Nama : Andyan Yogawardhana

NIM : 21/482180/PA/21030

Kelas : KOMB1

Tugas 7 – Dijkstra

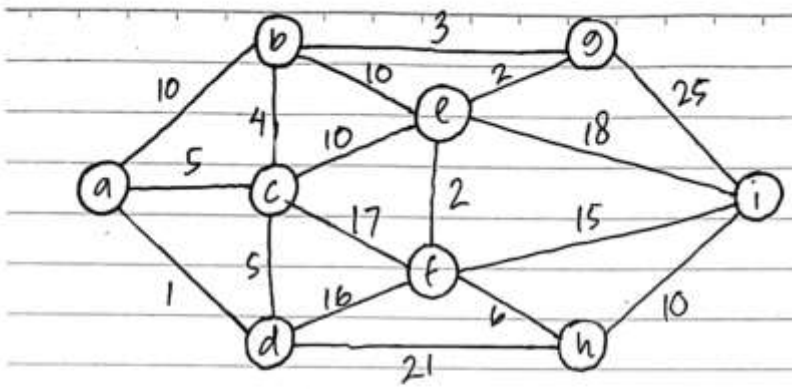
Source Code

```
1  class package Adjacency;
2
3  class Graph {
4      private int numVertex;
5      private int[][] adjMatrix;
6      private char charSrc, charDst;  // variabel untuk menyimpan
   index char vertex yang akan diubah dari integer
7
8      public Graph(int numVertex) {
9          this.numVertex = numVertex;
10         adjMatrix = new int[numVertex][numVertex];
11     }
12
13     public void addEdge(int from, int to, int weight) {
14         adjMatrix[from][to] = weight;
15         adjMatrix[to][from] = weight;
16     }
17
18     public void displayGraph() {
19         for(int i = 0; i < numVertex; i++) {
20             for(int j = 0; j < numVertex; j++) {
21                 System.out.print(adjMatrix[i][j] + " ");
22             }
23             System.out.println();
24         }
25     }
26
27     public void dijkstra(int src, int dst) {
28         int[] distance = new int[numVertex];
29         int[] parent = new int[numVertex];      // array untuk
   menyimpan vertex dengan urutan sebelum vertex fixed dari path
30         boolean[] fixed = new boolean[numVertex];
31
32         for(int i = 0; i < numVertex; i++) {
33             distance[i] = Integer.MAX_VALUE;
34             fixed[i] = false;
35         }
36
```

```
37          distance[src] = 0;
38
39          while(true) {
40              int marked = minIndex(distance, fixed);
41              if(marked < 0) break;
42              if(distance[marked] == Integer.MAX_VALUE) break;
43              fixed[marked] = true;
44
45              for(int j = 0; j < numVertex; j++) {
46                  if(adjMatrix[marked][j] > 0 && !fixed[j]) {
47                      int newDistance = distance[marked] +
    adjMatrix[marked][j];
48                      if(newDistance < distance[j])  {
49                          distance[j] = newDistance;
50                          parent[j] = marked;          // assign
    nilai marked ke array parent
51                      }
52                  }
53              }
54          }
55
56          int x;      // variabel untuk menyimpan nilai integer
    yang akan diubah ke char dengan ASCII
57          for(int i = 0; i < 9; i++) {
58              if(src == i) {
59                  x = i + 65;         // mencari char dari index
    ASCII dimulai dari huruf A kapital (65)
60                  charSrc = (char)x;  // casting tipe data dari
    int ke char
61              }
62              if(dst == i) {
63                  x = i + 65;         // mencari char dari index
    ASCII dimulai dari huruf A kapital (65)
64                  charDst = (char)x;  // casting tipe data dari
    int ke char
65              }
66              if(parent[i] == i) {
67                  x = i + 65;           // mencari char dari index
    ASCII dimulai dari huruf A kapital (65)
68                  parent[i] = (char)x;  // casting tipe data dari
    int ke char
69              }
70          }
71
72          if(distance[dst] == Integer.MAX_VALUE) {
73              System.out.println("no route");
74          } else {
```

```
75              System.out.println("Distance from " + charSrc + " to
   " + charDst + " = " + distance[dst]);
76          }
77
78          printPath(src, dst, parent);
79          System.out.println();
80      }
81
82      // fungsi untuk menampilkan vertex-vertex yang dilewati
   shortest path
83      private void printPath(int src, int dst, int[] parent) {
84          char[] charPar = new char[numVertex];   // array untuk
   menyimpan index char vertex parent yang akan diubah dari integer
