|  |  |
| --- | --- |
| **NAME:** | Vaishnavi Bhagawan Borkar |
| **UID:** | 2021300016 |
| **SUBJECT** | Data Analysis and Algorithm |
| **EXPERIMENT NO:** | Experiment 6 |
| **DATE OF PERFORMANCE** | 17/3/23 |
| **AIM:** | To find shortest path using Dijkstra’s Algorithm (prim’s algorithm) |
| **THEORY:** | Dijkstra Algorithm is a graph algorithm for finding the shortest path from a source node to all other nodes in a graph (single source shortest path).  It is a type of greedy algorithm. It only works on weighted graphs with positive weights. It has a time complexity of ***O*(*V*2)** using the adjacency matrix representation of graph. The time complexity can be reduced to ***O*((*V*+*E*)*logV*)** using adjacency list representation of graph, where E is the number of edges in the graph and V is the number of vertices in the graph. |
| **ALGORITHM:** | 1 **function** Dijkstra(*Graph*, *source*):  2   1. **for each** vertex *v* in *Graph.Vertices*: 2. dist[*v*] ← INFINITY 3. prev[*v*] ← UNDEFINED 4. add *v* to *Q* 5. *dist[source]* ← 0 6. **while** *Q* is not empty: 7. *u* ← vertex in *Q* with min dist[u] 8. remove u from *Q*   12   1. **for each** neighbor *v* of *u* still in *Q*: 2. *alt* ← dist[*u*] + Graph.Edges(*u*, *v*) 3. **if** *alt* < dist[*v*]: 4. dist[*v*] ← *alt* 5. prev[*v*] ← *u*   18 **return** dist[], prev[] |
| **PROGRAM:** | *#include* <limits.h>  *#include* <stdbool.h>  *#include* <stdio.h>  int V;  int minDistance(int *dist*[], bool *sptSet*[])  {      int min = INT\_MAX, min\_index;  *for* (int v = 0; v < V; v++)  *if* (*sptSet*[v] == false && *dist*[v] <= min)              min = *dist*[v], min\_index = v;  *return* min\_index;  }  void printSolution(int *dist*[])  {      printf("Vertex \t\t Distance from Source\n");  *for* (int i = 0; i < V; i++)          printf("%d \t\t\t\t %d\n", i, *dist*[i]);  }  void dijkstra(int *graph*[V][V], int *src*)  {      int dist[V];      bool sptSet[V];  *for* (int i = 0; i < V; i++)          dist[i] = INT\_MAX, sptSet[i] = false;      dist[*src*] = 0;  *for* (int count = 0; count < V - 1; count++)      {          int u = minDistance(dist, sptSet);          sptSet[u] = true;  *for* (int v = 0; v < V; v++)  *if* (!sptSet[v] && *graph*[u][v] && dist[u] != INT\_MAX && dist[u] + *graph*[u][v] < dist[v])                  dist[v] = dist[u] + *graph*[u][v];      }      printSolution(dist);  }  int main()  {      printf("Enter the order:");      scanf("%d", &V);      int graph[V][V];  *for* (int i = 0; i < V; i++)      {          printf("Elememts of row number %d:", (i + 1));  *for* (int j = 0; j < V; j++)          {              scanf("%d", &graph[i][j]);          }      }      dijkstra(graph, 0);  *return* 0;  } |
| **RESULT:** | |
| **CONCLUSION:** | By performing the above experiment i have successfully found the shortest part of different vertices from a single source using Dijkstra’s algorithm. |