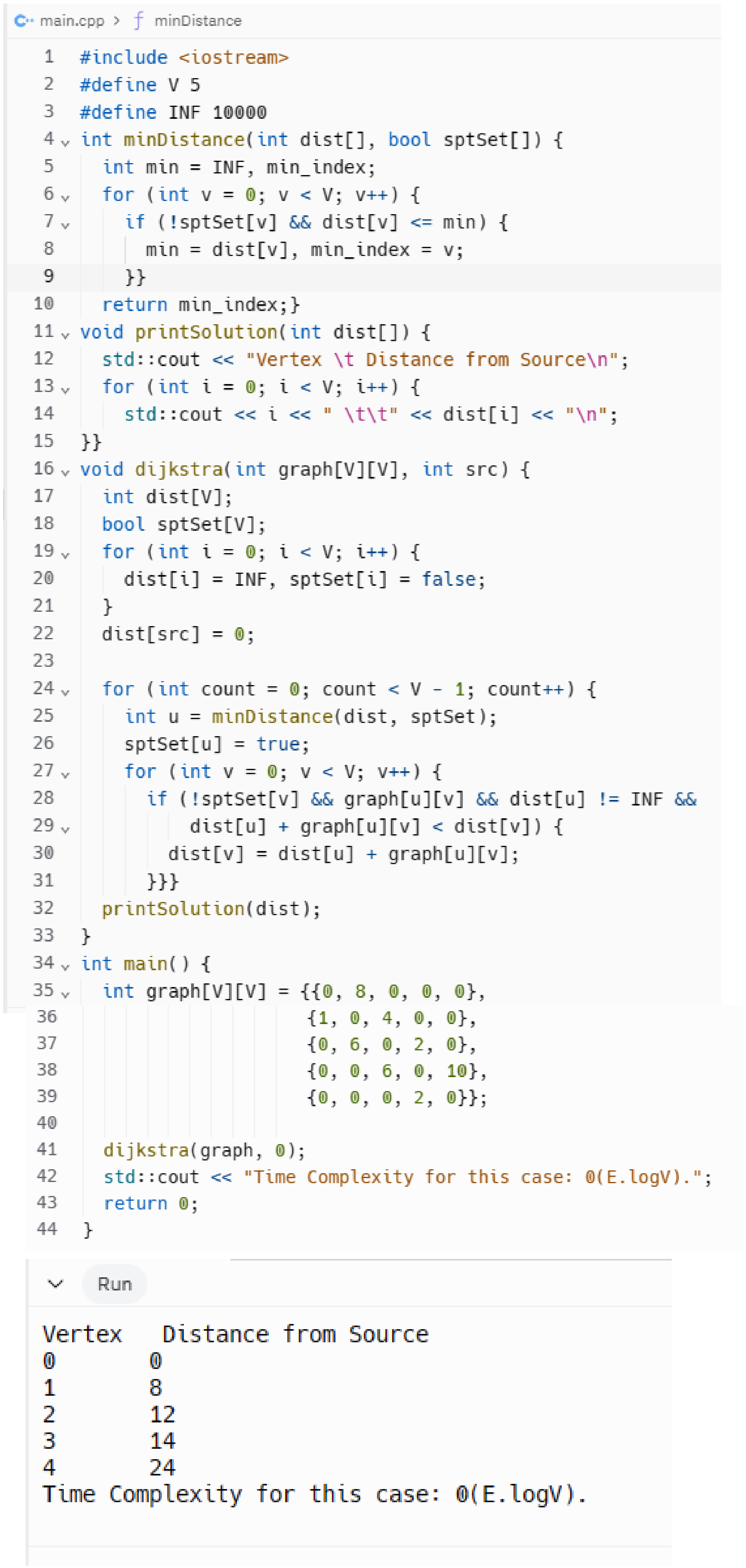

```cpp
#include <iostream>
#define V 5
#define INF 10000
int minDistance(int dist[], bool sptSet[]) {
  int min = INF, min_index;
  for (int v = 0; v < V; v++) {
    if (!sptSet[v] && dist[v] <= min) {
      min = dist[v], min_index = v;
    }}
  return min_index;}
void printSolution(int dist[]) {
  std::cout << "Vertex \t Distance from Source\n";
  for (int i = 0; i < V; i++) {
    std::cout << i << " \t\t" << dist[i] << "\n";
}}
void dijkstra(int graph[V][V], int src) {
  int dist[V];
  bool sptSet[V];
  for (int i = 0; i < V; i++) {
    dist[i] = INF, 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] != INF &&
          dist[u] + graph[u][v] < dist[v]) {
        dist[v] = dist[u] + graph[u][v];
      }}}
  printSolution(dist);
}
int main() {
  int graph[V][V] = {{0, 8, 0, 0, 0},
                     {1, 0, 4, 0, 0},
                     {0, 6, 0, 2, 0},
                     {0, 0, 6, 0, 10},
                     {0, 0, 0, 2, 0}};

  dijkstra(graph, 0);
  std::cout << "Time Complexity for this case: O(E.logV).";
  return 0;
}
```

```
Vertex   Distance from Source
0        0
1        8
2        12
3        14
4        24
Time Complexity for this case: O(E.logV).
```