

树上莫队算法

2014-11-24 23:34

1266人阅读

评论(0)

收藏

举报

分类： 学习总结 (3)

版权声明：本文为博主原创文章，未经博主允许不得转载。

继续回来写博客.....记录点有意思的题目什么的。

貌似写过这个的没多少人.....所以我也记录一点。

首先序列上的莫队大家都应该很熟悉了.....

那么树上的莫队要怎么搞呢？

先来看个题目.....SPOJ COT2：求树上两点间路径上有多少个不同的点权。

序列上的莫队是把询问按照左端点分块了.....可是树上没有左端点，怎么办呢？我们把树分块。

按照DFS时间戳顺序，将树分成 $O(\sqrt{n})$ 个大小为 $O(\sqrt{n})$ 的块，那么树上的莫队询问排序的第一关键字就是第一个节点所在的块了！

这样分块以后，任意两个块之间的距离也是 $O(\sqrt{n})$ 级别的，所以时间复杂度是有保证的。

第二个关键字自然就是节点的DFS时间戳了！

但是，还有一个问题。树上的区间要怎么转移呢？要怎么从一个区间变到另一个区间呢？

这就有些难了，因为树上有LCA，貌似不好处理。

Orz了wyfcyx后，找到了vfk的博客看了一下。

“

用 $S(v, u)$ 代表 v 到 u 的路径上的结点的集合。

用 $root$ 来代表根结点，用 $lca(v, u)$ 来代表 v 、 u 的最近公共祖先。

那么

$$S(v, u) = S(root, v) \text{ xor } S(root, u) \text{ xor } lca(v, u)$$

其中 xor 是集合的对称差。

简单来说就是节点出现两次消掉。

lca 很讨厌，于是再定义

$$T(v, u) = S(root, v) \text{ xor } S(root, u)$$

观察将 $curV$ 移动到 $targetV$ 前后 $T(curV, curU)$ 变化:

$$T(curV, curU) = S(root, curV) \text{ xor } S(root, curU)$$

$$T(targetV, curU) = S(root, targetV) \text{ xor } S(root, curU)$$

取对称差:

$$T(curV, curU) \text{ xor } T(targetV, curU) = (S(root, curV) \text{ xor } S(root, curU)) \text{ xor } (S(root, targetV) \text{ xor } S(root, curU))$$

由于对称差的交换律、结合律:

$$T(curV, curU) \text{ xor } T(targetV, curU) = S(root, curV) \text{ xor } S(root, targetV)$$

两边同时xor $T(curV, curU)$:

$$T(targetV, curU) = T(curV, curU) \text{ xor } S(root, curV) \text{ xor } S(root, targetV)$$

发现最后两项很爽.....哇哈哈

$$T(targetV, curU) = T(curV, curU) \text{ xor } T(curV, targetV)$$

(有公式恐惧症的不要走啊 T_T)

也就是说, 更新的时候, xor $T(curV, targetV)$ 就行了。

即, 对 $curV$ 到 $targetV$ 路径(除开 $lca(curV, targetV)$)上的结点, 将它们的存在性取反即可。

”

——vfk博客

这样就很好处理了, 只要把LCA扔出去, 考虑剩下的部分, 转移一下就可以了。查答案的时候再把LCA那个点反过来, 就能统计出答案了。

这样, 类比序列上的莫队, 我们对树上的询问也可以分块了, 时间复杂度同样是 $O(n\sqrt{n})$ 。

BZOJ 3757貌似是同一个题, 就贴这个代码吧.....COT2没写.....SPOJ上怕T.....

