|  |  |
| --- | --- |
| **Min Cost to collect all cities in C++** | |
| #include <iostream>  #include <vector>  #include <queue>  using namespace std;  struct Edge {  int v;  int wt;  Edge(int nbr, int weight) {  this->v = nbr;  this->wt = weight;  }  };  struct CompareEdge {  bool operator()(const Edge& e1, const Edge& e2) {  return e1.wt > e2.wt; // Min-Heap based on edge weight  }  };  int main() {  // Hardcoded input  int vtces = 7;  int edges = 8;  vector<vector<Edge>> graph(vtces);  // Hardcoded edges  vector<vector<int>> hardcoded\_edges = {  {0, 1, 10},  {1, 2, 10},  {2, 3, 10},  {0, 3, 40},  {3, 4, 2},  {4, 5, 3},  {5, 6, 3},  {4, 6, 8}  };  // Populating the graph with hardcoded edges  for (auto& edge : hardcoded\_edges) {  int v1 = edge[0];  int v2 = edge[1];  int wt = edge[2];  graph[v1].emplace\_back(v2, wt);  graph[v2].emplace\_back(v1, wt);  }  int ans = 0;  priority\_queue<Edge, vector<Edge>, CompareEdge> pq;  vector<bool> vis(vtces, false);  pq.push(Edge(0, 0)); // Start with any vertex (0 in this case) with 0 weight  while (!pq.empty()) {  Edge rem = pq.top();  pq.pop();  if (vis[rem.v]) {  continue;  }  vis[rem.v] = true;  ans += rem.wt;  for (Edge nbr : graph[rem.v]) {  if (!vis[nbr.v]) {  pq.push(nbr);  }  }  }  cout << ans << endl;  return 0;  } | ****Core Concepts in the Code:****  * Uses a **priority queue (min-heap)** to always pick the edge with the **least weight**. * Starts from vertex 0. * Adds edge weights to the total MST weight only when visiting **unvisited vertices**. * vis[] tracks visited vertices.  📊 ****Hardcoded Graph (7 vertices, 8 edges):**** Edges:  {v1,v2,wt}  {0, 1, 10}  {1, 2, 10}  {2, 3, 10}  {0, 3, 40}  {3, 4, 2}  {4, 5, 3}  {5, 6, 3}  {4, 6, 8} 🧾 ****Dry Run Table: Prim's MST****  | **Step** | **Vertex Visited** | **Edge Added (from)** | **Weight Added** | **Total MST Weight** | **Priority Queue (next min weight edges)** | | --- | --- | --- | --- | --- | --- | | 1 | 0 | - (start) | 0 | 0 | (1,10), (3,40) | | 2 | 1 | 0 → 1 | 10 | 10 | (2,10), (3,40) | | 3 | 2 | 1 → 2 | 10 | 20 | (3,10), (3,40) | | 4 | 3 | 2 → 3 | 10 | 30 | (4,2), (3,40) | | 5 | 4 | 3 → 4 | 2 | 32 | (5,3), (6,8), (3,40) | | 6 | 5 | 4 → 5 | 3 | 35 | (6,3), (6,8), (3,40) | | 7 | 6 | 5 → 6 | 3 | 38 | (6,8), (3,40) → both discarded (visited) |  ✅ ****MST Total Weight:**** 38 Even though there's a 40-weight edge from 0 to 3, we never pick it because we reach 3 through a cheaper path (0→1→2→3). 🖨️ ****Output:**** 38 |
| Output:- 38 | |