3/8/23 Kruskal's Algorithm

```
#include <stdio.h>
#include <stdbool.h>
#define MAX_Vertices 100

int parent [MAX_Vertices];
int rank [MAX_Vertices];
int find (int vertex) {
  if (parent [vertex] != vertex)
    parent [vertex] = find (parent [vertex]);
  return parent [vertex];

Void Union Sets (int x, int y) {
  int xroot = find (x);
  int yroot = find (y);

if (rank [xroot] < rank [yroot])
    parent [xroot] = yroot;
  else if (rank [xroot] > rank [yroot])
    parent [yroot] = xroot;
else {
  parent [yroot] = xroot;
  rank [xroot] ++;
}
}
Void KruskalMST (int numVertices, int graph []
                 [MAX_Vertices]) {
for (int i = 0; i < numVertices; i++) {
    parent [i] = i;
    rank [i] = 0;
}
```

```
int mstWeight = 0;
int mstEdges = 0;

while (mstEdges < numVertices - 1) {
    int minWeight = -INT_MAX;
    int minSrc = -1, minDst = -1;

for (int src = 0; src < numVertices; src++)
    {
    if (graph[src][dst] != 0 && find(src) !=
    find(dst) && graph[src][dst] < minWeight)
    {
    minWeight = graph[src][dst];
    minSrc = src;
    minDst = dst;
  }
 }
}
 if (minSrc != -1 && minDst != -1) {
  Printf ("%d -- %d weight: %d \n",
    minSrc, minDst, minWeight);
Union Sets (minSrc, minDst);
 mstWeight += minWeight;
 mstEdges++;
 } else {
    Break;
  }
}
Printf ("Min Spanning tree weight: %d\n
        , mstWeight);
}
```

```
int main() {
    int numVertices;
Printf("Enter the no of vertices :");
Scanf("%d", &numVertices);

int graph[MAX_Vertices][MAX_Vertices];

Printf("Enter the weighted Sq Matrix (0 for no
    edges) :\n");
for(int i=0; i<numVertices; i++) {
  for(int j=0; j<numVertices; j++) {
    Scanf("%d", &graph[i][j]);
  }
}
  Printf("Min Spanning tree :\n");
  KruskalMST(numVertices, graph);
  return 0;
}
```

output

Enter the no of Vertices = 5

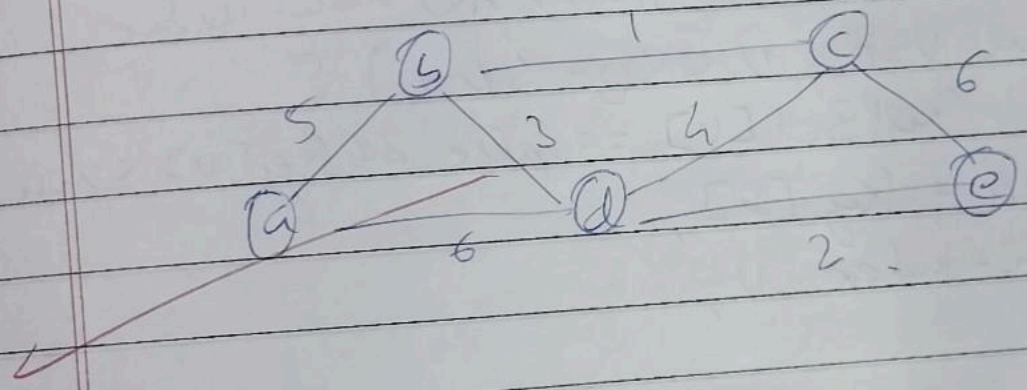
Enter the weighted Sq Matrix

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

Minimum Spanning Tree

0 - 1 weight - 2
1 - 2 weight - 3
1 - 4 weight - 5
0 - 3 weight - 6.

Min Spanning Tree weight : 16.



5/23 Prim's Algorithm

```
#include <stdio.h>
#include <stdbool.h>
#include <limits.h>
#include <stdlib.h>

#define MAX_SIZE 100
int findMinimumKey(int key[], bool mstSet
    [], int size) {
        int min = INT_MAX, min_index;
for (int v=0; v<size; v++) {
    if (mstSet[v] == false && key[v] < min) {
    min = key[v];
    min_index = v;
  }
}
return min_index;
void printMST(int parent[], int graph
        [MAX_SIZE][MAX_SIZE], int size) {
Printf("Edge \t Weight \n");
for(int i=1; i<size; i++) {
  Printf("%d - %d \t %d \n", parent[i], i,
graph[i][parent[i]]);

void primMST(int graph[MAX_SIZE][MAX_SIZE],
        int SIZE) {
int parent[MAX_SIZE];
int key[MAX_SIZE];
bool mstSet[MAX_SIZE];
```

```
for(int i =0 ; i < Size; i++){
    key [i] = INT - MAX;
    mstSet [i] = false;
}
key[0] = 0;
parent [0] = -1

for (int count =0; count < Size - 1; count ++){
    int u = findMinimum(key, mstSet, Size);
    mstSet[u] = true;

    for (int v=0; v< Size; v++){
        if (graph [u][v] && mstSet[v] == false &&
        graph [u][v] < key [v] ){
            parent [v] = u;
            key [v] = graph [u][v];
        }
    }
}
printMST( Parent, graph, Size);
}

int main (){
    int Size, Graph [MAX - SIZE ][MAX - SIZE]

Pf ("Enter the Size of the Sq Matrix (max %d)
    :", MAX-SIZE );
Sf ("%d", &SIZE);

Pf(" Enter the element of the Sq matrix
        (%d x %d) : \n", Size, Size);
```

```
for (int i=0; i<size; i++) {
    for (int j=0; j<size; j++){
      scanf ("%d", &graph [i][j] );
  }
}
PrimMST (graph, size);
  return 0;
}.
```

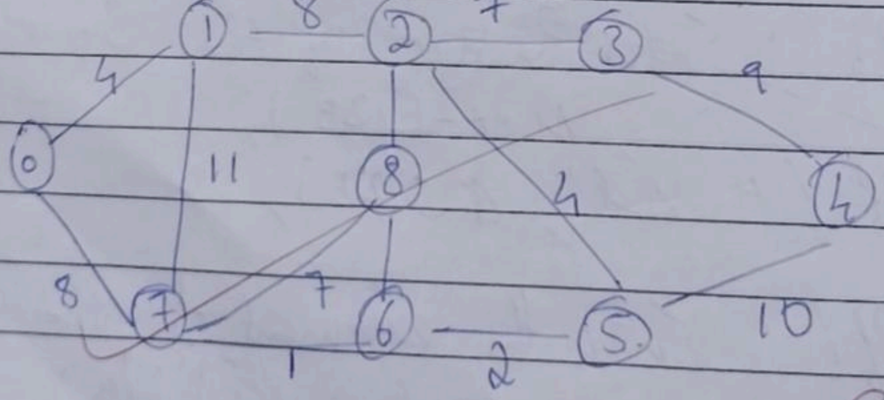
Output

Enter the size of the Sq Matrix - 5
Enter the element of the Sq Matrix

```
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
```

Edge	Weight
0 - 1	2
1 - 2	3
0 - 3	6
1 - 4	5

Min Spanning tree = 16.



03/08/2023