模板题 题解

Posted on 2019-04-13

Description:

给出一棵以 1 号点为根的树，每个点上有一个字符，每个点所代表的字符串 $s[i]$ 为从这个点到根的字符连起来形成的字符串，求：

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n}\operatorname{lcs}(i,j)\times\operatorname{lcp}(i,j)$$

其中：lcs 表示两个字符串的最长公共后缀，lcp 表示两个字符串的最长公共前缀。

Solution:

对于 10% 的数据满足：$n\leqslant 200$

直接枚举点对暴力就行了，时间复杂度 $O(n^3)$ 。

对于 10% 的数据满足：$n\leqslant 5000$

显然两个串的 lcp 就是 lca 的深度，那么我们把所有串拿出来反过来建出 $trie$ ，还是暴力枚举点对用 ST 表 $O(1)$ 求 LCA 即可。

时间复杂度 $O(n^2)$ 。

对于 25% 的数据满足：$n\leqslant 300000$ 并且给出的树是一条链并且 1 号点是链的一个端点。

这个部分分就相当于是一个序列上的情况，那么不难想到我们对于这个序列反过来建后缀数组，那么我们考虑枚举每一个位置作为靠前的那个位置计算答案。

先建出 $height$ 数组的笛卡尔树，因为字符是随机的因此 $height$ 数组也可以看成是随机的，那么有一个结论是笛卡尔树的期望深度是 $\log n$，那么我们从后往前枚举这个位置，刚开始笛卡尔树为空，每次把当前这个位置的下一个位置插进树里去，同时在树上维护子树大小，然后对于当前这个询问，从对应的点不停向上跳，在跳的同时统计和他相对的那棵子树的答案即可。

时间复杂度期望 $O(n\log n)$。

```cpp
#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstdio>
#include<cmath>
#include<cctype>
#include<cstring>
using namespace std;
inline int rd() {
    register int res = 0,f = 1;register char c = getchar();
    while(!isdigit(c)){if(c == '-')f = -1;c = getchar();}
    while(isdigit(c))res = (res << 1) + (res << 3) + c - '0',c = getchar();
    return res * f;
}
int n;
#define MAXN 300010
int nxt[MAXN];
int s[MAXN];
int dfn[MAXN];
int sa[MAXN],rnk[MAXN],c[MAXN],c1[MAXN],c2[MAXN],h[MAXN];
void make_SA(int n,int m) {
    int p = 0;
    int *x = c1,*y = c2;
    for(int i = 1;i <= m;++i)c[i] = 0;
    for(int i = 1;i <= n;++i)++c[x[i] = s[i]];
    for(int i = 1;i <= m;++i)c[i] += c[i - 1];
    for(int i = n;i >= 1;--i)sa[c[x[i]]--] = i;
    for(int k = 1;k <= n;k = k << 1) {
        p = 0;
        for(int i = 1;i <= n;++i)y[i] = 0;
        for(int i = 1;i <= m;++i)c[i] = 0;
        for(int i = n - k + 1;i <= n;++i)y[++p] = i;
        for(int i = 1;i <= n;++i)if(sa[i] > k)y[++p] = sa[i] - k;
        for(int i = 1;i <= n;++i)++c[x[y[i]]];
        for(int i = 1;i <= m;++i)c[i] += c[i - 1];
        for(int i = n;i >= 1;--i)sa[c[x[y[i]]]--] = y[i];
        p = 1;
        swap(x,y);
        x[sa[1]] = 1;
        for(int i = 2;i <= n;++i) {
            x[sa[i]] = (y[sa[i]] == y[sa[i - 1]] && y[sa[i] + k] == y[sa[i - 1] + k] ?
p : ++p);
        }
        if(p >= n)break;
        m = p;
    }
    for(int i = 1;i <= n;++i)rnk[sa[i]] = i;
    int k = 0;
    for(int i = 1;i <= n;++i) {
        if(rnk[i] == 1)continue;
```

```
        if(k)--k;
        int j = sa[rnk[i] - 1];
        while(j + k <= n && i + k <= n && s[j + k] == s[i + k])++k;
        h[rnk[i]] = k;
    }
    return;
}
int root,lc[MAXN << 1],rc[MAXN << 1],fa[MAXN << 1],len[MAXN << 1];
int sum[MAXN << 1];
int tot;
int build(int &rt,int l,int r) {
    if(l == r) {
        rt = l;
        return rt;
    }
    int mn = 0x3f3f3f3f,pos = 0;
    for(int i = l + 1;i <= r;++i)if(h[i] < mn){pos = i;mn = h[i];}
    rt = ++tot;len[rt] = h[pos];
    fa[build(lc[rt],l,pos - 1)] = rt;
    fa[build(rc[rt],pos,r)] = rt;
    return rt;
}
#define MOD 998244353
int main() {
    scanf("%d",&n);
    for(int i = 2;i <= n;++i)nxt[rd()] = i;
    for(int i = 1,cur = 1;i <= n;++i) {
        dfn[cur] = i;
        cur = nxt[cur];
    }
    for(int i = 1;i <= n;++i)s[n - dfn[i] + 1] = rd() + 1;
    make_SA(n,256);
    tot = n;
    build(root,1,n);
    int ans = 0;
    for(int i = 2;i <= n;++i) {
        int cur = rnk[i - 1];
        while(cur != 0) {
            ++sum[cur];
            cur = fa[cur];
        }
        cur = rnk[i];
        while(fa[cur] != 0) {
            ans = (ans + 1ll * len[fa[cur]] * (n - i + 1) % MOD * (sum[fa[cur]] -
sum[cur]) % MOD) % MOD;
            cur = fa[cur];
        }
    }
    cout << ans << endl;
    return 0;
}
```