[cpp]

```
01. #include<iostream>
02. #include<cstdio>
03. #include<algorithm>
04. #include<string>
05. #include<cstring>
06. #include<queue>
07. #include<vector>
08. #include<cmath>
09. using namespace std;
10. inline int getint()
11. {
12.     char c=getchar();
13.     int con=0;
14.     while(c<'0' || c>'9') c=getchar();
15.     while(c>='0' && c<='9') con=con*10+c-'0', c=getchar();
16.     return con;
17. }
18. const int MAXN=100010;
19. int n,m,K,lca,u,v,c[MAXN],dfn[MAXN],belongn[MAXN];
20. int tot,root,dfs_clock,remain;
21. int head[MAXN],to[MAXN],next[MAXN],cnt;
22. int anc[MAXN][21],dep[MAXN],Log[MAXN];
23. int Stack[MAXN],top;
24. int p[MAXN],ans,con[MAXN];
```

载:

```

25. bool used[MAXN];
26. struct Query
27. {
28.     int u,v,a,b,sub;
29.     friend bool operator<(const Query &i,const Query &j)
30.     {
31.         if(belongn[i.u]==belongn[j.u]) return dfn[i.v]<dfn[j.v];
32.         else return belongn[i.u]<belongn[j.u];
33.     }
34. }Q[MAXN];
35. inline void adde(int f,int t)
36. {
37.     cnt++,to[cnt]=t,next[cnt]=head[f],head[f]=cnt;
38.     cnt++,to[cnt]=f,next[cnt]=head[t],head[t]=cnt;
39. }
40. int DFS(int x)
41. {
42.     int size=0;
43.     dfn[x]=++dfs_clock;
44.     for(int i=head[x];i;i=next[i])
45.         if(to[i]!=anc[x][0])
46.         {
47.             dep[to[i]]=dep[x]+1,anc[to[i]][0]=x;
48.             size+=DFS(to[i]);
49.             if(size>=K)
50.             {
51.                 tot++;
52.                 for(int i=1;i<=size;i++)
53.                     belongn[Stack[top--]]=tot;
54.                 size=0;
55.             }
56.         }
57.     Stack[++top]=x;
58.     return size+1;
59. }
60. int LCA(int p,int q)
61. {
62.     if(dep[p]<dep[q]) swap(p,q);
63.     int d=dep[p]-dep[q];
64.     for(int i=Log[d];i>=0;i--)
65.         if(d&(1<<i)) p=anc[p][i];
66.     for(int i=Log[n];i>=0;i--)
67.         if(anc[p][i]!=anc[q][i]) p=anc[p][i],q=anc[q][i];
68.     if(p!=q) return anc[p][0];
69.     else return p;
70. }
71. void work(int u,int v,int lca)
72. {
73.     while(u!=lca)
74.     {
75.         if(!used[u]) {p[c[u]]++,used[u]=true;if(p[c[u]]==1) ans++;}
76.         else {p[c[u]]--,used[u]=false;if(p[c[u]]==0) ans--;}
77.         u=anc[u][0];
78.     }
79.     while(v!=lca)
80.     {

