**Assignment 10**

**Name** : Komal Potdar

**Roll No**.: 92

**PRN No**.: 12320165

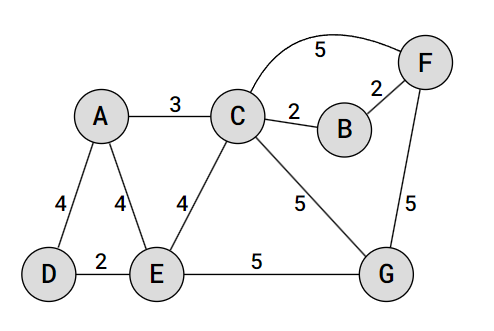
**Div**: CS B SY

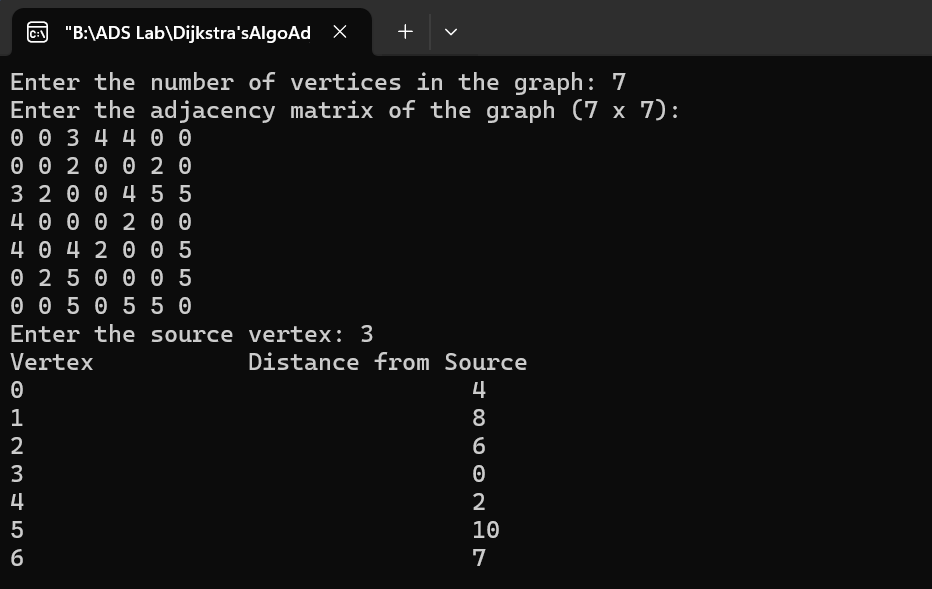
**Batch**: 3

1. **WAP to generate Single Source Shortest Path  using Dijkstras Algorithm when graph is represented by  
   A. Adjacency Matrix**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <stdbool.h>  #include <limits.h>  int minDistance(int dist[], bool sptSet[], int V) {  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[], int V) {  printf("Vertex \t\t Distance from Source\n");  for (int i = 0; i < V; i++) {  if (dist[i] == INT\_MAX)  printf("%d \t\t\t\t Infinite\n", i);  else  printf("%d \t\t\t\t %d\n", i, dist[i]);  }  }  void dijkstra(int \*\*graph, int src, int V) {  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, V);  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, V);  }  int main() {  int V;  printf("Enter the number of vertices in the graph: ");  scanf("%d", &V);  int \*\*graph = (int \*\*)malloc(V \* sizeof(int \*));  for (int i = 0; i < V; i++)  graph[i] = (int \*)malloc(V \* sizeof(int));  printf("Enter the adjacency matrix of the graph (%d x %d):\n", V, V);  for (int i = 0; i < V; i++) {  for (int j = 0; j < V; j++) {  scanf("%d", &graph[i][j]);  }  }  int src;  printf("Enter the source vertex: ");  scanf("%d", &src);  dijkstra(graph, src, V);    for (int i = 0; i < V; i++)  free(graph[i]);  free(graph);  return 0;  } |

Output:





1. **WAP to generate Single Source Shortest Path  using Dijkstras Algorithm when graph is represented by  
   B. Adjacency List**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <stdbool.h>  #include <limits.h>  #include <string.h>  typedef struct Node {  int dest;  int weight;  struct Node\* next;  } Node;  typedef struct Graph {  int V;  Node\*\* array;  } Graph;  Node\* createNode(int dest, int weight) {  Node\* newNode = (Node\*)malloc(sizeof(Node));  newNode->dest = dest;  newNode->weight = weight;  newNode->next = NULL;  return newNode;  }  Graph\* createGraph(int V) {  Graph\* graph = (Graph\*)malloc(sizeof(Graph));  graph->V = V;  graph->array = (Node\*\*)malloc(V \* sizeof(Node\*));  for (int i = 0; i < V; ++i)  graph->array[i] = NULL;  return graph;  }  void addEdge(Graph\* graph, char\* src, char\* dest, int weight) {  int srcIndex = src[0] - 'A';  int destIndex = dest[0] - 'A';    Node\* newNode = createNode(destIndex, weight);  newNode->next = graph->array[srcIndex];  graph->array[srcIndex] = newNode;    newNode = createNode(srcIndex, weight);  newNode->next = graph->array[destIndex];  graph->array[destIndex] = newNode;  }  int minDistance(int dist[], bool sptSet[], int V) {  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[], int V) {  printf("Vertex \t\t Distance from Source\n");  for (int i = 0; i < V; i++) {  if (dist[i] == INT\_MAX)  printf("%d \t\t\t\t Infinite\n", i);  else  printf("%d \t\t\t\t %d\n", i, dist[i]);  }  }  void dijkstra(Graph\* graph, int src) {  int V = graph->V;  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, V);  sptSet[u] = true;  Node\* tmp = graph->array[u];  while (tmp != NULL) {  int v = tmp->dest;  int weight = tmp->weight;  if (!sptSet[v] && dist[u] != INT\_MAX && dist[u] + weight < dist[v])  dist[v] = dist[u] + weight;  tmp = tmp->next;  }  }  printSolution(dist, V);  }  int main() {  int V, E;  printf("Enter the number of vertices in the graph: ");  scanf("%d", &V);  Graph\* graph = createGraph(V);  printf("Enter the number of edges: ");  scanf("%d", &E);  printf("Enter the edges as source destination weight:\n");  for (int i = 0; i < E; i++) {  char src[2], dest[2];  int weight;  scanf("%s %s %d", src, dest, &weight);  addEdge(graph, src, dest, weight);  }  char src[2];  printf("Enter the source vertex: ");  scanf("%s", src);  int srcIndex = src[0] - 'A';  dijkstra(graph, srcIndex);    for (int i = 0; i < V; i++) {  Node\* tmp = graph->array[i];  while (tmp != NULL) {  Node\* prev = tmp;  tmp = tmp->next;  free(prev);  }  }  free(graph->array);  free(graph);  return 0;  } |

Output:

