求最小支撑树的克鲁斯卡尔算法

```
void Graph_Matrix::kruskal(){
  struct node{ //边集：起点、终点、权值
        int head;
        int tail;
        int cost;
  }E[10001],TE[10001];
  int Make_set[1001],m;
  int rank[1001];//用于低秩合并算法
  int n=graphsize;
  int l=0;//边数
  for(int i=0;i<n;i++){    //把所有的边保存在E中
        for(int j=i;j<n;j++)
        {
              if(edge[i][j]!=0){
                    E[l].head=i;
                    E[l].tail=j;
                    E[l].cost=edge[i][j];
                    l=l+1;
              }
        }
  }
  int cmp(const void *a,const void *b);
  //下面是采用路径压缩的方法查找元素：
  int find_set(int x, int *Make_set);//查找
  //按秩合并
  void Union(int a,int b,int *Make_set,int *rand);//合并
  for(int i = 0; i <= n; i++)
  {
        Make_set[i] = i;  //每个节点初始化自成一个集合
        rank[i]=0;
  }
  int T=n;//初始n个连通分量
  m=l;
  qsort(E,m,sizeof(E[0]),cmp); //升序排序    快速排序
  int j=0;
  int count=0;
  while(T>1)//按边的权值大小加边
  {
        //找当前点到最小生成树的最短边
        int vex1 = E[j].head;
        int vex2 = E[j].tail;
        int cost = E[j].cost;
        if(find_set(vex1,Make_set) != find_set(vex2,Make_set))
        {

              TE[count].head=vex1;
              TE[count].tail=vex2;
              TE[count].cost=cost;

              count=count+1;
              Union(vex1,vex2,Make_set,rank);
              T=T-1;
        }
        j=j+1;
  }
  cout<<"最小支撑树："<<endl;
  for (int i=0;i<n-1;i++)
        cout<<"("<<TE[i].head<<", "<<TE[i].tail<<", "<<TE[i].cost<<")"<<endl;
}
int cmp(const void *a,const void *b)
{
  struct node{ //边集：起点、终点、权值
        int head;
        int tail;
```

```
        int cost;
  };
  return((struct node *)a)->cost -((struct node *)b)->cost ;

}
int find_set(int x,int *Make_set)
{
  int k, j, r;
  r = x;
  while(r != Make_set[r])         //查找跟节点
        r = Make_set[r];          //找到跟节点，用r记录下
  k = x;
  while(k != r)                   //非递归路径压缩操作
  {
        j = Make_set[k];              //用j暂存Make_set[k]的父节点
        Make_set[k] = r;              //Make_set[k]指向跟节点
        k = j;                        //k移到父节点
  }
  return r;               //返回根节点的值
}
void Union(int a,int b,int *Make_set,int *rank)
{
  int f1=find_set(a,Make_set);
  int f2=find_set(b,Make_set);
  if (rank[f1] <= rank[f2]) {
        Make_set[f1] = f2;
        if (rank[f1] == rank[f2]) {
              rank[f2] ++;
        }
  } else {
        Make_set[f2] = f1;
  }
}
```