```cpp
//Dijkstra para matriz de adjacencias
int dijkstra(int ag[MAX][MAX], int s, int t){
  int dist[MAX], vis[MAX];
  for(int i = 0; i < n; i++) { dist[i] = INF; vis[i] = 0; }
  dist[s] = 0;
  priority_queue<ii, vector<ii>, greater<ii>> q;
  q.push(ii(dist[s], s));
  while(!q.empty()){
    int current = q.top().second;
    q.pop();
    if(vis[current]) continue;
    vis[current] = 1;
    for(int i = 0; i < n; i++)
      if(ag[current][i] != -1 && !vis[i] && dist[i] > dist[current] + ag[current][i]){
        dist[i] = dist[current] + ag[current][i];
        q.push(ii(dist[i], i));
      }
    if(current == t) return dist[t];
  }
  return dist[t];
}

//Dijkstra para lista de adjacencias
int dijkstra(int s, int t) {
  int dist[MAXC], vis[MAXC], i;
  memset(vis, 0, sizeof(vis));
  for(i = 0; i < 2 * c; i++) dist[i] = INF;
  dist[s] = 0;
  priority_queue<pi, vpi, greater<pi>> kiwi;
  kiwi.push(make_pair(dist[s], s));
  while(!kiwi.empty()) {
    pi current = kiwi.top();
    kiwi.pop();
    if(vis[current.second]) continue;
    vis[current.second] = 1;
    for(i = 0; i < (int) adj[current.second].size(); i++) {
      int v = adj[current.second][i].second;
      int vpeso = adj[current.second][i].first;
      if(!vis[v] && dist[v] > dist[current.second] + vpeso) {
        dist[v] = dist[current.second] + vpeso;
        kiwi.push(make_pair(dist[v], v));
      }
    }
  }
  return dist[t] == INF ? -1 : dist[t];
}
```