Experiment 3

Foundations of AI

Dijkstra algorithm

By- Shivansh Jain,VIT Chennai

AIM:-

To find the shortest distance of every node from the starting node using Dijkstra algorithm

Concept:-

The basic principle behind the Dijkstra algorithm is to iteratively look at the node with the currently smallest distance to the source and update all not yet visited neighbors if the path to it *via* the current node is shorter.

ALGORITHM

1. Initialize the distance to the starting node as 0 and the distances to all other nodes as infinite
2. Set all nodes to “unvisited”
3. While we haven’t visited all nodes:
    1. Find the node with currently shortest distance from the source
    2. For all nodes next to it that we haven’t visited yet, check if the currently smallest distance to that neighbor is bigger than if we were to go via the current node
    3. If it is, update the smallest distance of that neighbor to be the distance from the source to the current node plus the distance from the current node to that neighbor

STEP BY STEP IMPLEMENTATION:-

```
#R version 4.1.0

#RStudio version 1.4.1717


rm(list = ls())

#to ensure a clean environment before executing the code


dijkstra <- function(graph, start){

  # function dijkstra with arguments graph and start

  # graph is an adjacency-matrix-representation of the graph where (x,y) is TRUE if there is an edge between nodes x and y
```

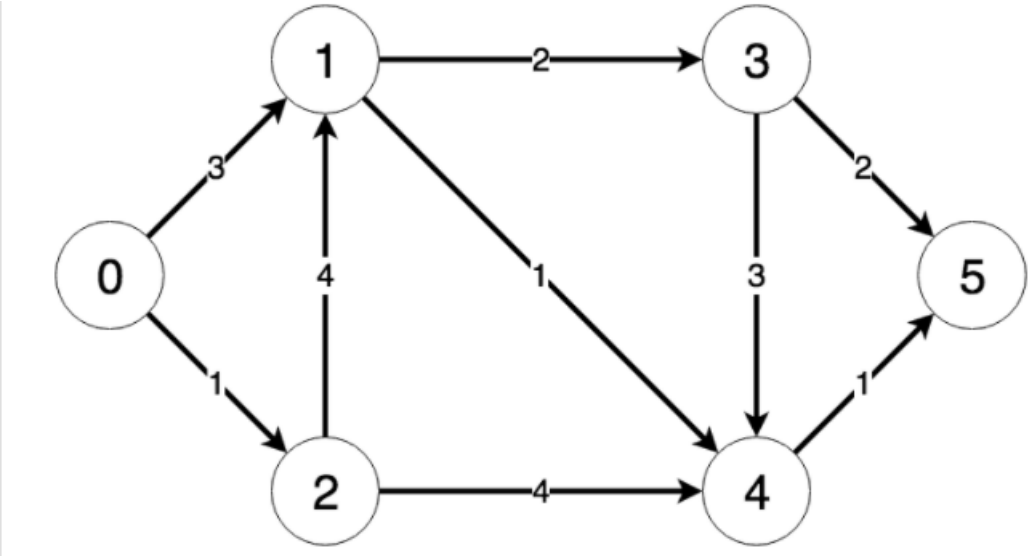
```
# start the node to start from.
# returns an array containing the shortest distance of every node from the starting node


#create array distance and set distance of every node as inf
distances = rep(Inf, nrow(graph))

#create array visited and set status of every node as FALSE
visited = rep(FALSE, nrow(graph))

#Set distance of starting node as 0
distances[start] = 0


# While there are nodes left to visit
repeat{

 # find the node with the currently shortest distance from the start node
 min_distance = Inf
 min_index = -1
 for(i in seq_along(distances)) {
   if(distances[i] < min_distance && !visited[i]){
     min_distance = distances[i]
     min_index = i
   }
 }
```

```
  if(min_index == -1){
    #if there are no nodes to visit
    return (distances)
  }

  #then, for all neighboring nodes that haven't been visited yet
  for(i in seq_along(graph[min_index,])) {
    #if the path over this edge is shorter...
    if(graph[min_index,i] != 0 && distances[i] > distances[min_index] + graph[min_index,i]){
      #Save this path as new shortest path.
      distances[i] = distances[min_index] + graph[min_index,i]
    }
    #set visited as true for the current node
    visited[min_index] = TRUE
  }
 }
}

#driver code

test_case_1 <- matrix(c(0,3,1,0,0,0,0,0,0,2,1,0,0,4,0,0,4,0,0,0,0,0,0,3,2,0,0,0,0,0,0,1,0,0,0,0,0,0), nrow = 6,byrow = TRUE)

test_case_2 <- matrix(c(0, 4, 0, 0, 0, 0, 0, 8, 0,4, 0, 8, 0, 0, 0, 0, 11, 0,0, 8, 0, 7, 0, 4, 0, 0, 2,0, 0, 7, 0, 9, 14, 0, 0, 0,0, 0, 0, 9, 0, 10, 0, 0, 0,0, 0, 4, 14, 10, 0, 2, 0, 0,0, 0, 0, 0, 0, 2, 0, 1, 6,8, 11, 0, 0, 0, 0, 1, 0, 7,0, 0, 2, 0, 0, 0, 6, 7, 0), nrow = 9,byrow = TRUE)

dijkstra(test_case_1,1)

dijkstra(test_case_2,1)
```

