2016级数据结构第十一次上机解题报告

A

这道题求的是一个图中有几个连通分量，我们可以对任意一个点做如下操作，如果该点已经被访问过了，则跳过它，否则将返回值加1，并从该点开始DFS，一直重复该操作，直到所有的节点都被访问就结束。

参考代码：

#include <iostream>

#include <cstdio>

#include <cstring>

using namespace std;

const int maxn = 100;

int m, k;

bool ways[maxn][maxn];

bool vis[maxn];

void dfs(int s) {

for (int i = 0; i < k; i++) {

if (!vis[i] && ways[s][i]) {

vis[i] = true;

dfs(i);

}

}

}

int main() {

while (~scanf("%d%d", &k, &m)) {

int u, v, res = 0;

memset(ways, 0, sizeof(ways));

memset(vis, 0, sizeof(vis));

for (int i = 0; i < m; i++) {

scanf("%d%d", &u, &v);

ways[u][v] = ways[v][u] = true;

}

for (int i = 0; i < k; i++) {

if (!vis[i]) {

res++;

vis[i] = true;

dfs(i);

}

}

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

}

}

B

简单的并查集/搜索/暴力都能过，没有过不了的方法。

参考代码：

#include <iostream>

#include <cstdio>

#include <algorithm>

#include <cstring>

#include <string>

using namespace std;

int a[30001], pre[30001];

int find(int x)

{

if (pre[x] == x)

return x;

else

return pre[x] = find(pre[x]);

}

void mix(int x, int y)

{

int fx = find(x), fy = find(y);

if (fx != fy)

pre[fy] = fx;

}

int main()

{

int n, m;

while (scanf("%d%d", &n, &m) != EOF)

{

int sum = 0;

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

pre[i] = i;

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

{

int k;

scanf("%d", &k);

if (k >= 1)

{

scanf("%d", &a[0]);

for (int j = 1; j < k; j++)

{

scanf("%d", &a[j]);

mix(a[0], a[j]);

}

}

}

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

if (find(i) == pre[0])

sum++;

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

}

return 0;

}

C

同上

参考代码：

#include <iostream>

using namespace std;

const int MAXN = 50005; /\*结点数目上线\*/

int pa[MAXN]; /\*p[x]表示x的父节点\*/

int rank1[MAXN]; /\*rank[x]是x的高度的一个上界\*/

int n, ans;

void make\_set(int x)

{/\*创建一个单元集\*/

pa[x] = x;

rank1[x] = 0;

}

int find\_set(int x)

{/\*带路径压缩的查找\*/

if (x != pa[x])

pa[x] = find\_set(pa[x]);

return pa[x];

}

/\*按秩合并x，y所在的集合\*/

void union\_set(int x, int y)

{

x = find\_set(x);

y = find\_set(y);

if (x == y)return;

ans--; //统计

if (rank1[x] > rank1[y])/\*让rank比较高的作为父结点\*/

{

pa[y] = x;

}

else

{

pa[x] = y;

if (rank1[x] == rank1[y])

rank1[y]++;

}

}

int main()

{

int m, i, j = 1, x, y;

while (scanf("%d%d", &n, &m) != EOF)

{

if (n == m && m == 0) break;

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

make\_set(i);

ans = n;

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

{

scanf("%d%d", &x, &y);

union\_set(x, y);

}

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

j++;

}

return 0;

}

D

此题为典型的最小生成树题目。

Kruskal算法是一种用来寻找最小生成树的算法，由Joseph Kruskal在1956年发表。用来解决同样问题的还有Prim算法和Boruvka算法等。三种算法都是贪婪算法的应用。和Boruvka算法不同的地方是，Kruskal算法在图中存在相同权值的边时也有效。

 算法简单描述

1).记Graph中有v个顶点，e个边

2).新建图Graphnew，Graphnew中拥有原图中相同的e个顶点，但没有边

3).将原图Graph中所有e个边按权值从小到大排序

4).循环：从权值最小的边开始遍历每条边 直至图Graph中所有的节点都在同一个连通分量中

                if 这条边连接的两个节点于图Graphnew中不在同一个连通分量中

                                         添加这条边到图Graphnew中

参考代码：

#include <iostream>

#include <algorithm>

using namespace std;

#define MaxN 10000

#define MaxM 600000

struct edge {

int x, y, w;

edge(int x = 0, int y = 0, int w = 0) : x(x), y(y), w(w) {}

} e[MaxM];

bool cmp(const edge &edge1, const edge &edge2) {

return (edge1.w < edge2.w);

