自动机综合

建模专题

从最简单的开始

删除后缀1

有一个S串和一个T串，长度均小于1,000,000。
设当前串为U串，然后从前往后枚举S串一个字符一个字符往U串里添加。**若U串后缀为T，则去掉这个后缀并**继续流程。

输入样例：
abcoooood
oo

输出样例：
abcod

输入样例：
abcmomoood
moo

输出样例：
abcd

暴力→直接模拟一个栈，每次操作进行一个O(N)的匹配

优化→KMP求出当前匹配位置

```
getnext();
int top=0;
for (int i=0;i<len;i++)
{
    int j=pos[top];
    s[++top]=t[i];
    while (j!=-1 && p[j+1]!=s[top]) j=next[j];
    if (p[j+1]==s[top]) ++j;
    pos[top]=j;
    if (pos[top]==lenT-1) top-=lenT;
}
```

回顾：KMP的next数组表示什么？

不匹配时
i不变
j=next[j]
（j的指针左移保证后缀依然匹配）

改成多串

删除后缀2

有一个S串和n个屏蔽词T1,T2,...Tn，S的长度和T的长度总和均小于100,000。**保证T1...Tn互不包含**。
若S串有子串在T1,...Tn中，则去掉这个子串（优先删左端点最靠左的）并继续流程。

输入样例：
begintheescapeexecutionatthebreakofdawn
2
escape
execution

输出样例：
beginthatthebreakofdawn

单串匹配→多串匹配

KMP→AC自动机

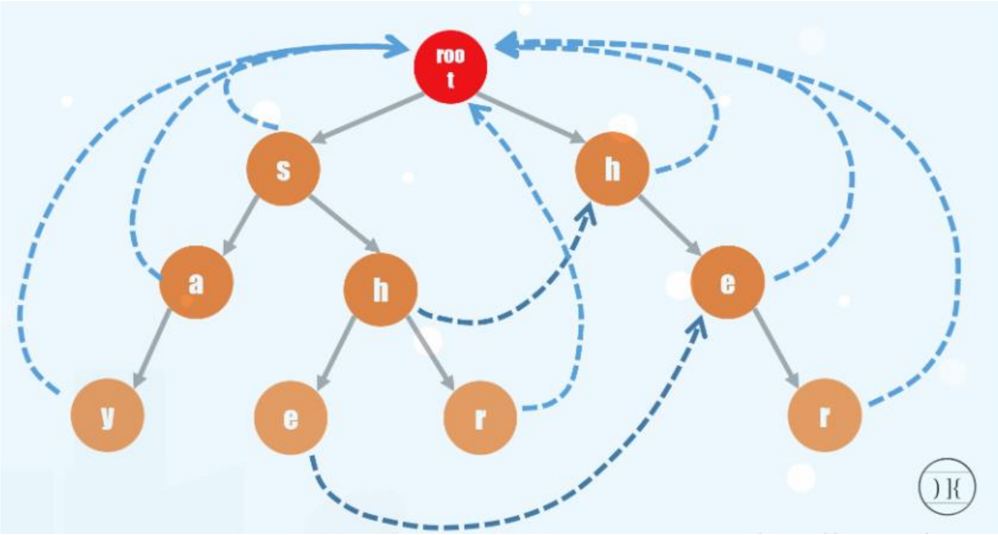
回顾：KMP的next数组和AC自动机的next(fail)指针有什么联系？

有两种常见写法：
1. getnext只计算fail
2. getnext时候更新转移(建议)

```
getnext();
for (int i=0;i<len;i++){
    pos[i]=ch[pos[q[top]]][s[i]-'a'];
    q[++top]=i;
    top-=len_[pos[i]];
}
```

回顾题目：为什么能实现优先删左端点靠左的？

```
void getnext(){
    for(int i=0;i<26;i++) ch[0][i]=1;
    next[1]=0;
    q.push(1);
    while(!q.empty()){
        int x=q.front();
        q.pop();
        for(int i=0;i<26;i++){
            int u=ch[x][i];
            if(!u){
                ch[x][i]=ch[next[x]][i];
            }
            else{
                int p=next[x];
                q.push(u);
                next[u]=ch[p][i];
            }
        }
    }
}
```



增加一些应用

方案统计

给 m 个长度1~10的单词，统计所有由a-z组成长度为 n 的串中，包含至少 其中k 个单词的方案数%MOD的值。(1<=n<=25, 0<=k<=m<=10)

