A. Browser Games

题意

给定 n 个字符串，对 $i=1\sim n$找出一个最小的前缀集合，满足：

对前 i 个字符串都至少有一个前缀位于该集合且对于后面的 $n-i$ 个字符串都不存在前缀属于这个集合。

数据保证不存在一个字符串是另一个字符串前缀的情况，且内存限制为 32 megabytes

题解

首先不考虑内存限制，可以对所有串建立字典树，并用叶子结点代表每个字符串。

然后问题转化为从树上选择最少的结点集合，使得前 i 个字符串至少有一个祖先结点被选中，且后 $n-i$ 个字符串不存在祖先结点被选中。

不难发现，如果一个结点的子树中叶子结点都属于前 i 个字符串，则以该结点为根的子树答案为 1。否则该结点答案等于所有儿子结点答案之和。

建立字典树后依次处理 $i=1\sim n$ 的询问，动态更新每个结点的子树中的后 $n-i$ 个字符串个数以及以该结点为根的子树答案。

每次询问的答案记为字典树根节点的答案，注意特判 $i=n$ 的询问，因为前缀不能是空串。

然后考虑内存限制，注意到只有一个儿子结点的结点都是可以压缩的，于是树上的关键结点个数可以卡到 $O(n)$

最坏的情况是完全二叉树，这时节点个数是 $2n-1$于是开两倍空间即可。

关于建树，可以递归构建，如果当前结点的字符串个数为 1 则直接返回。

否则找到最小的当前结点的所有字符串的非公共前缀长度，然后划分字符串，继续递归，同时记录非公共前缀的位置用于后续更新操作比较。

```
const int MAXN=1e5+5,MAXS=MAXN<<1,MAXL=105;
char s[MAXN][MAXL],suf[MAXS];
int head[MAXS],nxt[MAXS],dep[MAXS],s1[MAXS],s2[MAXS],node_cnt;
vector<int> c[MAXS];
void build(int k,int d){
    s1[k]=c[k].size();
    if(s1[k]==1){
        c[k].clear();
        return;
    }
    while(true){
        int p1=c[k][0];
        bool flag=true;
        for(int p2:c[k]){
            if(s[p2][d]!=s[p1][d]){
```

```
                flag=false;
                break;
            }
        }
        if(flag)
        d++;
        else
        break;
    }
    dep[k]=d;
    for(int t:c[k]){
        int i=head[k];
        for(;i;i=nxt[i]){
            if(suf[i]==s[t][d])
            break;
        }
        if(!i){
            i=++node_cnt;
            nxt[i]=head[k];
            suf[i]=s[t][d];
            head[k]=i;
        }
        c[i].push_back(t);
    }
    c[k].clear();
    for(int i=head[k];i;i=nxt[i])
    build(i,d+1);
}
void update(int k,char *t){
    s1[k]--;
    if(s1[k]==0){
        s2[k]=1;
        return;
    }
    for(int i=head[k];i;i=nxt[i]){
        if(suf[i]==t[dep[k]]){
            s2[k]-=s2[i];
            update(i,t);
            s2[k]+=s2[i];
            break;
        }
    }
}
int main()
{
    int n=read_int();
    if(n==1){
        puts("1");
        return 0;
    }
    _rep(i,1,n){
```

```
        scanf("%s",s[i]);
        c[0].push_back(i);
    }
    build(0,0);
    _for(i,1,n){
        update(0,s[i]);
        enter(s2[0]);
    }
    int ans=0;
    for(int i=head[0];i;i=nxt[i])
    ans++;
    enter(dep[0]==0?ans:1);
    return 0;
}
```

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

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:%E7%BB%84%E9%98%9F%E8%AE%AD%E7%BB%83%E6%AF%94%E8%B5%9B%E8%AE%B0%E5%BD%95:%E7%BC%93%E5%86%B2%E5%8C%BA&rev=1629191753

Last update: **2021/08/17 17:15**