对于 25% 的数据满足：$n \leqslant 50000$

还是考虑随机的特殊性，另有一个结论是在随机情况下 lcs 的长度期望是 $\log_{\Sigma} n$ ，那么我们就可以考虑使用一个复杂度与 lcs 长度有关的算法。

既然 $\operatorname{lcp}(i,j) = dep[\mathrm{LCA}(i,j)]$ ，那么我们考虑枚举一个点并计算他作为 LCA 的代价，我们可以对于每个点哈希向上大约 $1 \sim \log_2 n$ 长度的字符串，出题人生成的数据长度大概不超过 50 ，然后就是要求 len 相等但是 $len+1$ 不相等的点对数，这个可以开一个桶计算，计算的过程可以用树上启发式合并($dsu\ on\ tree$)来优化。

时间复杂度 $O(n \log^2 n)$ 。

```
#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstdio>
#include<cmath>
#include<map>
#include<cctype>
#include<cstring>
using namespace std;
inline int rd() {
    register int res = 0,f = 1;register char c = getchar();
    while(!isdigit(c)){if(c == '-')f = -1;c = getchar();}
    while(isdigit(c))res = (res << 1) + (res << 3) + c - '0',c = getchar();
    return res * f;
}
int n;
#define MAXN 100010
struct edge {
    int to,nxt;
} e[MAXN];
int edgenum = 0;
int lin[MAXN] = {0};
void add(int a,int b) {
    e[++edgenum] = (edge){b,lin[a]};lin[a] = edgenum;
    return;
}
int fa[MAXN];
int dep[MAXN];
int s[MAXN];
#define MAX 45
#define MAXX 50
#define BASE1 19260817
#define MODN1 1000000007
#define BASE2 1001001
#define MODN2 998244353
typedef unsigned int uint;
int siz[MAXN],son[MAXN];
void dfs1(int k) {
    siz[k] = 1;
    for(int i = lin[k];i != 0;i = e[i].nxt) {
        dfs1(e[i].to);
        siz[k] += siz[e[i].to];
        if(son[k] == 0 || siz[e[i].to] > siz[son[k]])son[k] = e[i].to;
    }
    return;
}
#define pii pair<uint,uint>
#define fi first
#define se second
#define mp make_pair
```

```
long long kkk = 0;
pii hash[MAXN][MAXX];
typedef unsigned long long ull;
ull hsh[MAXN][MAXX];
struct table {
    #define MO 1001001
    int head[MO];
    ull st[MAXN];
    int val[MAXN],nxt[MAXN];
    int cntnum;
    int stack[MAXN],top;
    int& operator [] (ull x) {
        int modx = x % MO;
        for(int i = head[modx];i != 0;i = nxt[i]) {
            if(st[i] == x)return val[i];
        }
        stack[++top] = modx;
        ++cntnum;
        st[cntnum] = x;
        val[cntnum] = 0;
        nxt[cntnum] = head[modx];
        head[modx] = cntnum;
        return val[cntnum];
    }
    void clear() {
        while(top)head[stack[top--]] = 0;
