|  |  |
| --- | --- |
| **Print All Paths in C++** | |
| #include <iostream>  #include <vector>  using namespace std;  // Define the Edge structure  struct Edge {  int src;  int nbr;  int wt;  Edge(int s, int n, int w) {  src = s;  nbr = n;  wt = w;  }  };  // Function prototypes  void addEdge(vector<Edge>\* graph, int src, int nbr, int wt);  void printAllPaths(vector<Edge>\* graph, int src, int dest, vector<bool>& visited, string psf);  int main() {  int vtces = 6; // Number of vertices  //int edges = 7; // Number of edges  // Create the graph using vector of vectors  vector<Edge>\* graph = new vector<Edge>[vtces];  // Add edges statically  addEdge(graph, 0, 1, 10);  addEdge(graph, 0, 3, 40);  addEdge(graph, 1, 2, 10);  addEdge(graph, 2, 3, 10);  addEdge(graph, 3, 4, 2);  addEdge(graph, 4, 5, 2);  addEdge(graph, 2, 4, 3);  int src = 0; // Source vertex  int dest = 5; // Destination vertex  // Array to track visited vertices  vector<bool> visited(vtces, false);  // Call the function to print all paths from src to dest  printAllPaths(graph, src, dest, visited, to\_string(src));  return 0;  }  // Function to add an edge to the graph  void addEdge(vector<Edge>\* graph, int src, int nbr, int wt) {  graph[src].emplace\_back(src, nbr, wt);  graph[nbr].emplace\_back(nbr, src, wt);  }  // Function to print all paths from src to dest  void printAllPaths(vector<Edge>\* graph, int src, int dest, vector<bool>& visited, string psf) {  if (src == dest) {  cout << psf << endl;  return;  }  visited[src] = true;  for (Edge edge : graph[src]) {  if (!visited[edge.nbr]) {  printAllPaths(graph, edge.nbr, dest, visited, psf + to\_string(edge.nbr));  }  }  visited[src] = false;  } | ****Graph Structure****:Edges: 0 -- 1 (10)  0 -- 3 (40)  1 -- 2 (10)  2 -- 3 (10)  3 -- 4 (2)  4 -- 5 (2)  2 -- 4 (3)  This gives us the adjacency list:   | **Vertex** | **Neighbors** | | --- | --- | | 0 | 1, 3 | | 1 | 0, 2 | | 2 | 1, 3, 4 | | 3 | 0, 2, 4 | | 4 | 3, 5, 2 | | 5 | 4 |  🎯 ****Goal:**** Find all paths from src = 0 to dest = 5. 🧾 Dry Run Table:  | **Recursive Call** | **Current src** | **Path So Far (psf)** | **Action** | | --- | --- | --- | --- | | 1 | 0 | "0" | Explore neighbors 1, 3 | | 2 | 1 | "01" | Explore neighbors 2 | | 3 | 2 | "012" | Explore 3, 4 | | 4 | 3 | "0123" | Explore 4 | | 5 | 4 | "01234" | Explore 5 | | 6 | 5 | "012345" | ✅ **Print this path** | | Backtrack to 4 |  |  |  | | Backtrack to 3 |  |  |  | | 4 (alt) | 4 | "0124" | Explore 5 | | 5 | 5 | "01245" | ✅ **Print this path** | | Backtrack to 2 |  |  |  | | Backtrack to 1 |  |  |  | | Backtrack to 0 |  |  |  | | 2 | 3 | "03" | Explore 2, 4 | | 3 | 2 | "032" | Explore 4 | | 4 | 4 | "0324" | Explore 5 | | 5 | 5 | "03245" | ✅ **Print this path** | | Backtrack to 3 |  |  |  | | 3 (alt) | 4 | "034" | Explore 5 | | 4 | 5 | "0345" | ✅ **Print this path** |  ✅ ****Final Output****: 012345  01245  03245  0345 |
| Output:-  012345  01245  03245  0345 | |