**DATA STRUCTURE**

**PROGRAMS:**

**1.Prim’s Algorithm**

*#include <stdio.h>*

*#include <limits.h>*

*#include <stdbool.h>*

*#define V 5*

*int minKey(int key[], bool mstSet[]) {*

*int min = INT\_MAX, min\_index;*

*for (int v = 0; v < V; v++)*

*if (mstSet[v] == false && key[v] < min)*

*min = key[v], min\_index = v;*

*return min\_index;*

*}*

*void printMST(int parent[], int graph[V][V]) {*

*printf("Edge \tWeight\n");*

*for (int i = 1; i < V; i++)*

*printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);*

*}*

*void primMST(int graph[V][V]) {*

*int parent[V];*

*int key[V];*

*bool mstSet[V];*

*for (int i = 0; i < V; i++)*

*key[i] = INT\_MAX, mstSet[i] = false;*

*key[0] = 0;*

*parent[0] = -1;*

*for (int count = 0; count < V - 1; count++) {*

*int u = minKey(key, mstSet);*

*mstSet[u] = true;*

*for (int v = 0; v < V; v++)*

*if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v])*

*parent[v] = u, key[v] = graph[u][v];*

*}*

*printMST(parent, graph);*

*}*

*int main() {*

*int graph[V][V] = {{0, 2, 0, 6, 0},*

*{2, 0, 3, 8, 5},*

*{0, 3, 0, 0, 7},*

*{6, 8, 0, 0, 9},*

*{0, 5, 7, 9, 0}};*

*primMST(graph);*

*return 0;*

*}*

**OUTPUT:**

*Edge Weight*

*0 - 1 2*

*1 - 2 3*

*0 - 3 6*

*1 - 4 5*

**2.Krushal’s Algorithm**

*#include <stdio.h>*

*#include <stdlib.h>*

*#define V5*

*struct Edge {*

*int src, dest, weight;*

*};*

*struct Graph {*

*int V, E*

*struct Edge\* edge;*

*};*

*struct subset {*

*int parent;*

*int rank;*

*};*

*struct Graph\* createGraph(int V, int E) {*

*struct Graph\* graph = (struct Graph\*) malloc(sizeof(struct Graph));*

*graph->V = V;*

*graph->E = E;*

*graph->edge = (struct Edge\*) malloc(graph->E \* sizeof(struct Edge));*

*return graph;*

*}*

*int find(struct subset subsets[], int i) {*

*if (subsets[i].parent != i)*

*subsets[i].parent = find(subsets, subsets[i].parent);*

*return subsets[i].parent;*

*}*

*void Union(struct subset subsets[], int x, int y) {*

*int rootX = find(subsets, x);*

*int rootY = find(subsets, y);*

*if (subsets[rootX].rank < subsets[rootY].rank)*

*subsets[rootX].parent = rootY;*

*else if (subsets[rootX].rank > subsets[rootY].rank)*

*subsets[rootY].parent = rootX;*

*else {*

*subsets[rootY].parent = rootX;*

*subsets[rootX].rank++;*

*}*

*}*

*int compareEdges(const void\* a, const void\* b) {*

*struct Edge\* a1 = (struct Edge\*) a;*

*struct Edge\* b1 = (struct Edge\*) b;*

*return a1->weight > b1->weight;*

*}*

*void KruskalMST(struct Graph\* graph) {*

*int V = graph->V;*

*struct Edge result[V];*

*int e = 0;*

*int i = 0;*

*qsort(graph->edge, graph->E, sizeof(graph->edge[0]), compareEdges);*

*struct subset\* subsets = (struct subset\*) malloc(V \* sizeof(struct subset));*

*for (int v = 0; v < V; ++v) {*

*subsets[v].parent = v;*

*subsets[v].rank = 0;*

*}*

*while (e < V - 1 && i < graph->E) {*

*struct Edge next\_edge = graph->edge[i++];*

*int x = find(subsets, next\_edge.src);*

*int y = find(subsets, next\_edge.dest);*

*if (x != y) {*

*result[e++] = next\_edge;*

*Union(subsets, x, y);*

*}*

*}*

*printf("Edge \tWeight\n");*

*for (i = 0; i < e; ++i)*

*printf("%d - %d \t%d \n", result[i].src, result[i].dest, result[i].weight);*

*free(subsets);*

*}*

*int main() {*

*/\* Example graph:*

*0*

*(0)--(1)*

*|6 / \8*

*| / \*

*(2) (3)*

*| \ /*

*|7 \5 /9*

*(4)---(5)*

*\*/*

*int V = 4;*

*int E = 5;*

*struct Graph\* graph = createGraph(V, E);*

*graph->edge[0].src = 0;*

*graph->edge[0].dest = 1;*

*graph->edge[0].weight = 10;*

*graph->edge[1].src = 0;*

*graph->edge[1].dest = 2;*

*graph->edge[1].weight = 6;*

*graph->edge[2].src = 0;*

*graph->edge[2].dest = 3;*

*graph->edge[2].weight = 5;*

*graph->edge[3].src = 1;*

*graph->edge[3].dest = 3;*

*graph->edge[3].weight = 15;*

*graph->edge[4].src = 2;*

*graph->edge[4].dest = 3;*

*graph->edge[4].weight = 4;*

*KruskalMST(graph);*

*free(graph->edge);*

*free(graph);*

*return 0;*

*}*

**OUTPUT:**

*Edge Weight*

*2 - 3 4*

*0 - 3 5*

*0 - 2 6*