        cntnum = 0;
        return;
    }
};
namespace T {
    table f[MAXX];
    table sum[MAXX];
    int ans = 0;
    void clear() {
        for(int i = 0;i < MAX;++i)f[i].clear();
        for(int i = 0;i < MAX;++i)sum[i].clear();
        ans = 0;
        return;
    }
    void query(int k) {
        for(int i = 0;i <= min(MAX - 1,dep[k]);++i) {
            ans += (sum[i][hsh[k][i]] - f[i][hsh[k][i] ^ hsh[k][i + 1]]) * i;
        }
        return;
    }
    void insert(int k) {
        for(int i = 0;i <= min(MAX - 1,dep[k]);++i) {
            ++sum[i][hsh[k][i]];++kkk;
            ++f[i][hsh[k][i] ^ hsh[k][i + 1]];
        }
```

```cpp
            return;
        }
}
void calc(int k) {
    T::query(k);
    for(int i = lin[k];i != 0;i = e[i].nxt)calc(e[i].to);
    return;
}
void add(int k) {
    T::insert(k);
    for(int i = lin[k];i != 0;i = e[i].nxt)add(e[i].to);
    return;
}
int ans = 0;
#define MOD 998244353
void dfs(int k,int ty) {
    if(son[k] != 0) {
        for(int i = lin[k];i != 0;i = e[i].nxt)if(e[i].to != son[k])dfs(e[i].to,0);
        dfs(son[k],1);
        T::ans = 0;
        for(int i = lin[k];i != 0;i = e[i].nxt) {
            if(e[i].to == son[k])continue;
            calc(e[i].to);
            add(e[i].to);
        }
    }
    T::query(k);
    T::insert(k);
    ans = (ans + 1ll * T::ans * dep[k] % MOD) % MOD;
    if(ty == 0)T::clear();
    return;
}
int main() {
    scanf("%d",&n);
    for(int i = 2;i <= n;++i)add(fa[i] = rd(),i);
    for(int i = 1;i <= n;++i)dep[i] = dep[fa[i]] + 1;
    for(int i = 1;i <= n;++i)s[i] = rd() + 1;
    s[0] = 257;
    int cnt = -1;
    for(int i = 1;i <= n;++i) {
        for(int j = 1,cur = i;j <= min(dep[i],MAX);++j,cur = fa[cur]) {
            hash[i][j].fi = (1ll * hash[i][j - 1].fi * BASE1 + s[cur]) % MODN1;
            hash[i][j].se = (1ll * hash[i][j - 1].se * BASE2 + s[cur]) % MODN2;
            hsh[i][j] = (1ull * hash[i][j].fi) << 32 | hash[i][j].se;
        }
        for(int j = dep[i] + 1;j <= MAX;++j)hsh[i][j] = --cnt;
    }
    dfs1(1);
    dfs(1,0);
    cout << ans << endl;
```

```
    return 0;
}
```

