix[from][to] = weight;
12. adjMatrix[to][from] = weight;
13. }
14. public void displayGraph() {
15. for(int i = 0; i < numVertex; i++) {
16. for(int j = 0; j < numVertex; j++) {
17. System.out.print(adjMatrix[i][j] + " ");
18. }
19. System.out.println();
20. }
21. }
22. public void dijkstra(int src, int dst) {
23. int[] distance = new int[numVertex];
24. int[] parent = new int[numVertex]; // array untuk menyimpan vertex dengan urutan sebelum vertex fixed dari path
25. boolean[] fixed = new boolean[numVertex];
26. for(int i = 0; i < numVertex; i++) {
27. distance[i] = Integer.MAX\_VALUE;
28. fixed[i] = false;
29. }
30. distance[src] = 0;
31. while(true) {
32. int marked = minIndex(distance, fixed);
33. if(marked < 0) break;
34. if(distance[marked] == Integer.MAX\_VALUE) break;
35. fixed[marked] = true;
36. for(int j = 0; j < numVertex; j++) {
37. if(adjMatrix[marked][j] > 0 && !fixed[j]) {
38. int newDistance = distance[marked] + adjMatrix[marked][j];
39. if(newDistance < distance[j]) {
40. distance[j] = newDistance;
41. parent[j] = marked; // assign nilai marked ke array parent
42. }
43. }
44. }
45. }
47. int x; // variabel untuk menyimpan nilai integer yang akan diubah ke char dengan ASCII
48. for(int i = 0; i < 9; i++) {
49. if(src == i) {
50. x = i + 65; // mencari char dari index ASCII dimulai dari huruf A kapital (65)
51. charSrc = (char)x; // casting tipe data dari int ke char
52. }
53. if(dst == i) {
54. x = i + 65; // mencari char dari index ASCII dimulai dari huruf A kapital (65)
55. charDst = (char)x; // casting tipe data dari int ke char
56. }
57. if(parent[i] == i) {
58. x = i + 65; // mencari char dari index ASCII dimulai dari huruf A kapital (65)
59. parent[i] = (char)x; // casting tipe data dari int ke char
60. }
61. }
63. if(distance[dst] == Integer.MAX\_VALUE) {
64. System.out.println("no route");
65. } else {
66. System.out.println("Distance from " + charSrc + " to " + charDst + " = " + distance[dst]);
67. }
68. printPath(src, dst, parent);
69. System.out.println();
70. }
71. // fungsi untuk menampilkan vertex-vertex yang dilewati shortest path
72. private void printPath(int src, int dst, int[] parent) {
73. char[] charPar = new char[numVertex]; // array untuk menyimpan index char vertex parent yang akan diubah dari integer
74. char charDst; // variabel untuk menyimpan index char vertex dst yang akan diubah dari integer
75. if(parent[dst] == src) { // jika nilai parent dari vertex dst sama dengan vertex src
76. parent[dst] += 65; // mencari char dari index ASCII dimulai dari huruf A kapital (65)
77. charPar[dst] = (char)parent[dst]; // casting tipe data dari int ke char
78. int temp = dst + 65; // mencari char dari index ASCII dimulai dari huruf A kapital (65)
79. charDst = (char)temp; // casting tipe data dari int ke char
80. System.out.print("Route: " + charPar[dst] + " - " + charDst);
81. }
82. else {
83. printPath(src, parent[dst], parent); // fungsi rekursif dengan parameter dst adalah parent dari dst itu sendiri
84. dst += 65; // mencari char dari index ASCII dimulai dari huruf A kapital (65)
85. charDst = (char)dst; // casting tipe data dari int ke char
86. System.out.print(" - " + charDst);
87. }
88. }
89. // mencari curr, vertex mana yang akan dicek/dikunjungi
90. public int minIndex(int[] distance, boolean[] fixed) {
91. int idx = 0;
92. // lebih efektif: min heap / priority queue
93. for(;idx < fixed.length; idx++) {
94. if(!fixed[idx]) break;
95. }
96. if(idx == fixed.length) return -1;
97. for(int i = idx + 1; i < fixed.length; i++) {
98. if(!fixed[i] && distance[i] < distance[idx]) idx = i;
99. }
100. return idx;
101. }
102. }
103. public class Dijkstra {
104. public static void main(String[] args) {
105. int numVertex = 9;
106. Graph map = new Graph(numVertex);
107. // (A,B,C,D,E,F,G,H,I) = (0,1,2,3,4,5,6,7,8)
108. // Create the graph here
109. map.addEdge(0, 1, 10);
110. map.addEdge(0, 2, 5);
111. map.addEdge(0, 3, 1);
112. map.addEdge(1, 2, 4);
113. map.addEdge(1, 6, 3);
114. map.addEdge(1, 4, 10);
115. map.addEdge(2, 3, 5);
116. map.addEdge(2, 4, 10);
117. map.addEdge(2, 5, 17);
118. map.addEdge(3, 5, 16);
119. map.addEdge(3, 7, 21);
120. map.addEdge(4, 5, 2);
121. map.addEdge(4, 6, 2);
122. map.addEdge(4, 8, 18);
123. map.addEdge(5, 7, 6);
124. map.addEdge(5, 8, 15);
125. map.addEdge(6, 8, 25);
126. map.addEdge(7, 8, 10);
127. map.displayGraph();
128. map.dijkstra(0, 8); // vertex A ke I
129. map.dijkstra(0, 7); // vertex A ke H
130. }
131. }



Output Terminal



Terdapat beberapa modifikasi terhadap kode yang saya lakukan untuk membuat suatu fungsi sesuai dengan yang diperintahkan. Untuk mengubah index vertex yang semula integer menjadi character agar sesuai dengan graph di soal, saya mengimplementasikan casting tipe data dengan memanfaatkan kode ASCII di line ke-56 hingga line ke-69. Selain itu, metode ini juga saya gunakan di line ke-84 hingga line ke-98. Pada line ke-29 saya mendeklarasikan array parent untuk menampung vertex sebelum vertex ke-i (vertex ke-(i-1)) dalam suatu path. Array ini akan diisi dengan value dari marked vertex seperti pada line ke-50. Kemudian, array ini juga digunakan sebagai parameter untuk fungsi printPath (line ke-83) yang merupakan fungsi untuk menampilkan vertex-vertex yang dilalui shortest path dari vertex src hingga vertex dst dengan memanfaatkan algoritma rekursi. Basis dari fungsi ini adalah jika vertex parent dari vertex dst sama dengan vertex src, maka akan dilakukan print seperti pada line ke-91.