11/3/24 Week-10

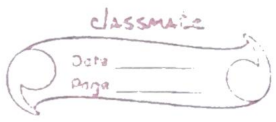
Dijkstras

```
#include<stdio.h>
#include<limits.h>
#define MAX_VERTICES 100
void dijkstras(int c[MAX_VERTICES][MAX_VERTICES], int n,
        int src) {
    int dist[MAX_VERTICES];
    int vis[MAX_VERTICES];
    int count, u, i, j;
    for(i=0; i<n; i++) {
        dist[i] = INT_MAX;
        vis[i] = 0;
    }
    dist[src] = 0;
    count = 0;
    while(count < n-1) {
        int min_dist = INT_MAX;
        for(i=0; i<n; i++) {
            if(!vis[i] && dist[i] < min_dist) {
                min_dist = dist[i];
                u = i;
            }
        }
        vis[u] = 1;
        for(j=0; j<n; j++) {
            if(!vis[j] && c[u][j] && dist[u] !=
              INT_MAX && dist[u] + c[u][j] < dist[j])
            {
                dist[j] = dist[u] + c[u][j];
            }
        }
```

```
        count++;
    }
    printf("Shortest distances from source %d:\n", src);
    for(i=0; i<n; i++) {
        if(dist[i]==INT_MAX) {
            printf("%d->%d :unreachable\n", src, i);
        } else {
            printf("%d->%d: %d\n", src, i, dist[i]);
        }
    }
}
int main() {
    int n, src, i, j;
    int c[MAX_VERTICES][MAX_VERTICES];
    printf("Enter the number of vertices:");
    scanf("%d", &n);
    printf("Enter the cost adjacency matrix:\n");
    for(i=0; i<n; i++) {
        for(j=0; j<n; j++) {
            scanf("%d", &c[i][j]);
        }
    }
    printf("Enter the source vertex:");
    scanf("%d", &src);
    dijkstras(c, n, src);
    return 0;
}
```

Output

```
Enter the number of vertices: 6
enter the cost adjacency matrix:
```



0 15 10 9999 45 9999
9999 0 15 9999 20 9999
20 9999 0 20 9999 9999
9999 10 9999 0 35 9999
9999 9999 9999 30 0 9999
9999 9999 9999 4 9999 0
Enter the source vertex: 5
Shortest distances from source: 5:
5 -> 0 : 49
5 -> 1 : 14
5 -> 2 : 29
5 -> 3 : 4
5 -> 4 : 34
5 -> 5 : 0

→ b) Kruskal

```
#include<stdio.h>
#include<stdlib.h>
#define MAX_VERTICES 100
struct Edge {
    int src, dest, weight;
};
struct subset {
    int parent;
    int rank;
};
int find(struct subset subsets[], int i);
void Union(struct subset subsets[], int x, int y);
void KruskalMST(struct Edge edges[], int n, int e);
int compare(const void* a, const void* b) {
    struct Edge* edge1 = (struct Edge*) a;
    struct Edge* edge2 = (struct Edge*) b;
```

```
    return edge1->weight - edge2->weight;
}
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 xroot = find(subsets, x);
    int yroot = find(subsets, y);
    if (subsets[xroot].rank < subsets[yroot].rank) {
        subsets[xroot].parent = yroot;
    } else if (subsets[xroot].rank > subsets[yroot].
                                 rank) {
        subsets[yroot].parent = xroot;
    } else {
        subsets[yroot].parent = xroot;
        subsets[xroot].rank++;
    }
}
void KruskalMST(struct Edge edges[], int n, int e) {
    struct Edge result[n];
    int i=0;
    int minCost=0;
    struct subset subsets[MAX_VERTICES];
    for(int v=0; v<n; v++) {
        subsets[v].parent=v;
        subsets[v].rank=0;
    }
    qsort(edges, e, sizeof(struct Edge), compare);
```

```
int edge_count=0;
while(edge_count<n-1&&i<e){
    struct Edge next_edge=edges[i++];
    int x=find(subsets,next_edge.src);
    int y=find(subsets,next_edge.dest);
    if(x!=y){
        result[edge_count++]=next_edge;
        Union(subsets,x,y);
        mincost+=next_edge.weight;
    }
}
printf("Edges in the Minimum spanning Tree:\n");
for(i=0;i<edge_count;i++){
    printf("%d-%d:%d\n",result[i].src,
        result[i].dest,result[i].weight);
}
printf("Minimum cost of MST:%d\n",mincost);
}
int main(){
    int n,i,j;
    struct Edge graph[MAX_VERTICES*MAX_VERTICES];
    int e=0;
    printf("Enter the number of vertices:");
    scanf("%d",&n);
    printf("Enter the cost adjacency matrix:\n");
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            int weight;
            scanf("%d",&weight);
            if(weight!=9999&&i!=j){
                graph[e].src=i;
                graph[e].dest=j;
```

```
        graph[e].weight = weight;
        e++;
    }
  }
}
kruskal MST(graph, n, e);
return 0;
}
```

Output:

Enter the number of vertices: 7

Enter the adjacency matrix:

0 28 9999 9999 9999 10 9999

28 0 16 9999 9999 9999 14

9999 16 0 12 9999 9999 9999

9999 9999 12 0 22 9999 18

9999 9999 9999 22 0 25 24

10 9999 9999 9999 25 9999 9999

9999 14 9999 18 24 9999 9999

Minimum cost of MST: 99