对于 30% 的数据满足：$n \leqslant 300000$

由于字符集太大无法使用后缀自动机，我们考虑后缀数组，首先在 $trie$ 上建立后缀数组，那么两个点间的 lcp 就是他们对应位置之间 $height$ 的最小值，还是建立 $height$ 数组上的笛卡尔树，它的深度也是期望 log_{Σ} 的，那么类似线段树合并，我们可以使用笛卡尔树合并，实现方法和线段树基本相同，我们只要在合并的时候顺便计算跨过根节点的贡献就行了。

时间复杂度 $O(n\log n)$ 。

```
#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstdio>
#include<cmath>
#include<vector>
#include<cctype>
#include<cstring>
using namespace std;
inline int rd() {
    register int res = 0,f = 1; register char c = getchar();
    while(!isdigit(c)) { if(c == '-') f = -1; c = getchar(); }
    while(isdigit(c)) res = (res << 1) + (res << 3) + c - '0', c = getchar();
    return res * f;
}
int n;
#define MAXN 500010
#define LOG 19
#define MAXM 256
int fa[MAXN],ch[MAXN];
#define R register
#define I inline
struct edge {
    int to,nxt;
} e[MAXN];
int edgenum = 0;
int lin[MAXN] = {0};
I void add(int a,int b) {
    e[++edgenum] = (edge) { b, lin[a] }; lin[a] = edgenum;
    return;
}
int dep[MAXN];
int f[MAXN][LOG];
int sa[MAXN], rnk[LOG][MAXN], h[MAXN], c[MAXN], c1[MAXN], c2[MAXN], rk[MAXN];
vector<int> v[MAXN];
struct table {
    int head[MAXN], nxt[MAXN], val[MAXN];
    int cntnum;
    void add(int a,int b) {
        ++cntnum; val[cntnum] = b; nxt[cntnum] = head[a]; head[a] = cntnum;
        return;
    }
    table() { cntnum = 0; }
} l;
I void make_SA(int n, int m) {
    R int p;
    R int *x = c1, *y = c2;
    for(R int i = 1;i <= m;++i)c[i] = 0;
    for(R int i = 1;i <= n;++i)++c[x[i] = ch[i]];
    for(R int i = 1;i <= m;++i)c[i] += c[i - 1];
```

```
    for(R int i = n;i >= 1;--i)sa[c[x[i]]--] = i;
    for(R int i = 1;i <= n;++i)rnk[0][i] = ch[i];
    for(R int k = 1,j = 1;(1 << j) <= n;k = k << 1,++j) {
        p = 0;
        l.cntnum = 0;
        for(R int i = 1;i <= n;++i)l.head[i] = 0;
        for(R int i = 1;i <= m;++i)c[i] = 0;
        for(R int i = 1;i <= n;++i)if(dep[sa[i]] > k)l.add(f[sa[i]][j - 1],sa[i]);
        for(R int i = 1;i <= n;++i)for(R int k = l.head[sa[i]];k != 0;k =
l.nxt[k])y[++p] = l.val[k];
        for(R int i = 1;i <= n;++i)if(dep[i] <= k)y[++p] = i;
        for(R int i = 1;i <= n;++i)++c[x[y[i]]];
        for(R int i = 1;i <= m;++i)c[i] += c[i - 1];
        for(R int i = n;i >= 1;--i)sa[c[x[y[i]]]--] = y[i];
        p = 1;
        swap(x,y);
        x[sa[1]] = 1;
        for(R int i = 2;i <= n;++i) {
            x[sa[i]] = (y[sa[i]] == y[sa[i - 1]] && y[f[sa[i]][j - 1]] == y[f[sa[i -
1]][j - 1]] ? p : ++p);
        }
        for(R int i = 1;i <= n;++i)rnk[j][i] = x[i];
        m = p;
    }
    for(R int i = 1;i <= n;++i)rk[sa[i]] = i;
    return;
}
I int LCP(int a,int b) {
    if(a == b)return dep[a];
    R int res = 0;
    for(R int i = LOG - 1;i >= 0;--i) {
        if(min(dep[a],dep[b]) >= (1 << i) && rnk[i][a] == rnk[i][b]) {
            res += (1 << i);