RESULTS AND OUTPUT:-

→

Test case 1:-



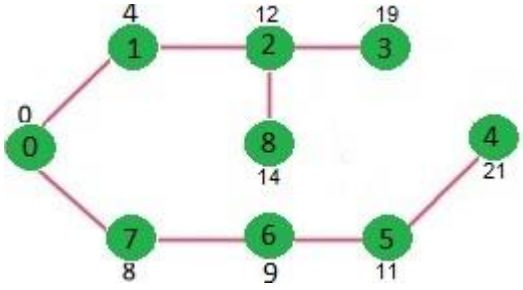
Graph

	V1	V2	V3	V4	V5	V6
1	0	3	1	0	0	0
2	0	0	0	2	1	0
3	0	4	0	0	4	0
4	0	0	0	0	3	2
5	0	0	0	0	0	1
6	0	0	0	0	0	0

Output

```
> test_case_1 <- matrix(c(0,3,1,0,0,0,0,0,0,2,1,0,0,4,0,0,4,0,0,0,0,0,0,3,2,0,0,0,0,0,0,1,0,0,0,0,0,0), nrow = 6,byrow = TRUE)
> dijkstra(test_case_1,1)
[1] 0 3 1 5 4 5
>
```

Test case 2:-



Graph

	V1	V2	V3	V4	V5	V6	V7	V8	V9
1	0	4	0	0	0	0	0	8	0
2	4	0	8	0	0	0	0	11	0
3	0	8	0	7	0	4	0	0	2
4	0	0	7	0	9	14	0	0	0
5	0	0	0	9	0	10	0	0	0
6	0	0	4	14	10	0	2	0	0
7	0	0	0	0	0	2	0	1	6
8	8	11	0	0	0	0	1	0	7
9	0	0	2	0	0	0	6	7	0

Output

```
> test_case_2 <- matrix(c(0, 4, 0, 0, 0, 0, 0, 8, 0,4, 0, 8, 0, 0, 0, 0, 11, 0,0, 8, 0, 7, 0, 4, 0, 0, 2,0, 0, 7, 0, 9, 14, 0, 0, 0,0, 0, 0, 9, 0, 10, 0, 0, 0,
0, 0, 4, 14, 10, 0, 2, 0, 0,0, 0, 0, 0, 0, 2, 0, 1, 6,8, 11, 0, 0, 0, 0, 1, 0, 7,0, 0, 2, 0, 0, 0, 6, 7, 0), nrow = 9,byrow = TRUE)
> dijkstra(test_case_2,1)
[1]  0  4 12 19 21 11  9  8 14
>
```