原创 LCA（最近公共祖先）+RMQ（区间最值查询）

2019-08-12 11:02:26 _-Y-_-Y-_ 阅读数 82 更多 编辑

倍增法求解，预处理复杂度是 O(n log n) 每次询问的复杂度是 O(log n)

暴力求解LCA

```
/*
DFS建树 + 普通搜索
思路：
建树：找到根节点，
然后记录每个节点的父亲节点
和该节点的的深度，
搜索：从两个节点开始向上找，
一直找的两个节点重合为止
*/
#include <bits/stdc++.h>
using namespace std;
#define maxn 10010
vector<int> tree[maxn];
int fa[maxn];
int deep[maxn];
bool find_root[maxn];
int root,n,q;
void dfs(int x){
    for(int i=0;i<tree[x].size();i++){
        int y=tree[x][i];
        deep[y]=deep[x]+1;
        fa[y]=x;
        dfs(y);
    }
}
void init(){
    for(int i=1;i<=n;i++){
        if(!find_root[i]){
            root=i;
            break;
        }
    }
    deep[root]=1;
    fa[root]=root;
    dfs(root);
}
int lca(int x,int y){
    while(deep[x]>deep[y]) x=fa[x];
    while(deep[x]<deep[y]) y=fa[y];
    while(x!=y){
        x=fa[x];
        y=fa[y];
    }
    return x;
}
int main(){
    scanf("%d %d", &n, &q);
    memset(find_root,false,sizeof find_root);
    int u,v;
    for(int i=1;i<n;i++){
        scanf("%d %d", &u, &v);
        tree[u].push_back(v);
        find_root[v]=true;
    }
```

```
    init();
    while(q--){
        scanf("%d %d", &u, &v);
        printf("%d\n", lca(u,v));
    }
    return 0;
}

```

倍增优化

```
/*
DFS+倍增优化
与暴力求解对比:
不必一步一步的向上寻找父亲节点，可以跳着寻找
思路:
倍增思想：任何数字都可以转换为二进制，
那么我存每一个节点的二的幂次位祖先，
是不是想要那个节点的第多少祖先可以很快查询出来
*/
#include <bits/stdc++.h>
using namespace std;
#define maxn 10010
vector<int> tree[maxn];
int anc[maxn][25];
int fa[maxn];
int deep[maxn];
bool find_root[maxn];
int root,n,q;
void dfs(int x){
    anc[x][0]=fa[x];
    for(int i=1;i<=22;i++){
        anc[x][i]=anc[anc[x][i-1]][i-1];//体现倍增
    }
    for(int i=0;i<tree[x].size();i++){
            int y=tree[x][i];
            fa[y]=x;
            deep[y]=deep[x]+1;
            dfs(y);
    }
}
void init(){
    for(int i=1;i<=n;i++){
        if(!find_root[i]){
            root=i;
            break;
        }
    }
    deep[root]=1;
    fa[root]=root;
    dfs(root);
}
int lca(int x,int y) {
    if (deep[x]<deep[y]) swap(x,y);
    for (int i=22;i>=0;i--) {
        if (deep[y]<=deep[anc[x][i]]) {
            x=anc[x][i];
        }
    }
    if (x==y) return x;
    for (int i=22;i>=0;i--){
        if (anc[x][i]!=anc[y][i]) {
            x=anc[x][i];
            y=anc[y][i];
        }
    }
    return anc[x][0];//注意第二步IF语句的条件。
}
int main(){
    scanf("%d %d", &n, &q);
```

```
    memset(find_root,false,sizeof find_root);
    int u,v;
    for(int i=1;i<n;i++){
        scanf("%d %d", &u, &v);
        tree[u].push_back(v);
        find_root[v]=true;
    }
    init();
    while(q--){
        scanf("%d %d", &u, &v);
        printf("%d\n", lca(u,v));
    }
    return 0;
}

```

利用欧拉序转化为RMQ问题，用ST表求解RMQ问题

预处理复杂度 O(n + n log n) 每次询问的复杂度为 O(1)

ST表

```
/*
ST表求解 RMQ 区间最值查询
*/
#include <bits/stdc++.h>
using namespace std;
#define maxn 1000010
int num[maxn][25];
int a[maxn];
int n,q;
void ST(){
    int l=int(log((double)n)/log(2.0));
    for(int j=1;j<=l;j++){
        for(int i=1;i+(1<<(j-1))-1<=n;i++){
            num[i][j]=max(num[i][j-1], num[i+(1<<(j-1))][j-1]);
        }
    }
}
int rmq(int l,int r){
    int k=int(log((double)(r-l+1))/log(2.0));
    return max(num[l][k],num[r-(1<<k)+1][k]);
}
int main(){
    int x,y;
    scanf("%d %d", &n, &q);
    for(int i=1;i<=n;i++){
        scanf("%d", &a[i]);
        num[i][0]=a[i];
    }
    ST();
    while(q--){
        scanf("%d %d", &x, &y);
        printf("%d\n", rmq(x,y));
    }
    return 0;
}

```

欧拉序+ST表

```
/*
欧拉序+ST表
最近公共祖先是在该序列上两个点第一次出现的区间内深度最小的那个点
*/
#include <bits/stdc++.h>
```

```
using namespace std;
#define maxn 10010
vector<int> tree[maxn];
struct node{
    int deep;
    int m;
}a[maxn<<2],num[maxn<<2][25];//a存的是欧拉序，num是ST表
int first[maxn];//存每个节点第一次出现的cnt
int deep[maxn];//深度
bool find_root[maxn];
int root,n,q;
int cnt;
node calc(node a,node b){
    if(a.deep<b.deep) return a;
    return b;
}
void dfs(int x){
    first[x]=cnt;
    if(tree[x].size()==0){
        a[cnt].m=x;
        a[cnt++].deep=deep[x];
    }
    for(int i=0;i<tree[x].size();i++){
        int y=tree[x][i];
        deep[y]=deep[x]+1;
        a[cnt].m=x;
        a[cnt++].deep=deep[x];
        dfs(y);
    }
}
void ST(){
    int l=int(log((double)cnt)/log(2.0));
    for(int j=1;j<=l;j++){
        for(int i=1;i+(1<<(j-1))-1<=cnt;i++){
            num[i][j]=calc(num[i][j-1], num[i+(1<<(j-1))][j-1]);
        }
    }
}
void init(){
    for(int i=1;i<=n;i++){
        if(!find_root[i]){
            root=i;
            break;
        }
    }
    cnt=1;
    deep[root]=1;
    dfs(root);
    cnt--;
    for(int i=1;i<=cnt;i++){
        num[i][0]=a[i];
    }
    ST();
}
int rmq(int l,int r){
    l=first[l];
    r=first[r];
    if(l>r) swap(l,r);
    int k=int(log((double)(r-l+1))/log(2.0));
    return calc(num[l][k],num[r-(1<<k)+1][k]).m;
}
int main(){
    int x,y;
    scanf("%d %d", &n, &q);
    memset(find_root,false,sizeof find_root);
    for(int i=1;i<n;i++){
        scanf("%d %d", &x, &y);
        tree[x].push_back(y);
        find_root[y]=true;
    }
    init();
```

```
    while(q--){
        scanf("%d %d", &x, &y);
        printf("%d\n", rmq(x,y));
    }
    return 0;
}

```

有 0 个人打赏

文章最后发布于: 2019-08-12 11:02:26