

2020牛客暑期多校训练营（第四场）

比赛网址

训练结果

- 时间:2020.7.20 12:00~17:00
- rank:200/1112
- 完成情况 : 3/6/10

题解

A.Ancient Distance

题意

题解1（补题byfyh）

换个方式思考，问题转化为最大距离为 x 的时候的最小覆盖关键点是几个 x 时候对应的答案是 a ， $x+1$ 时候对应的答案是 b ，那么 $a \leq K < b$ 这段的答案就都是 x 了。

具体做法是：每次选当前深度最深的点，把他往上跳 x 步的祖先 y 设为关键点，然后把这个 y 的子树全部打上标记，就这样直到整棵树被打上标记，比较关键节点数量和 k 的大小关系，这个可以用线段树维护dfs序得到。

```
#include<bits/stdc++.h>
using namespace std;
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<int,int> PII;
#define orgin first
#define mx second
inline int read()
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
const int maxn=200010;
int
n,first[maxn],ce,A,deep[maxn],dfs_clock,pa[20][maxn],tag[maxn<<2],L[maxn],R
[maxn],pos[maxn],now_ans,ans[maxn];
```

```
PII a[maxn<<2];
LL final_ans;
struct Edge
{
    int u,v,next;
    Edge() {}
    Edge(int _1,int _2,int _3):u(_1),v(_2),next(_3) {}
}e[maxn<<1];
void addEdge(int a,int b)
{
    e[++ce]=Edge(a,b,first[a]);first[a]=ce;
    e[++ce]=Edge(b,a,first[b]);first[b]=ce;
}
void dfs(int now,int fa)
{
    L[now]=++dfs_clock;
    pos[dfs_clock]=now;
    for(int i=first[now];i!=-1;i=e[i].next)
        if(e[i].v!=fa)
            deep[e[i].v]=deep[now]+1,
            pa[0][e[i].v]=now,
            dfs(e[i].v,now);
    R[now]=dfs_clock;
}
int MAX(int x,int y){return deep[x]>deep[y] ? x : y;}
int jump(int now,int step)
{
    int tmp=0;
    for(int i=0;step;i++)
    {
        if(step&1)now=pa[i][now];
        step>>=1;
    }
    return now;
}
void build(int L,int R,int o)
{
    tag[o]=0;
    if(L==R)
    {
        a[o]=make_pair(pos[L],pos[L]);
        return;
    }
    int mid=L+R>>1,lo=o<<1,ro=lo|1;
    build(L,mid,lo);build(mid+1,R,ro);
    a[o].orgin=a[o].mx=MAX(a[lo].mx,a[ro].mx);
}
void pushdown(int L,int R,int o)
{
    int lo=o<<1,ro=lo|1;
```

```

    if(tag[o]==1)a[lo].mx=a[ro].mx=0;
    else a[lo].mx=a[lo].origin,a[ro].mx=a[ro].origin;
    tag[lo]=tag[ro]=tag[o];
    tag[o]=0;
}
void update(int L,int R,int o,int ql,int qr)
{
    if(L==ql && R==qr)
    {
        tag[o]=1;
        a[o].mx=0;
        return;
    }
    if(tag[o])pushdown(L,R,o);
    int mid=L+R>>1,lo=o<<1,ro=lo|1;
    if(qr<=mid)update(L,mid,lo,ql,qr);
    else if(ql>mid)update(mid+1,R,ro,ql,qr);
    else update(L,mid,lo,ql,mid),update(mid+1,R,ro,mid+1,qr);
    a[o].mx=MAX(a[lo].mx,a[ro].mx);
}
int calc(int index)
{
    int cnt=0;
    while(1)
    {
        int x=a[1].mx;
        if(!x)break;
        int y=jump(x,index);
        cnt++;
        update(1,n,1,L[y],R[y]);
    }
    tag[1]=2;a[1].mx=a[1].origin;
    return cnt;
}
int main()
{
    while(scanf("%d",&n)!=EOF)
    {
        ce=-1;dfs_clock=0;final_ans=0;
        for(int i=1;i<=n;i++)first[i]=-1,deep[i]=0,ans[i]=n+1;
        for(int i=1;i<n;i++)A=read(),addEdge(A,i+1);
        deep[1]=1;dfs(1,0);pa[0][1]=1;
        for(int Log=1;(1<<Log)<=n;Log++)
            for(int i=1;i<=n;i++)
                pa[Log][i]=pa[Log-1][pa[Log-1][i]];
        build(1,n,1);
        for(int i=n;i;i--)ans[calc(i)]=i;
        for(int i=2;i<=n;i++)ans[i]=min(ans[i-1],ans[i]);
        for(int i=1;i<n;i++)final_ans+=ans[i];
        printf("%lld\n",final_ans);
    }
}

```

```
    return 0;  
}
```

题解2 (补题bywxg)