```

```

81.         if(!used[v]) {p[c[v]]++,used[v]=true;if(p[c[v]]==1) ans++;}
82.         else {p[c[v]]--,used[v]=false;if(p[c[v]]==0) ans--;}
83.         v=anc[v][0];
84.     }
85. }
86. int main()
87. {
88.     //freopen("apple.in","r",stdin);
89.     //freopen("apple.out","w",stdout);
90.     n=getint(),m=getint();
91.     K=(int)sqrt(n);
92.     for(int i=1;i<=n;i++) c[i]=getint();
93.     for(int i=1;i<=n;i++)
94.     {
95.         u=getint(),v=getint();
96.         if(u==0) root=v;
97.         else if(v==0) root=u;
98.         else adde(u,v);
99.     }
100.    for(int i=1;i<=m;i++)
101.    {
102.        Q[i].u=getint(),Q[i].v=getint();
103.        Q[i].a=getint(),Q[i].b=getint();
104.        Q[i].sub=i;
105.    }
106.    remain=DFS(root);
107.    for(int i=1;i<=remain;i++) belongn[Stack[top--]]=tot;
108.    sort(Q+1,Q+m+1);
109.    Log[0]=-1;
110.    for(int i=1;i<=n;i++) Log[i]=Log[i>>1]+1;
111.    for(int i=1;i<=Log[n];i++)
112.        for(int j=1;j<=n;j++)
113.            anc[j][i]=anc[anc[j][i-1]][i-1];
114.    work(Q[1].u,Q[1].v,lca=LCA(Q[1].u,Q[1].v));
115.    if(!used[lca]) {p[c[lca]]++,used[lca]=true;if(p[c[lca]]==1) ans++;}
116.    else {p[c[lca]]--,used[lca]=false;if(p[c[lca]]==0) ans--;}
117.    con[Q[1].sub]=ans;
118.    if(p[Q[1].a]!=0&&p[Q[1].b]!=0) con[Q[1].sub]--;
119.    if(!used[lca]) {p[c[lca]]++,used[lca]=true;if(p[c[lca]]==1) ans++;}
120.    else {p[c[lca]]--,used[lca]=false;if(p[c[lca]]==0) ans--;}
121.    for(int i=2;i<=m;i++)
122.    {
123.        work(Q[i-1].u,Q[i].u,LCA(Q[i-1].u,Q[i].u));
124.        work(Q[i-1].v,Q[i].v,LCA(Q[i-1].v,Q[i].v));
125.        lca=LCA(Q[i].u,Q[i].v);
126.        if(!used[lca]) {p[c[lca]]++,used[lca]=true;if(p[c[lca]]==1) ans++;}
127.        else {p[c[lca]]--,used[lca]=false;if(p[c[lca]]==0) ans--;}
128.        con[Q[i].sub]=ans;
129.        if(p[Q[i].a]!=0&&p[Q[i].b]!=0&&Q[i].a!=Q[i].b) con[Q[i].sub]--;
130.        if(!used[lca]) {p[c[lca]]++,used[lca]=true;if(p[c[lca]]==1) ans++;}
131.        else {p[c[lca]]--,used[lca]=false;if(p[c[lca]]==0) ans--;}
132.    }
133.    for(int i=1;i<=m;i++) printf("%d\n",con[i]);
134.    return 0;
135. }

```

感觉好神啊.....vfk的那个方法确实好用，这个算法也好有趣。

(未完待续.....)

WC2013 糖果公园 待填坑

[cpp]

```
01. #include<iostream>
02. #include<cstdio>
03. #include<algorithm>
04. #include<string>
05. #include<cstring>
06. #include<queue>
07. #include<vector>
08. #include<cmath>
09. using namespace std;
10. typedef long long LL;
11. const int MAXN=300010;
12. int n,m,u,v,Q,K,Type,C[MAXN],Ctmp[MAXN];
13. LL V[MAXN],W[MAXN],ans,con[MAXN];
14. int head[MAXN],to[MAXN],next[MAXN],cnt;
15. int belongn[MAXN],tot,remain;
16. int anc[MAXN][20],dep[MAXN],Log[MAXN];
17. int Stack[MAXN],top;
18. int Qtot,Ctot,Query_clock=1;
19. int p[MAXN],lca;
20. bool used[MAXN];
21. struct QQ
22. {
23.     int x,y,t,id;
24.     friend bool operator<(const QQ &i,const QQ &j)
25.     {
26.         if(belongn[i.x]<belongn[j.x]) return true;
27.         else if(belongn[i.x]==belongn[j.x]&&belongn[i.y]<belongn[j.y]) return true;
28.         else if(belongn[i.x]==belongn[j.x]&&belongn[i.y]==belongn[j.y]&&i.t<j.t) return true;
29.         return false;
30.     }
31. }Query[MAXN];
32. struct CC{int pos,x,y;}Change[MAXN];
33. inline void adde(int f,int t)
34. {
35.     cnt++,to[cnt]=t,next[cnt]=head[f],head[f]=cnt;
36.     cnt++,to[cnt]=f,next[cnt]=head[t],head[t]=cnt;
37. }
38. int DFS(int x)
39. {
40.     int size=0;
41.     for(int i=head[x];i;i=next[i])
42.         if(to[i]!=anc[x][0])
43.         {
44.             anc[to[i]][0]=x;
45.             dep[to[i]]=dep[x]+1;
46.             size+=DFS(to[i]);
47.             if(size>=K)
48.             {
```