输入样例：
10 2 2
hello
world

输出样例：
2

输入样例：
1 0 0

输出样例：
26

动态规划：dp[i][S][visited]表示当前长度为i，串为S（len(S)==i），已经用了单词集合为visited数组的方案数

AC自动机上的动态规划： dp[i][Node][visited]表示当前长度为i，匹配状态到了自动机上的Node节点，已经用了单词集合为visited数组的方案数

写出两个DP的转移

思考：AC自动机的作用是什么？

完成“方案统计”

```
int cur=1;
dp[0][0][0]=1;
for(int i=0;i<n;++i){
    memset(dp[cur],0,sizeof(dp[cur]));
    for(int j=0;j<=cnt;++j){
        for(int s=0;s<(1<<m);++s) if(dp[cur^1][j][s]){
            for(int c=0;c<26;++c){
                int &nxt=dp[cur][ch[j][c]][s|val[ch[j][c]]];
                nxt=(nxt+dp[cur^1][j][s])%mod;
            }
        }
    }
    cur^=1;
}
```

突然增加难度

树上统计

有一棵n个节点的树，每条边上有一个小写字母。
有m次询问，每次询问包含两个节点u,v和一个字符串st
问从u到v这个路径上每个边按顺序依次组成的字符串S(u->v)中，st出现了多少次？n<=100000,m<=100000,询问串的总长<=300000

输入样例：
12 3
1 2 w
2 3 w
3 4 x
4 5 w
5 6 w
6 7 x
7 8 w
8 9 w
9 10 x
10 11 w
11 12 w
1 7
wwx
1 12
www
1 12
w

输出样例：
2
0
8

思考：对于一个查询u,v,st，是有跨越LCA(u,v)的字符串st?

如果没有，那么只需要转化成查询S(u->LCA(u,v))和S(LCA(u,v)->v)两个串

单独考虑跨越LCA(u,v)的字符串，截取长度2*len(st)-1的部分S(u0->v0)，做KMP即可。

考虑查询S(u->LCA(u,v))这个问题
只需要计算两个前缀和S(u->root)和S(u0->root)

问题转化为：m组询问S(root->v)包含多少个st

问题转化为：m组询问S(root->v)包含多少个st

离线化：给你m个字符串st，统计每个点上他们出现的次数

对比以前的问题（查看每个点上他们的匹配状态）

对这些查询字符串建立AC自动机

DFS原始的树，在AC自动机上转移
查询==DFS到这个点v的时候，经历了多少次字符串st
DFS过程中，往下走一步，到达AC自动机上的点P，
把P的fail树子树中的所有字符串(P的所有后缀)出现次数+1
回溯时把这些字符串出现次数-1

总结：AC自动机

每一个点表示一个状态：到目前为止的最长匹配

fail树：失配时往前退到哪
→退到我的后缀

所有匹配：在这个点的fail树上

常见题目：

1. 最简单的多模式串匹配
2(重要). 自动机上的DP ← 匹配状态的转移
3. 数据结构来维护fail树

先来个基础模板题

最长公共子串

题目：略
数据范围：分别不超过250000个小写字母

输入样例：
alsdfkjfjkdsal
fdjskalajfkdsla

输出样例：
3

提示：构造S1的SAM
若trans(x,s2[i])!=null，那么x=trans(x,s2[i])，且当前答案cnt++
若trans(x,s2[i])==null，那么x在Parent树向上找到第一个满足trans(y,s2[i])!=null的，令cnt=len(y)+1,x=trans(y,s2)
若Parent树上没有，则令cnt=0,x=root

解释三种情况

一个长度为 n 的字符串 S，令 Ti 表示它从第 i 个字符开始的后缀。求

$$\sum_{1 \le i < j \le n} \mathrm{len}(Ti) + \mathrm{len}(Tj) - 2 * \mathrm{lcp}(Ti, Tj)$$

其中，len(a)表示字符串 a 的长度，lcp(a, b)表示字符串 a 和字符串 b 的最长公共前缀。

输入样例：	输出样例：
ababc	54

SAM常用技巧：后缀转前缀

翻转字符串，构建SAM
举例：ABCA→ACBA
这个SAM可以识别ACBA的所有后缀
（ABCA的所有前缀）

画出这棵树
它的parent树是什么样的？

SAM的parent树中父亲是孩子的后缀
对应翻转前的原串，父亲是孩子的前缀

原串的LCP==parent树上LCA的maxlen

现在要对两两字符串的LCA的maxlen求和

回顾：SAM中每一个节点u代表多少个字符串？

maxlen(u)-maxlen(p[u])

回顾：SAM中每一个节点u的子树代表多少个前缀字符串？

Right(u).size()

