Design & Analysis
Of Algorithm
Pranav .N
CH.SC.U4CSE24236

1.Prim’s Algorithm:~

CODE:

```
#include <stdio.h>
#define INF 999999   // A large number to represent "no edge"
#define MAX 20       // Maximum number of vertices
void primMST(int graph[MAX][MAX], int n) {
   int selected[MAX];   // selected[i] = 1 if vertex i is in MST
   int i, j, u, v;
   int edges = 0;
   int minCost = 0;
  for (i = 0; i < n; i++)
   selected[i] = 0;
   selected[0] = 1;
   printf("Edges in Minimum Spanning Tree:\n");
   // MST will have (n - 1) edges
   while (edges < n - 1) {
       int min = INF;
       u = v = -1;
       for (i = 0; i < n; i++) {
           if (selected[i]) {
               for (j = 0; j < n; j++) {
                   if (!selected[j] && graph[i][j] && graph[i][j] < min) {
                       min = graph[i][j];
                       u = i;
                       v = j;
                   }
               }
           }
       }
       selected[v] = 1;
       edges++;
       minCost += min;
       printf("%d -- %d  (weight = %d)\n", u, v, min);
   }
   printf("Total cost of MST = %d\n", minCost);
}
int main() {
   int n, i, j;
   int graph[MAX][MAX];
   printf("Enter number of vertices: ");
```

Design & Analysis
Of Algorithm
Pranav .N
CH.SC.U4CSE24236

```
    scanf("%d", &n);
    printf("Enter adjacency matrix (0 if no edge):\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", &graph[i][j]);
            if (graph[i][j] == 0 && i != j)
                graph[i][j] = INF;   // replace 0 with INF (no direct edge)
        }
    }
    primMST(graph, n);
    return 0;
}
```

OUTPUT:

```
pranavneelakandan@Pranavs-MacBook-Air DAA % cd "/Users/pranavneelakandan/DAA/" && gcc tempCodeRunnerFile.c
mpCodeRunnerFile && "/Users/pranavneelakandan/DAA/"tempCodeRunnerFile
Enter number of vertices: 5
Enter adjacency matrix (0 if no edge):
0 1 0 1 0
1 0 1 0 0
1 1 1 1 0
1 0 1 0 1
1 1 1 1 1
Edges in Minimum Spanning Tree:
0 -- 1  (weight = 1)
0 -- 3  (weight = 1)
1 -- 2  (weight = 1)
3 -- 4  (weight = 1)
Total cost of MST = 4
pranavneelakandan@Pranavs-MacBook-Air DAA %
```

Interpretation:

Prim’s algorithm grows a minimum spanning tree by standing from any one vertex , then repeatedly adding the cheapest edge that connects the current tree to a new vertex without making a cycle , until all vertices are included.

Design & Analysis
Of Algorithm
Pranav .N
CH.SC.U4CSE24236

2.Kruskal's Algorithm:~
CODE:

```
#include <stdio.h>
#define MAXE 100   // max edges
#define MAXV 20    // max vertices
typedef struct {
   int u, v;   // endpoints
   int w;      // weight
} Edge;
Edge edges[MAXE];
int parent[MAXV];
// Find with path compression
int find(int x) {
   if (parent[x] == x)
      return x;
   return parent[x] = find(parent[x]);
}
// Union of two sets
void unite(int a, int b) {
   a = find(a);
   b = find(b);
   if (a != b)
      parent[b] = a;
}
// Simple bubble sort edges by weight (for clarity, not speed)
void sortEdges(int m) {
   int i, j;
   for (i = 0; i < m - 1; i++) {
      for (j = 0; j < m - 1 - i; j++) {
         if (edges[j].w > edges[j + 1].w) {
            Edge temp = edges[j];
            edges[j] = edges[j + 1];
            edges[j + 1] = temp;
         }
      }
```

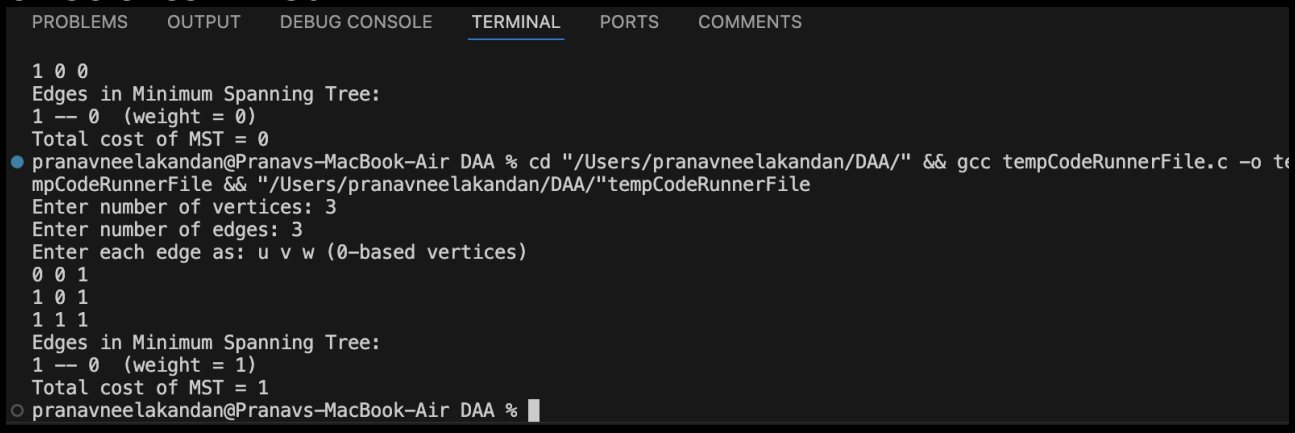
Design & Analysis
Of Algorithm
Pranav .N
CH.SC.U4CSE24236

```
    }
}
int main() {
    int n, m;
    int i;
    int mstCost = 0;
    int edgesUsed = 0;

    printf("Enter number of vertices: ");
    scanf("%d", &n);
    printf("Enter number of edges: ");
    scanf("%d", &m);
    printf("Enter each edge as: u v w (0-based vertices)\n");
    for (i = 0; i < m; i++) {
        scanf("%d %d %d", &edges[i].u, &edges[i].v, &edges[i].w);
    }
// Initialize disjoint sets
    for (i = 0; i < n; i++)
        parent[i] = i;
    sortEdges(m);
    printf("Edges in Minimum Spanning Tree:\n");
    for (i = 0; i < m && edgesUsed < n - 1; i++) {
        int u = edges[i].u;
        int v = edges[i].v;
        int w = edges[i].w;
        // If u and v are in different sets, include this edge
        if (find(u) != find(v)) {
            printf("%d -- %d  (weight = %d)\n", u, v, w);
            mstCost += w;
            edgesUsed++;
            unite(u, v);
        }
    }
    printf("Total cost of MST = %d\n", mstCost);
    return 0;
}
```

OUTPUT:

Design & Analysis
Of Algorithm
Pranav .N
CH.SC.U4CSE24236



Interpretation:

Kruskal’s algorithm builds a minimum spanning tree by sorting all edges by weight and then repeatedly taking the next cheapest edge that does not form a cycle, effectively joining smaller trees into a single tree (or forest) step by step.