85          char charDst;                           // variabel
   untuk menyimpan index char vertex dst yang akan diubah dari
   integer
86          if(parent[dst] == src) {                // jika nilai
   parent dari vertex dst sama dengan vertex src
87              parent[dst] += 65;                  // mencari char
   dari index ASCII dimulai dari huruf A kapital (65)
88              charPar[dst] = (char)parent[dst];   // casting tipe
   data dari int ke char
89              int temp = dst + 65;                // mencari char
   dari index ASCII dimulai dari huruf A kapital (65)
90              charDst = (char)temp;               // casting tipe
   data dari int ke char
91              System.out.print("Route: " + charPar[dst] + " - " +
   charDst);
92          }
93          else {
94              printPath(src, parent[dst], parent);    // fungsi
   rekursif dengan parameter dst adalah parent dari dst itu sendiri
95              dst += 65;                              // mencari
   char dari index ASCII dimulai dari huruf A kapital (65)
96              charDst = (char)dst;                    // casting
   tipe data dari int ke char
97              System.out.print(" - " + charDst);
98          }
99      }
100
101         // mencari curr, vertex mana yang akan
   dicek/dikunjungi
102         public int minIndex(int[] distance, boolean[] fixed) {
103             int idx = 0;
104             // lebih efektif: min heap / priority queue
105             for(;idx < fixed.length; idx++) {
106                 if(!fixed[idx]) break;
107             }
```

```
108
109             if(idx == fixed.length) return -1;
110
111             for(int i = idx + 1; i < fixed.length; i++) {
112                 if(!fixed[i] && distance[i] < distance[idx])
  idx = i;
113             }
114
115             return idx;
116         }
117     }
118
119     public class Dijkstra {
120         public static void main(String[] args) {
121             int numVertex = 9;
122             Graph map = new Graph(numVertex);
123
124             // (A,B,C,D,E,F,G,H,I) = (0,1,2,3,4,5,6,7,8)
125
126             // Create the graph here
127             map.addEdge(0, 1, 10);
128             map.addEdge(0, 2, 5);
129             map.addEdge(0, 3, 1);
130             map.addEdge(1, 2, 4);
131             map.addEdge(1, 6, 3);
132             map.addEdge(1, 4, 10);
133             map.addEdge(2, 3, 5);
134             map.addEdge(2, 4, 10);
135             map.addEdge(2, 5, 17);
136             map.addEdge(3, 5, 16);
137             map.addEdge(3, 7, 21);
138             map.addEdge(4, 5, 2);
139             map.addEdge(4, 6, 2);
140             map.addEdge(4, 8, 18);
141             map.addEdge(5, 7, 6);
142             map.addEdge(5, 8, 15);
143             map.addEdge(6, 8, 25);
144             map.addEdge(7, 8, 10);
145
146             map.displayGraph();
147
148             map.dijkstra(0, 8); // vertex A ke I
149             map.dijkstra(0, 7); // vertex A ke H
150         }
151     }
```



LANGKAH	a	b	c	d	e	f	g	h	i
1	0	10 a	5 a	1 a	inf	inf	inf	inf	inf
2	0	10 a	5 a	1 a	inf	17 d	inf	22 d	inf
3	0	9 c	5 a	1 a	15 c	17 d	inf	22 d	inf
4	0	9 c	5 a	1 a	15 c	17 d	12 b	22 d	inf
5	0	9 c	5 a	1 a	14 g	17 d	12 b	22 d	37 g
6	0	9 c	5 a	1 a	14 g	16 e	12 b	22 d	32 e
7	0	9 c	5 a	1 a	14 g	16 e	12 b	22 d	31 f
8	0	9 c	5 a	1 a	14 g	16 e	12 b	22 d	31 f

Shortest path (A - I) : A - C - B - G - E - F - I

Jarak : 31

Output Terminal

```
1 0 5 0 0 16 0 21 0
0 10 10 0 0 2 2 0 18
0 0 17 16 2 0 0 6 15
0 3 0 0 2 0 0 0 25
0 0 0 21 0 6 0 0 10
0 0 0 0 18 15 25 10 0
Distance from A to I = 31
Route: A - C - B - G - E - F - I
Distance from A to H = 22
Route: A - D - H
```

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.