一:并查集的介绍

并查集是若干个不相交集合，能够实现较快的合并和判断元素所在集合的操作，应用很多。一般采取树形结构来存储并查集，并利用一个rank数组来存储集合的深度下界，在查找操作时进行路径压缩使后续的查找操作加速。这样优化实现的并查集，空间复杂度为O(N)，建立一个集合的时间复杂度为O(1)，N次合并M查找的时间复杂度为O(M Alpha(N))，这里Alpha是Ackerman函数的某个反函数，在很大的范围内（人类目前观测到的宇宙范围估算有10的80次方个原子，这小于前面所说的范围）这个函数的值可以看成是不大于4的，所以并查集的操作可以看作是线性的。

二：并查集的基本操作

1.初始化并查集

void MakeSet(int x)

{

set[x]=x;

rank[x]=0;

}

或者

void Init\_Set(int n)

{

for(int i=0;i<n;i++)

set[i]=-1;

}

视具体而定

2.查找某个元素属于哪个集合，即寻找树根，并路径压缩

int FindSet(int x)

{

if(set[x]!=x)

set[x]=FindSet(set[x]);

return set[x];

}

或者

int Find\_Set(int id)

{

if(set[id]<0) return id;

set[id]=Find\_Set(set[id]);

return set[id];

}

或者非递归，即循环

int find\_root(int x)

{

int r=x,i=x;

while(s[r]>=0)

r=s[r];

while(x!=r)

{

i=s[x];

s[x]=r;

x=i;

}

return r;

}

或者更简化

int getfather(int v)

{

return (father[v]==0)?v:(father[v]=getfather(father[v]));

}

结束条件判断视情况而定

3.判断两个元素是否属于同一个集合 FindSet(a)==FindSet(b)

4.合并两个元素所属的两个集合

方法一：比较树的深度(比较好)

void Link(int a,int b)

{

if(a==b) return;

if(rank[a]>rank[b])

set[b]=a;

else if(rank[a]<rank[b])

set[a]=b;

else

{

set[a]=b;

rank[b]++;

}

}

void Union(int a,int b)

{

Link(FindSet(a),FindSet(b));

}

方法二：比较集合元素的个数

void merge(int a,int b)

{

int k=find\_root(a-1),l=find\_root(b-1);

if(k==l) return;

if(s[l]<=s[k])

{

s[l]+=s[k];

s[k]=l;

}

else

{

s[k]+=s[l];

s[l]=k;

}

}

方法三：或者按照数组下标大小

void merge(int i,int j)

{

i=getfather(i);

j=getfather(j);

if(i==j) return;

else if (i<j)

father[j]=i;

else

father[i]=j;

}

三：并查集的优化思想

方法一：

将深度小的树合并到深度大的树

效果：任意顺序的合并操作以后，包含k个节点的树的最大高度不超过lgk

如果不采取这种优化，这棵树可能会退化为高度n

方法二：

路径压缩

思想：每次查找的时候，如果路径较长，则修改信息，以便下次查找的时候速度更快

步骤:

第一步，找到根结点

第二步，修改查找路径上的所有节点，将它们都指向根结点

四：并查集的应用(通过并查集实现Kruskal算法求图的最小生成树)

#define MAX 100

#include <iostream>

using namespace std;

struct edge

{

int cost;

int first;

int second;

};

int set[MAX];

edge my\_pair[MAX];

int rank[MAX];

int FindSet(int x)

{

if(set[x]!=x)

set[x]=FindSet(set[x]);

return set[x];

}

void MakeSet(int n)

{

for(int i=1;i<=n;i++){

set[i]=i;rank[i]=0;

}

}

void Link(int a,int b)

{

if(rank[a]>rank[b])

set[b]=a;

else if(rank[a]<rank[b])

set[a]=b;

else

{

set[a]=b;

rank[b]++;

}

}

void Union(int a,int b)

{

Link(FindSet(a),FindSet(b));

}

int my\_cmp(const void\* a,const void\* b)

{

return (\*(edge\*)a).cost-(\*(edge\*)b).cost;

}

int main()

{

int N,M,lowcost(0),i,count(1);

scanf("%d%d",&N,&M);

MakeSet(N);

for(i=0;i<M;i++)

{

scanf("%d%d%d",&my\_pair[i].first,&my\_pair[i].second,&my\_pair[i].cost);

}

qsort(my\_pair,M,sizeof(edge),my\_cmp);

for(i=0;i<M&&count<N;i++)

{

int first\_root=FindSet(my\_pair[i].first);

int second\_root=FindSet(my\_pair[i].second);

if(first\_root!=second\_root)

{

Union(first\_root,second\_root);

printf("%d----%d\n",my\_pair[i].first,my\_pair[i].second);

lowcost+=my\_pair[i].cost;

count++;

}

}

printf("%d\n",lowcost);

return 0;

}

五：并查集的OJ题及解题报告(待续和整理中.............)