}

int father[MaxN];

int getfather(int x) {

if (x == father[x]) return x;

else return father[x] = getfather(father[x]);

}

void unite(int x, int y) {

father[getfather(x)] = father[getfather(y)];

}

int kruscal(int N, int M) {

sort(e, e + M, cmp);

int cnt = N;

int ans = 0;

for (int i = 1; i <= N; ++i) father[i] = i;

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

int t1 = getfather(e[i].x);

int t2 = getfather(e[i].y);

if (t1 != t2) {

unite(t1, t2);

ans += e[i].w;

if (cnt == 1) break;

}

}

return ans;

}

int main() {

ios::sync\_with\_stdio(false);

cin.tie(0);

cout.tie(0);

int n, m, M = 0;

cin >> n >> m;

while (m--) {

int x, y, w;

cin >> x >> y >> w;

e[M++] = edge(x, y, w);

}

cout << kruscal(n, M) << "\n";

return 0;

}

E

按g排序后枚举g，然后按s生成最小生成树，计算选的边的G∗gmax+S∗smax，找到所有情况的最小值，每一次重新生成最小生成树时只需要考虑原最小生成树的边和新增的这一条边就可以了。

参考代码：

#include <cstdio>

#include <algorithm>

using namespace std;

typedef struct road

{

long long x;

long long y;

long long g;

long long s;

}road;

bool cmp(road a,road b)

{

return a.g<b.g;

}

road rd[50001];

long long father[201];

long long findfather(long long x)

{

if(father[x]==x)

return x;

else return father[x]=findfather(father[x]);

}

road MST[201];

int main()

{

//freopen("2.in","r",stdin);

//freopen("2.out","w",stdout);

long long N,M,G,S,e,fx,fy,tmp,ans;

while(scanf("%lld%lld",&N,&M)!=EOF)

{

scanf("%lld%lld",&G,&S);

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

scanf("%lld%lld%lld%lld",&rd[i].x,&rd[i].y,&rd[i].g,&rd[i].s);

sort(rd,rd+M,cmp);

e=0;

ans=-1;

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

{

tmp=0;

for(long long j=1;j<=N;j++)

father[j]=j;

MST[e++]=rd[i];

for(long long j=e-1;j>=1;j--)

if(MST[j].s<MST[j-1].s)

swap(MST[j],MST[j-1]);

for(long long j=0;j<e;j++)

{

fx=findfather(MST[j].x);

fy=findfather(MST[j].y);

if(fx!=fy)

{

father[fy]=fx;

MST[tmp++]=MST[j];

}

}

if(tmp==N-1&&(ans==-1||ans>G\*rd[i].g+S\*MST[N-2].s))

ans=G\*rd[i].g+S\*MST[N-2].s;

e=tmp;

}

printf("%lld\n",ans);

}

return 0;

}

F

先做前两个条件的判断，如果需要判断第三个条件，可以使用bitset，按值从小到大插入，记录对于每一行有哪一位比现在的小，再看现在这个数的i和j，对第i行和第j行取交，如果不为空，就不符合条件，如果所有检查完都没问题就符合条件，另外这道题可以用最小生成树做。

参考代码：

#include <cstdio>

#include <bitset>

#include <algorithm>

using namespace std;

int A[2501][2501];

bitset<2501> bs[2501];

typedef struct matrix

{

int val;

int x;

int y;

}matrix;

bool cmp(matrix a,matrix b)

{

return a.val<b.val;

}

matrix mat[2501\*2501];

int main()

{

int n,f,e;

while(scanf("%d",&n)!=EOF)

{

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

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

{

scanf("%d",&A[i][j]);

mat[i\*n+j].val=A[i][j];

mat[i\*n+j].x=i;

mat[i\*n+j].y=j;

}

f=0;

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

if(A[i][i]!=0)

{

f=1;

break;

}

if(f==1)

printf("NOT MAGIC\n");

else

{

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

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

if(A[i][j]!=A[j][i])

{

f=1;

break;

}

if(f==1)

printf("NOT MAGIC\n");

else

{

sort(mat,mat+n\*n,cmp);

e=0;

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

{

while(e<n\*n&&mat[e].val<mat[i].val)

{

bs[mat[e].x][mat[e].y]=1;

bs[mat[e].y][mat[e].x]=1;

e++;

}

if((bs[mat[i].x]&bs[mat[i].y]).any())

{

f=1;

break;

}

}

if(f==1)

printf("NOT MAGIC\n");

else printf("MAGIC\n");

}

}

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

bs[i].reset();

}

return 0;

}