/\*\*

\* kruskal.c 克鲁斯卡尔算法

\*

\* @auther Darbuly 2018-2019

\* @copyright 2018-2019 Darbuly

\*

\*/

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define MAX\_VERTEX\_NUM 20

typedef struct

{

int begin;

int end;

int weight;

}Edge[MAX\_VERTEX\_NUM];

typedef struct

{

Edge edge;

int vexnum,arcnum;

}Graph;

void CreateGraph(Graph\* G)

{

int i;

printf("Please enter vexnum and arcnum:\n");

scanf("%d%d",&G->vexnum,&G->arcnum);

getchar();

printf("Please enter the begin,end,weight by asc order:\n");

for(i=0;i<G->arcnum;i++)

{

scanf("%d %d %d",&G->edge[i].begin,&G->edge[i].end,&G->edge[i].weight);

getchar();

}

for(i=0;i<G->arcnum;i++)

{

printf("%d---->%d:%d\n",G->edge[i].begin,G->edge[i].end,G->edge[i].weight);

}

}

int Find(int \*parent,int f)

{

while(parent[f]>0)

{

f = parent[f];

}

return f;

}

int MiniSpanTree\_Kruskal(Graph G)

{

int i,n,m;

int parent[MAX\_VERTEX\_NUM];

for(i=0;i<G.vexnum;i++)

{

parent[i]=0;

}

for(i=0;i<G.arcnum;i++)

{

n=Find(parent,G.edge[i].begin);

m=Find(parent,G.edge[i].end);

if(n!=m)

{

parent[n]=m;

printf("(%d, %d) %d",G.edge[i].begin,G.edge[i].end,G.edge[i].weight);

}

}

}

int main()

{

Graph G;

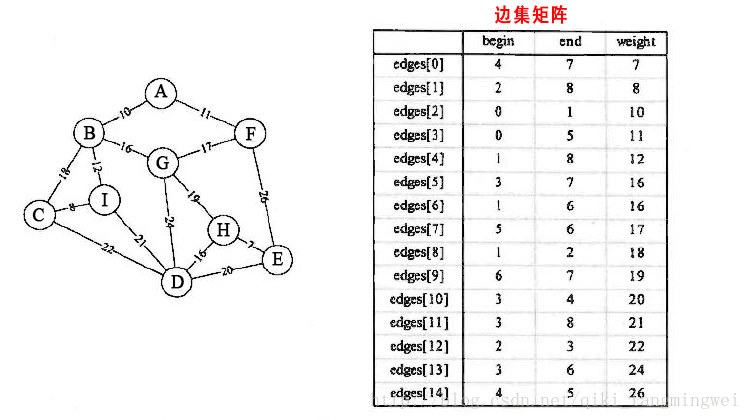
CreateGraph(&G);

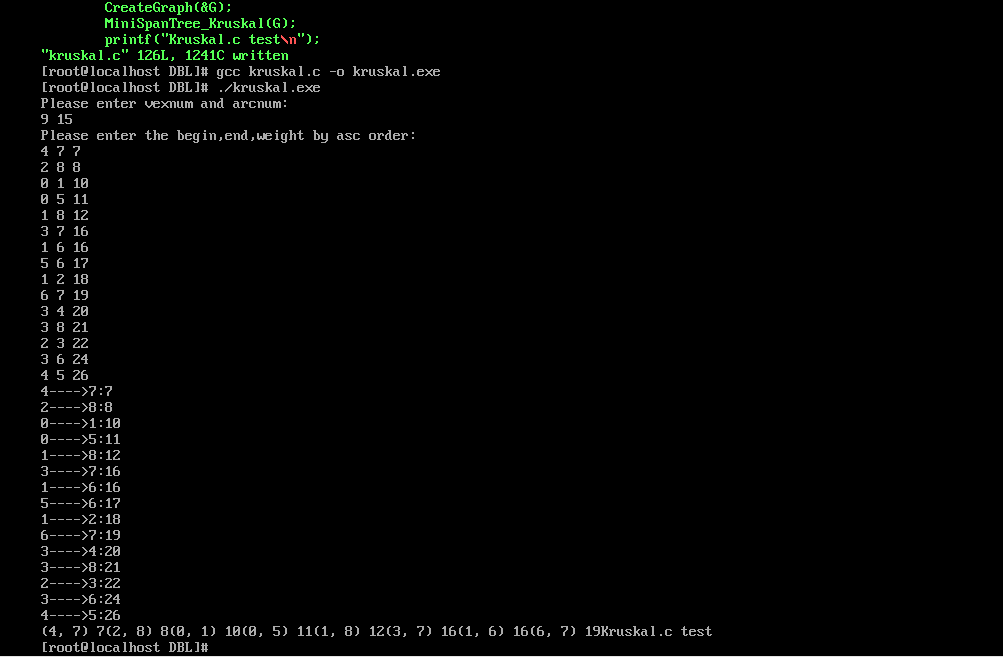
MiniSpanTree\_Kruskal(G);

printf("Kruskal.c test\n");

return 0;

}





算法分析报告：

1.克鲁斯卡尔算法重点需要解决的以下两个问题：

问题一 对图的所有边按照权值大小进行排序。

问题二 将边添加到最小生成树中时，怎么样判断是否形成了回路。

为什么find可以发现闭环现象

1.find基于parent数组，灵活的指出一颗生成树的支点，

1. Find返回的是哪个位置可以填空的问题，如果下一个可以存放的位置刚好和下下一个可以存放的位置相同，就会连环，或者说，find可以连接的问题，但是，如果刚好我要连的就是又是它本身，那就是头尾相连出现了