图论的补充：

输出最小字典序的欧拉路径：

```
// 欧拉路径的输出
#include <bits/stdc++.h>
using namespace std;
const int MAX=100010;
int n,m,u,v,del[MAX];
int du[MAX][2];//记录入度和出度
stack <int> st;
vector <int> G[MAX];
void dfs(int now)
{
    for(int i=del[now];i<G[now].size();i=del[now])
    {
        del[now]=i+1;
        dfs(G[now][i]);
    }
    st.push(now);
}
int main()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++) scanf("%d%d",&u,&v),G[u].push_back(v),du[u][1]++,du[v]
[0]++;
    for(int i=1;i<=n;i++) sort(G[i].begin(),G[i].end());
    int S=1,cnt[2]={0,0}; //记录
    bool flag=1; //flag=1表示,所有的节点的入度都等于出度,
    for(int i=1;i<=n;i++)
    {
        if(du[i][1]!=du[i][0])
        {
            flag=0;
            if(du[i][1]-du[i][0]==1/*出度比入度多1*/) cnt[1]++,S=i;
            else if(du[i][0]-du[i][1]==1/*入度比出度多1*/) cnt[0]++;
            else return puts("No"),0;
        }
    }
    if((!flag)&&!(cnt[0]==cnt[1]&&cnt[0]==1)) return !puts("No"),0;
    //不满足欧拉回路的判定条件，也不满足欧拉路径的判定条件，直接输出"No"
    dfs(S);
    while(!st.empty()) printf("%d ",st.top()),st.pop();
    return 0;
}
```

targan离线求lca:

```
using namespace std;
const int N = 1e4 + 10, M = 2 * N, INF = 0x3f3f3f3f;
int n, m, k;
int dist[N], st[N], p[N];
int h[N], e[M], ne[M], w[M], idx;
vector<PII> ques[N];
int res[M];
void add(int a, int b, int c)
{
    e[idx] = b, ne[idx] = h[a], w[idx] = c, h[a] = idx ++;
}
int find(int x)
{
    if(x != p[x]) p[x] = find(p[x]);
    return p[x];
}
void dfs(int u, int fa)
{
    for(int i = h[u]; ~i; i = ne[i])
    {
        int j = e[i];
        if(j == fa) continue;
        dist[j] = dist[u] + w[i];
        dfs(j, u);
    }
}
void tarjen(int u)
{
    st[u] = 1; // 标记为正在搜索的点
    for(int i = h[u]; ~i; i = ne[i])
    {
        int j = e[i];
        if(!st[j])
        {
            tarjen(j);
            p[j] = u;
        }
    }
    for(auto item : ques[u])
    {
        int y = item.ff, qid = item.ss;
        if(st[y] == 2)
        {
            int rot = find(y);
            res[qid] = dist[y] + dist[u] - 2 * dist[rot];
        }
    }
    st[u] = 2;
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
```

```
    cout.tie(0);
    memset(h, -1, sizeof h);
    cin >> n >> m;
    for(int i = 0; i < n - 1; i ++ )
    {
        int a, b, c;
        cin >> a >> b >> c;
        add(a, b, c), add(b, a, c);
    }
    for(int i = 0; i < m; i ++ )
    {
        int x, y;
        cin >> x >> y;
        if(x != y)
        {
            ques[x].pb({y, i});
            ques[y].pb({x, i});
        }
    }
    for(int i = 1; i <= n; i ++ )
        p[i] = i;
    dist[1] = 0;
    dfs(1, -1);
    tarjen(1);
    for(int i = 0; i < m; i ++ )
        cout << res[i] << endl;
    return 0;
}
```