发现最大距离的最大值是树的最大深度，整体二分可以求出最大距离为 i 时需要点的区间，最后统计答案即可

```
#include<iostream>  
#include<cstdio>  
#include<algorithm>  
#include<cstring>  
#define ll long long  
using namespace std;  
int read()  
{  
    int k=0,f=1;char c=getchar();  
    for(;;!isdigit(c);c=getchar()) if(c=='-') f=-1;  
    for(;;isdigit(c);c=getchar()) k=k*10+c-'0';return k*f;  
}  
const int N=400055;  
int n,m,dep[N],fa[N],head[N],nextt[N],to[N],f[N];  
int mxd,tot,cnt,dfn[N],ans[N];  
void add(int a,int b)  
{  
    to[++tot]=b;  
    nextt[tot]=head[a];  
    head[a]=tot;  
}  
void dfs(int u)  
{  
    dfn[++cnt]=u;  
    for(int i=head[u];i;i=nextt[i])  
        dep[to[i]]=dep[u]+1,dfs(to[i]);  
}  
void solve(int asl,int asr,int l,int r)  
{  
    // cout<<asl<<" "<<asr<<" "<<l<<" "<<r<<endl;  
    if(asl>asr||l>r) return;  
    if(l==r)  
    {  
        for(int i=asl;i<=asr;i++)  
            ans[i]=l;  
        return;  
    }  
    int mid=asl+asr>>1;  
    ans[mid]=0;  
    for(int i=0;i<=n;i++)
```

```

    f[i]=dep[i];
    for(int i=n;i;i--)
    {
        int u=dfn[i];
        if(f[u]==dep[u]+mid||u==1)
            ++ans[mid],f[u]=-1;
        f[fa[u]]=max(f[fa[u]],f[u]);
    }
    solve(asl,mid-1,ans[mid],r);
    solve(mid+1,asr,l,ans[mid]);
}
int main()
{
    while(~scanf("%d",&n))
    {
        for(int i=2;i<=n;i++)
            scanf("%d",&fa[i]),add(fa[i],i);
        dfs(1);
        for(int i=1;i<=n;i++)
            mxd=max(mxd,dep[i]);
        solve(0,mxd,1,n);
        ll as=0;
        for(int i=1;i<=mxd;i++)
            as+=1ll*i*(ans[i-1]-ans[i]);
        printf("%lld\n",as);
        cnt=tot=mxd=0;
        for(int i=1;i<=n;i++)
            head[i]=0;
    }
    return 0;
}

```

B.Basic Gcd Problem

题意

题解

C.Count New String

题意

给了一个函数。让一段字符串每一位变成前面最大的字母。问连续调用两次函数后不同字符串个数。

题解

调用一次后发现字符串一定变成递增的串，所以再次调用只要本质不同的字符串一定不同，题目变成了求 $f(s,i,n)$ 的所有本质不同的字符串。而这些串的 trie 的大小不会超过 $10n$ 直接暴力建出后缀自动机统计不同字符串。

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<stack>
#define ll long long
using namespace std;
int read()
{
    int k=0,f=1;char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return k*f;
}
const int N=2000055;
int n,m,ch[N][26],a[N],c[N],pos[N];
int last,tot,link[N],len[N],size[N];
char s[N];
void init()
{
    last=tot=0;len[0]=0;link[0]=-1;
}
void extend(int c)
{
    int cur=++tot,p=last;len[cur]=len[last]+1;size[cur]=1;
    for(;p!=-1&&!ch[p][c];p=link[p]) ch[p][c]=cur;
    if(p==-1) link[cur]=0;
    else
    {
        int q=ch[p][c];
        if(len[p]+1==len[q]) link[cur]=q;
        else
        {
            int clone=++tot;
            len[clone]=len[p]+1;
            link[clone]=link[q];
            memcpy(ch[clone],ch[q],sizeof(ch[q]));
            for(;p!=-1&&ch[p][c]==q;p=link[p])
                ch[p][c]=clone;
            link[q]=clone;link[cur]=clone;
        }
    }
}
```

```

    }
    last=cur;
}
stack<int> st;
int main()
{
    scanf("%s",s);
    n=strlen(s);
    init();
    pos[n]=0;
    for(int i=n-1;i>=0;i--)
    {
        while(!st.empty()&&s[st.top()]<s[i]) st.pop();
        int k;
        if(st.empty()) k=n;
        else k=st.top();
        last=pos[k];
        for(int j=i;j<k;j++)
            extend(s[j]-'a');
        pos[i]=last;
        st.push(i);
    }
    ll ans=0;
    for(int i=1;i<=tot;i++)
        ans+=len[i]-len[link[i]];
    printf("%lld\n",ans);
    return 0;
}

```

F.Finding the Order

题意

在两条平行线上各有两个点，给你这上下四条线的长度，让你判断平行线的方向。

题解

其实直接看最大值来自于哪就能判，，结果我还写了几个分类讨论。。

```

#include<bits/stdc++.h>
using namespace std;
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<int,int> PII;
#define X first
#define Y second
inline int read()

```

