[**poj 1330 LCA问题 （LCA问题转化为RMQ || tarjan算法）**](http://www.cnblogs.com/Missa/archive/2012/10/01/2709889.html)

LCA问题可以与RMQ问题互相转化，长郡中学 郭华阳的《RMQ&LCA问题》讲的很好。

# [LCA的三种解法](https://www.cnblogs.com/Mrleon/p/8512081.html):https://www.cnblogs.com/Mrleon/p/8512081.html

这个博客也讲的很好：<http://dongxicheng.org/structure/lca-rmq/>

RMQ问题：<http://www.cnblogs.com/Missa/archive/2012/10/01/2709686.html>

tarjan算法模版：<http://www.cnblogs.com/Missa/archive/2012/10/01/2709749.html>

RMQ+dfs：

[复制代码](javascript:void(0);)

1 #include <iostream>

2 #include <cstdio>

3 #include <cstring>

4 #include <cmath>

5

6 using namespace std;

7

8 #define MAXN 10005

9 #define MAXM 105

10 #define inf 0x7ffffff

11 int n;

12 struct Edge

13 {

14 int v,next;

15 }edge[MAXN];

16 int head[MAXN];

17 int e;

18

19 void clear()//初始化

20 {

21 memset(head,-1,sizeof(head));

22 e=0;

23 }

24 void addEdge(int u,int v)//加边

25 {

26 edge[e].v=v;

27 edge[e].next=head[u];head[u]=e++;

28 }

29 int first[MAXN];//结点在搜索顺序数组中最先出现的位置（下标）

30 int occur[MAXN<<1];//结点在出现的顺序数组重复的也要记录

31 int depth[MAXN<<1];//结点在搜索树中的深度，与occur相对应

32 int dp\_min[MAXN<<1][20];//dp\_min[i][j] 表示从第i个位置开始的2^j个元素中的最小值的下标

33 int m=0;//不断记录出现的下标

34

35 void dfs(int u,int deep)

36 {

37 occur[++m]=u;//进入该点时进行记录

38 depth[m]=deep;

39 if(!first[u])

40 first[u]=m;

41 for(int i=head[u];i+1;i=edge[i].next)

42 {

43 dfs(edge[i].v,deep+1);

44 occur[++m]=u;//访问子树返回也要标记

45 depth[m]=deep;

46 }

47 }

48 void init()

49 {

50 clear();

51 m=0;

52 memset(first,0,sizeof(first));

53 bool in[MAXN];//记录结点有无入度

54 memset(in,false,sizeof(in));

55 int u=0,v=0;

56 scanf("%d",&n);

57 for(int i=1;i<n;i++)//注意此题只有n-1条边

58 {

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

60 addEdge(u,v);//u->v单向

61 in[v]=true;

62 }

63 for(int i=1;i<=n;i++)//从根开始dfs

64 {

65 if(!in[i])

66 {

67 dfs(i,0);

68 break;

69 }

70 }

71 }

72

73 void RMQ\_init(int num)

74 {

75 for(int i=1;i<=num;i++)

76 dp\_min[i][0]=i;//注意dp\_min存的不是最小值，而是最小值的下标

77 for(int j=1;j<20;j++)

78 for(int i=1;i<=num;i++)

79 {

80 if(i+(1<<j)-1 <= num)

81 {

82 dp\_min[i][j] = depth[dp\_min[i][j-1]] < depth[dp\_min[i+(1<<(j-1))][j-1]] ? dp\_min[i][j-1] : dp\_min[i+(1<<(j-1))][j-1];

83 }

84 }

85 }

86

87 int RMQ\_min(int a,int b)

88 {

89 int l=first[a],r=first[b];//得到区间左右端点

90 if(l>r)

91 {

92 int t=l;

93 l=r;

94 r=t;

95 }

96 int k=(int)(log(double(r-l+1))/log(2.0));

97 int min\_id=depth[dp\_min[l][k]]<depth[dp\_min[r-(1<<k)+1][k]]?dp\_min[l][k]:dp\_min[r-(1<<k)+1][k];//最小值下标

98 return occur[min\_id];//取得当前下标表示的结点

99 }

100

101 int main()

102 {

103 int t;

104 int a,b;

105 scanf("%d",&t);

106 while(t--)

107 {

108 init();

109 RMQ\_init(m);

110 scanf("%d%d",&a,&b);

111 printf("%d\n",RMQ\_min(a,b));

112 }

113 return 0;

114 }

[复制代码](javascript:void(0);)

tarjan算法：

[复制代码](javascript:void(0);)

1 //O(n+Q)

2

3 #include <iostream>

4 #include <cstdio>

5 #include <cstring>

6 #include <vector>

7

8 using namespace std;

9

10 #define MAXN 10001

11

12 int n,fa[MAXN];

13 int rank[MAXN];

14 int indegree[MAXN];

15 int vis[MAXN];

16 vector<int> hash[MAXN],Qes[MAXN];

17 int ances[MAXN];//祖先

18

19

20 void init(int n)

21 {

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

23 {

24 fa[i]=i;

25 rank[i]=0;

26 indegree[i]=0;

27 vis[i]=0;

28 ances[i]=0;

29 hash[i].clear();

30 Qes[i].clear();

31 }

32 }

33

34 int find(int x)

35 {

36 if(x != fa[x])

37 fa[x]=find(fa[x]);

38 return fa[x];

39 }

40

41 void unio(int x,int y)

42 {

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

44 if(fx==fy) return ;

45 if(rank[fy]<rank[fx])

46 fa[fy]=fx;

47 else

48 {

49 fa[fx]=fy;

50 if(rank[fx]==rank[fy])

51 rank[fy]++;

52 }

53 }

54

55 void Tarjan(int u)

56 {

57 ances[u]=u;

58 int i,size = hash[u].size();

59 for(i=0;i<size;i++)

60 {

61 Tarjan(hash[u][i]);//递归处理儿子

62 unio(u,hash[u][i]);//将儿子父亲合并，合并时会将儿子的父亲改为u

63 ances[find(u)]=u;//此时find(u)仍为u,即

64 }

65 vis[u]=1;

66

67 //查询

68 size = Qes[u].size();

69 for(i=0;i<size;i++)

70 {

71 if(vis[Qes[u][i]]==1)//即查询的另一个结点开始已经访问过，当前的u在此回合访问。

72 {

73 printf("%d\n",ances[find(Qes[u][i])]);//由于递归，此时还是在u

74 return;

75 }

76 }

77 }

78

79 int main()

80 {

81 int t;

82 int i,j;

83 scanf("%d",&t);

84 while(t--)

85 {

86 scanf("%d",&n);

87 init(n);

88 int s,d;

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

90 {

91 scanf("%d%d",&s,&d);

92 hash[s].push\_back(d);

93 indegree[d]++;

94 }

95 scanf("%d%d",&s,&d);

96 Qes[s].push\_back(d);

97 Qes[d].push\_back(s);

98 for(j=1;j<=n;j++)

99 {

100 if(indegree[j]==0)

101 {

102 Tarjan(j);

103 break;

104 }

105 }

106 }

107 return 0;

108 }