2018-2019 ICPC Southwestern European Regional Programming Contest (SWERC 2018)

比赛链接

G. Strings

题意

给定初始字符串 S_0，接下来 n 条递推式。

类型 1：$S_i=S_j[l,r]$。

类型 2：$S_i=S_jS_k$。

求 S_n 的所有字符的 Ascii 码之和，保证所有字符串长度不超过 $2^{63}-1$。

题解

首先顺序读入同时维护所有字符串长度。然后设 $\text{dp}(k,l,r)$ 表示 $S_k[l,r]$ 的 Ascii 码和，记忆化搜索逆推。

关于时间复杂度，发现逆推过程有些类似线段树的区间询问，于是每层的询问仅有 $[1,R],[L,R],[L,|S(k)|]$。

并且可以通过数学归纳法，得到 $[L,R]$ 仅有 $[L(k),R],[L,R(k)]$ 两种形式，其中 $L(k),R(k)$ 为固定值。

于是每层区间最多多分裂 4 个结点，层数为 $O(n)$，每层结点数为 $O(n)$，结点数为 $O(n^2)$。

ps. 还可以在从后往前递推询问时将多个相交询问分割成若干不相交小区间，维护每个小区间对应的询问数。然后处理每个小区间对应的询问。

于是每层最多有一个小区间分裂，共 $O(n)$ 层，也可以证明时间复杂度等于结点数等于 $O(n^2)$。

```
const int MAXN=2505,MAXL=1005,Mod=1e9+7;
map<pair<LL,LL>,int> dp[MAXN];
LL len[MAXN];
struct{
    int type,x1;
    LL x2,x3;
}opt[MAXN];
char s[MAXL];
int ps[MAXL];
int dfs(int n,LL L,LL R){
    if(dp[n].find(make_pair(L,R))!=dp[n].end())return
```

```
dp[n][make_pair(L,R)];
    if(n==0)
    return ps[R+1]-ps[L];
    if(opt[n].type==0)
    return dp[n][make_pair(L,R)]=dfs(opt[n].x1,opt[n].x2+L,opt[n].x2+R);
    else{
        if(L>=len[opt[n].x1])
        return dp[n][make_pair(L,R)]=dfs(opt[n].x2,L-len[opt[n].x1],R-
len[opt[n].x1]);
        else if(R<len[opt[n].x1])
        return dp[n][make_pair(L,R)]=dfs(opt[n].x1,L,R);
        else
        return
dp[n][make_pair(L,R)]=(dfs(opt[n].x1,L,len[opt[n].x1]-1)+dfs(opt[n].x2,0,R-
len[opt[n].x1]))%Mod;
    }
}
char cmd[10];
int main()
{
    int n=read_int();
    scanf("%s",s+1);
    len[0]=strlen(s+1);
    _rep(i,1,len[0])ps[i]=ps[i-1]+s[i];
    _for(i,1,n){
        scanf("%s",cmd);
        if(cmd[0]=='S'){
            opt[i].type=0;
            opt[i].x1=read_int(),opt[i].x2=read_LL(),opt[i].x3=read_LL()-1;
            len[i]=opt[i].x3-opt[i].x2+1;
        }
        else{
            opt[i].type=1;
            opt[i].x1=read_int(),opt[i].x2=read_int();
            len[i]=len[opt[i].x1]+len[opt[i].x2];
        }
    }
    enter(dfs(n-1,0,len[n-1]-1));
    return 0;
}
```