```
//Name: 8. Graph: Shortest Path Algorithm
#include <iostream>
 #include <limits.h>
 using namespace std;
 const int MAX_VERTICES = 10;
int minDistance(int dist[], bool sptSet[], int V) {
   int min = INT_MAX, min_index;
   for (int v = 0; v < V; v++) {
      if (!sptSet[v] && dist[v] <= min) {
         min = dist[v];
         min_index = v;
      }
   }
   return min_index;
 }
void dijkstra(int graph[MAX_VERTICES][MAX_VERTICES], int src, int V) {
   int dist[MAX_VERTICES];
   bool sptSet[MAX_VERTICES];
   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++) {processed
```

```
        int u = minDistance(dist, sptSet, V);
        sptSet[u] = true;
        for (int v = 0; v < V; v++) {
from u to v,
than the current value of dist[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];
            }
        }
    }

    cout << "Vertex\tDistance from Source (A)\n";
    for (int i = 0; i < V; i++) {
        cout << char(i + 'A') << "\t" << dist[i] << endl;
    }
}
 int main() {
    const int V = 4;
    int graph[MAX_VERTICES][MAX_VERTICES] = {
        {0, 10, 0, 5},
        {0, 0, 1, 2},
        {0, 0, 0, 0},
        {0, 3, 9, 0}
```

```
    };
    dijkstra(graph, 0, V);
    return 0;
}
```