            a = f[a][i];b = f[b][i];
        }
    }
    return res;
}
#define pii pair<int,int>
#define fi first
#define se second
pii st[MAXN][LOG];
int lg[MAXN];
I pii mymin(pii a,pii b) {
    if(a.fi != b.fi)return min(a,b);
    else if(rand() % 2 == 0)return a;
    else return b;
}
I pii query(int l,int r) {
    R int len = lg[r - l + 1];
    return mymin(st[l][len],st[r - (1 << len) + 1][len]);
```

```
}
int trt,lc[MAXN << 1],rc[MAXN << 1],len[MAXN << 1],tot;
long long mv[MAXN];
int le[MAXN];
void build(int &rt,int l,int r) {
    if(l == r) {
        rt = l;
        return;
    }
    rt = ++tot;
    R int pos = query(l + 1,r).second;
    len[rt] = query(l + 1,r).first;
    for(R int i = l;i < pos;++i)mv[i] = mv[i] << 1 | 0,++le[i];
    build(lc[rt],l,pos - 1);
    for(R int i = pos;i <= r;++i)mv[i] = mv[i] << 1 | 1,++le[i];
    build(rc[rt],pos,r);
    return;
}
struct node {
    int lc,rc;
    int sum,l;
    node(){sum = 0;}
} t[MAXN * 30];
int ptr = 0;
I int newnode(){return ++ptr;}
int root[MAXN];
I void insert(int k) {
    root[k] = newnode();
    t[root[k]].sum = 1;t[root[k]].l = len[trt];
    R int cur = root[k],cur_ = trt;
    k = rk[k];
    for(R int i = le[k] - 1;i >= 0;--i) {
        if((mv[k] >> i) & 1) {
            cur = t[cur].rc = newnode();
            cur_ = rc[cur_];
        }
        else {
            cur = t[cur].lc = newnode();
            cur_ = lc[cur_];
        }
        t[cur].sum = 1;
        t[cur].l = len[cur_];
    }
    return;
}
#define MOD 998244353
int ans = 0;
int merge(int x,int y,int val) {
    if(x == 0 || y == 0)return x + y;
    ans = (ans + 1ll * val * t[x].l * t[t[x].lc].sum % MOD * t[t[y].rc].sum % MOD) %
MOD;
```

```
    ans = (ans + 1ll * val * t[x].l * t[t[x].rc].sum % MOD * t[t[y].lc].sum % MOD) %
MOD;
    t[x].sum += t[y].sum;
    t[x].lc = merge(t[x].lc,t[y].lc,val);
    t[x].rc = merge(t[x].rc,t[y].rc,val);
    return x;
}
void dfs(int k) {
    for(R int i = lin[k];i != 0;i = e[i].nxt) {
        dfs(e[i].to);
        root[k] = merge(root[k],root[e[i].to],dep[k]);
    }
    return;
}
int main() {
    scanf("%d",&n);
    for(R int i = 2;i <= n;++i)add(fa[i] = rd(),i),lg[i] = lg[i >> 1] + 1;
    for(R int i = 1;i <= n;++i)dep[i] = dep[fa[i]] + 1,ch[i] = rd() + 1;
    for(R int i = 1;i <= n;++i)f[i][0] = fa[i];
    for(R int k = 1;k < LOG;++k)
        for(R int i = 1;i <= n;++i)f[i][k] = f[f[i][k - 1]][k - 1];
    make_SA(n,MAXM);
    for(R int i = 1;i <= n;++i)st[i][0] = make_pair(h[i] = LCP(sa[i],sa[i - 1]),i);
    for(R int k = 1,l = 1;k < LOG;++k,l = l << 1)
        for(R int i = 1;i <= n - (l << 1) + 1;++i)st[i][k] = mymin(st[i][k - 1],st[i +
l][k - 1]);
    tot = n;
    build(trt,1,n);
    for(R int i = 1;i <= n;++i)insert(i);
    dfs(1);
    printf("%d\n",ans);
    return 0;
}
```