载:

```

49.         tot++;
50.         for(int i=1;i<=size;i++) belongn[Stack[top--]]=tot;
51.         size=0;
52.     }
53. }
54. Stack[++top]=x;
55. return size+1;
56. }
57. int LCA(int p,int q)
58. {
59.     if(dep[p]<dep[q]) swap(p,q);
60.     int d=dep[p]-dep[q];
61.     for(int i=Log[d];i>=0;i--)
62.         if(d&(1<<i)) p=anc[p][i];
63.     for(int i=Log[n];i>=0;i--)
64.         if(anc[p][i]!=anc[q][i]) p=anc[p][i],q=anc[q][i];
65.     if(p!=q) return anc[p][0];
66.     return p;
67. }
68. void Reverse(int x)
69. {
70.     if(!used[x]) p[C[x]]++,ans+=V[C[x]]*W[p[C[x]]];
71.     else ans-=V[C[x]]*W[p[C[x]]],p[C[x]]--;
72.     used[x]^=1;
73. }
74. void ChangeVer(int x,int d)
75. {
76.     int pos=Change[x].pos;
77.     if(d==1)
78.     {
79.         if(used[pos]) Reverse(pos),C[pos]=Change[x].y,Reverse(pos);
80.         else C[pos]=Change[x].y;
81.     }
82.     else
83.     {
84.         if(used[pos]) Reverse(pos),C[pos]=Change[x].x,Reverse(pos);
85.         else C[pos]=Change[x].x;
86.     }
87. }
88. void work(int x,int y,int lca)
89. {
90.     while(x!=lca) Reverse(x),x=anc[x][0];
91.     while(y!=lca) Reverse(y),y=anc[y][0];
92. }
93. int main()
94. {
95.     scanf("%d%d%d",&n,&m,&Q);
96.     K=(int)pow((double)n,2.0/3);
97.     for(int i=1;i<=m;i++) scanf("%lld",&V[i]);
98.     for(int i=1;i<=n;i++) scanf("%lld",&W[i]);
99.     for(int i=1;i<=n;i++) scanf("%d",&u,&v),adde(u,v);
100.    for(int i=1;i<=n;i++) scanf("%d",&C[i]),Ctmp[i]=C[i];
101.    for(int i=1;i<=Q;i++)
102.    {
103.        scanf("%d%d%d",&Type,&u,&v);
104.        if(Type==0)

```

```
105.     {
106.         Ctot++,Query_clock++;
107.         Change[Ctot].pos=u,Change[Ctot].x=Ctmp[u],Change[Ctot].y=v;
108.         Ctmp[u]=v;
109.     }
110.     else
111.     {
112.         Qtot++;
113.         Query[Qtot].x=u,Query[Qtot].y=v,Query[Qtot].t=Query_clock;
114.         Query[Qtot].id=Qtot;
115.     }
116. }
117. remain=DFS(1);
118. for(int i=1;i<=remain;i++) belongn[Stack[top--]]=tot;
119. sort(Query+1,Query+Qtot+1);
120. Log[0]=-1;
121. for(int i=1;i<=n;i++) Log[i]=Log[i>>1]+1;
122. for(int i=1;i<=Log[n];i++)
123.     for(int j=1;j<=n;j++)
124.         anc[j][i]=anc[anc[j][i-1]][i-1];
125. work(Query[1].x,Query[1].y,lca=LCA(Query[1].x,Query[1].y));
126. for(int i=1;i<Query[1].t;i++) ChangeVer(i,1);
127. Reverse(lca),con[Query[1].id]=ans,Reverse(lca);
128. for(int i=2;i<=Qtot;i++)
129. {
130.     work(Query[i-1].x,Query[i].x,LCA(Query[i-1].x,Query[i].x));
131.     work(Query[i-1].y,Query[i].y,LCA(Query[i-1].y,Query[i].y));
132.     for(int j=Query[i-1].t;j<Query[i].t;j++) ChangeVer(j,1);
133.     for(int j=Query[i-1].t-1;j>=Query[i].t;j--) ChangeVer(j,-1);
134.     lca=LCA(Query[i].x,Query[i].y);
135.     Reverse(lca),con[Query[i].id]=ans,Reverse(lca);
136. }
137. for(int i=1;i<=Qtot;i++) printf("%lld\n",con[i]);
138. return 0;
139. }
```