```
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
int T,a,b,c,d;
int main()
{
    T=read();
    while(T-->0)
    {
        a=read();b=read();c=read();d=read();
        if(a<c)
        {
            if(b>=d)
            {
                puts("AB//CD");
            }
            else if(c>d)
            {
                puts("AB//CD");
            }
            else
            {
                puts("AB//DC");
            }
        }
        else if(a>c)
        {
            if(b<=d)
            {
                puts("AB//DC");
            }
            else if(a>b)
            {
                puts("AB//DC");
            }
            else
            {
                puts("AB//CD");
            }
        }
        else
        {
            if(b>d)
            {
                puts("AB//CD");
            }
            else
            {
                puts("AB//DC");
            }
        }
    }
}
```

```

        {
            puts("AB//DC");
        }
    }
}
return 0;
}
/*
*/

```

H.Harder Gcd Problem

题意

给了 $1-n$ 让你把数字两两匹配，使得他们的 gcd 都不为 1。问最多可以匹配多少对。

题解

先从大往小匹配奇数，先看看能不能和自己约数匹配。再看看能不能和它减自己的一个约数匹配。最后偶数两两匹配。莫名其妙的过掉了

```

#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<vector>
#define ll long long
using namespace std;
int read()
{
    int k=0,f=1;char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return k*f;
}
const int N=200055;
vector<int> v[N];
int T,n,m,tot,sum[N],vis[N],ans1[N],ans2[N];
int main()
{
    for(T=read();T;T--)
    {
        tot=0;int sum1=0;
        n=read();m=n/2;
        for(int i=2;i<=n;i++)
            for(int j=i+i;j<=n;j+=i)
                v[j].push_back(i);
    }
}

```

```
for(int i=n;i>1;i--)
//    for(int i=m+1;i<=n;i++)
        if(i&1&&!vis[i])
        {
            int fl=0;
            for(int j=v[i].size()-1;j>=0;j--)
                if((!vis[v[i][j]])&&(v[i][j]&1))
                {
                    ans1[++tot]=v[i][j];
                    ans2[tot]=i;
                    vis[i]=vis[v[i][j]]=1;
                    fl=1;
                    break;
                }
            if(!fl&&v[i].size()!=0)
            {
                //        sum1++;
                //        cout<<i<<endl;
                //        cout<<v[i][0]<<endl;
                for(int j=0;j<v[i].size();j++)
                    if(!vis[i-v[i][j]])
                    {
                        ans1[++tot]=i-v[i][j];
                        ans2[tot]=i;
                        vis[i]=vis[i-v[i][j]]=1;
                        fl=1;break;
                    }
            }
            if(!fl) sum1++;
        }
//    cout<<sum1<<endl;
for(int i=2;i<=m;i++)
    if((i&1)&&!vis[i])
        for(int j=2;i*j<=n;j+=2)
            if(!vis[i*j])
            {
                ans1[++tot]=i;
                ans2[tot]=i*j;
                vis[i]=vis[i*j]=1;
            }

int fl=0,pre=0;
for(int i=2;i<=n;i+=2)
    if(!vis[i])
    {
        if(!fl) fl=1,pre=i;
        else
        {
            ans1[++tot]=pre;
            ans2[tot]=i;
        }
    }
```

```

        fl=0;
    }
}
for(int i=1;i<=n;i++)
    vis[i]=0,v[i].clear();
printf("%d\n",tot);
for(int i=1;i<=tot;i++)
    printf("%d %d\n",ans1[i],ans2[i]);
}
return 0;
}

```

训练实况

开局看B F是签到 wxg hxm想B fyh分类讨论F

12:16 hxm过B fyh脑子抽了分类讨论出问题

12:44 fyh过F hxmwxg讨论H，讨论出错误做法

过场 hxmfyh想J hxmwxg想A A想出 $O(\sqrt{n}\log(n))$ 的做法

过了一会 hxm开写A

14:30 wxg过H

14:54 hxmTLEA wxg和hxm轮流调，尝试卡常无果 fyh想J不会，想I不会

垃圾时间 wxg和fyh读D，经过一番讨论莫名把正解否定。想I hxm尝试乱搞做法，否定，结束。

训练总结

wxg:

hxm:

fyh:本场我发挥得跟**一样，签到F题就脑子抽了耽误了一些时间，以后应该多打cf保证第一题的又快又对。然后在wxg和hxm讨论H的时候我觉得没有参与讨论的意义了就没有参与，又耽误了一些时间，把大部分时间耽误在想一道不可做的J题上，之后和王兴罡讨论D一道不难的题基本上已经想到做法了却在讨论后给否定了。目前想到的改进措施是把当时脑子里的想法以及注意的细节尽可能写在纸上，脑子有时候可能转不动。

From: <https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link: https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:die_java:front_page_summertrain4&rev=1595605110

Last update: 2020/07/24 23:38