1. Dijkstra
2. 欧拉路
3. 双连通分量

Dijkstra

struct node

{

int dis;

int pos;

friend bool operator <(node one,node two)

{

return one.dis > two.dis;

}

};

priority\_queue<node> q;

void dijkstra()

{

for(int i = 1; i <= n; ++i)dis[i] = inf;

dis[s] = 0;

q.push( ( node ){0, s} );

while( !q.empty() )

{

node tmp = q.top();

q.pop();

int x = tmp.pos, d = tmp.dis;

if( vis[x] )

continue;

vis[x] = 1;

for( int i = first[x]; i; i = e[i].next )

{

int y = e[i].to;

if( dis[y] > dis[x] + e[i].w )

{

dis[y] = dis[x] + e[i].w;

if( !vis[y] )

{

q.push( ( node ){dis[y], y} );

}

}

}

}

}

欧拉路

结论 1：无向图存在欧拉回路的条件：1）、图是连通图；2）、每个点的度为偶数。

结论 2：无向图存在欧拉路的条件：1）、图是连通图；2）、只有两个奇点（从一个奇点出发，经过每 条边一次且仅一次后，在另一个奇点结束）。

结论 3、有向图存在欧拉回路的条件：1）、有向图基连通； 2）、每个点的出入于入度相等 结论 4、有向图存在欧拉路的条件：1）、有向图基连；2）、仅有两个点的入度不等于出度，且出发点 的出度比入度多 1，结束点的入度比初度多 1。

bool vis[maxm];

//标记每条边是否访问过

void DFS(int i)

{

int sz=g[i].size();

for(int k=0;k<sz;k++)

{

int j=g[i][k].v,id=g[i][k].id;

if(vis[id]) continue; //忽略访问过的边

vis[id]=true; //标记该边访问过了

DFS(j);

}

ep.push\_back(i); //后序序列

}

void euler()

{

//判定是否是欧拉图（根据结论 1~4 的某一个）;

//如果是欧拉图，则寻找起点 s;

DFS(s); //DFS 寻找欧拉路

int sz=ep.size(); //输出欧拉路径

for(int k=sz-1;k>=0;k--) //倒序才能保证字典序最小

printf("%d ",ep[k]);

}

双连通分量

边双

int belong[maxn]={0},size[maxn]={0},ebc=0,maxcc=0;

int stk[maxn],top=0;

void DFS(int i,int fd)

{

vis[i]=1;

dfn[i]=low[i]=++dfs\_clock;

stk[++top]=i; //i 进栈

for(int p=first[i];p;p=e[p].next)

{

int j=e[p].to,id=e[p].id;

if(vis[j])

{

if(dfn[j]<dfn[i] && id!=fd)

low[i]=min(low[i],dfn[j]);

continue;

}

DFS (j,id);

low[i]=min(low[i],low[j]);

}

if(low[i]==dfn[i]) //i 与其父亲的边是割边

{

mark[fd]=1;

++ebc; //i 及其子孙还在栈中的点一定是一个边-双连通分量

while(1)

{

int x=stk[top--];

belong[x]=ebc, size[ebc]++;

if(x==i) break;

}

}

}

点双

int stk[maxn],top=0;

int cut[maxn],bcc=0,maxcc=0;

vector<int>b[maxn]; //b[i]记录第 i 个点-双连通分量包含的结点

void DFS(int i,int fd)

{

dfn[i]=low[i]=++dfs\_clock;

stk[++top]=i;

int chd=0;

for(int p=first[i];p;p=e[p].next)

{

int j=e[p].to,id=e[p].id;

if(dfn[j]) //j 已经访问过

{

if(dfn[j]<dfn[i] && id!=fd)

low[i]=min(low[i],dfn[j]); //返祖边

continue;

}

chd++;

DFS(j,id);

low[i]=min(low[i],low[j]);

if(low[j]>=dfn[i])

{

cut[i]=1; //i 是割顶

bcc\_cnt++; //i 及其部分后代组成新的“双连通分量”

bcc[bcc\_cnt].push\_back(i)

while(1)

{

int x=stk[top--];

bcc[bcc\_cnt].push\_back(x);

if(x==j)break;

}

if(bcc.size()>maxcc) maxcc=bcc[bcc\_cnt].size();

}

}

if(f==0 && chd==1)cut[i]=0;

}