## 前向星

void add(int x,int y,int w){

cnt++;

edge[cnt].to=y;edge[cnt].w=w;edge[cnt].next=head[x];

head[x]=cnt;

}//to该边终点 next与该边同起点的下个边的位置 head[]以i为起点的实际位置

## 拓扑排序

void topu(){

queue<int>q;

for(int i=1;i<=n;i++)if(!sum[i])q.push(i);//同时出现多个会不确定

while(!q.empty()){

int x=q.front();q.pop();

for(int i=head[x];i!=-1;i=edge[i].next){

f[edge[i].to]--;

if(!sum[edge[i].to])q.push(edge[i].to);//同时出现多个会不确定

}

}//排出的总个数不等于n说明有内环

}

## 二分图匹配

二分图中最小顶点覆盖等于最大匹配数

二分图中最小边覆盖=顶点数-最小顶点覆盖（最大匹配）

二分图中最大独立集+最小顶点覆盖（最大匹配）=顶点数

二分图匹配：

int dfs(int x){

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

if(f[x][i] && !vis[i]){

vis[i]=1;

if(!link[i] || dfs(link[i])){

link[i]=x;

return 1;

}

}

}

return 0;

}

int find(){

memset(link,0,sizeof(link));

cnt=0;

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

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

if(dfs(i))cnt++;

}

return cnt;

}