接下来就是计数问题了
枚举LCA→这棵树上任意取两个点，不在同一棵子树的方案数有多少？

枚举LCA为点u→LCP为maxlen(u)
任取两个点：C(N,2)
在同一个子树上：C(N1,2)+C(N2,2)+…+C(Nk,2)

解释这里的N,N1,N2,…Nk

```cpp
inline void init_sam() {
    memset(trans, 0, sizeof(trans));
    root = last = sz = 1;
}
inline void extend(int c, int x) {
    int p = last, np = ++sz; last = np; maxlen[np] = x; Right[np] = 1;
    for(; p && !trans[p][c]; p = pa[p]) trans[p][c] = np;
    if(!p) {pa[np] = root; return;}
    int q = trans[p][c];
    if(maxlen[q] == maxlen[p] + 1) {
        pa[np] = q;
    }else {
        int nq = ++sz;
        memcpy(trans[nq], trans[q], sizeof(trans[q]));
        pa[nq] = pa[q]; maxlen[nq] = maxlen[p] + 1; pa[q] = pa[np] = nq;
        for(; trans[p][c] == q; p = pa[p]) trans[p][c] = nq;
    }
}
inline void build(char *s) {
    int len = strlen(s);
    for(int i = 0; i < len; ++i) extend(s[i] - 'a', i + 1);
}
```

```
int dfs(int u) {
    for(int i = head[u]; ~i; i = nxt[i]) Right[u] += dfs(to[i]);
    ans -= 1LL * Right[u] * (Right[u] - 1) * (maxlen[u] - maxlen[pa[u]]);
    return Right[u];
}
inline void get() {
    init_edge();
    for(int i = 2; i <= sz; ++i) add(pa[i], i);
    dfs(root);
}
char str[N];
int main() {
    scanf("%s", str);
    int len = strlen(str);
    reverse(str, str + len);
    init_sam();
    build(str);
    ans = 1LL * len * (len - 1) * (len + 1) / 2;
    get();
    cout << ans << endl;
    return 0;
}
```

写法1：转成树的形式，树形DP

```cpp
int cmp(int a,int b){
    return st[a].len>st[b].len;
}

for(int i=0;i<sz;i++)
    c[i]=i;
sort(c,c+sz,cmp);
for(int i=0;i<sz;i++){
    int o=c[i];
    if(st[o].link!=-1){
        dp[st[o].link]+=(LL)st[st[o].link].len*cnt[o]*cnt[st[o].link];
        cnt[st[o].link]+=cnt[o];
    }
}
```

写法2：按照maxlen从大到小排序，得到拓扑序，按照拓扑序dp

改成计数排序

统计总和

给 N个数字串(1≤N≤10000)，长度和不超过100000。
写出所有的子串，把它们转成数字，删去重复的组成一个集合。
求这个集合中的数字和%MOD的值。

输入样例：
2
101
123

输出样例：
275

SAM应用：找到所有不同的子串

多个串：
1.广义后缀自动机（简单方法：每次新串前把last指向root节点）
2.串之间加一个没用的字符10

建立完SAM以后，所有节点就是所有子串
（如果用第二种方法建立，则不允许经过字符10）

问题变化成DAG上每个节点代表一些数，求他们的和

拓扑排序+DP

写出DP转移

```
inline void extend(int x){
    int p = last;
    if(ch[p][x] && len[ch[p][x]] == len[p] + 1){
        last = ch[p][x];
        return ;
    }
    int np = ++tot;
    memset(ch[np], 0, sizeof(ch[np]));
    last = np;
    len[np] = len[p] + 1;
    while(p && !ch[p][x]) ch[p][x] = np, p = fa[p];
    if(!p) fa[np] = 1;
    else{
        int q = ch[p][x];
        if(len[q] == len[p] + 1) fa[np] = q;
        else{
            int nq = ++tot;
            len[nq] = len[p] + 1;
            memcpy(ch[nq], ch[q], sizeof(ch[q]));
            fa[nq] = fa[q];
            fa[q] = fa[np] = nq;
            while(p && ch[p][x] == q) ch[p][x] = nq, p = fa[p];
        }
    }
```

另一种写法：
判重写在外面

```
void SAM_build(char *s)
{
    int len = strlen(s);
    SAM_last = SAM_root;
    for(int i = 0;i < len;i++)
    {
        if( !SAM_last->next[s[i] - '0'] || !(SAM_last->next[s[i] - '0']->len == i+1) )
            SAM_add(s[i] - '0',i+1);
        else SAM_last = SAM_last->next[s[i] - '0'];
    }
}
```

```
int solve() {
    topo_sort();//计算rank
    for(int q=si;q>=0;q--){
        int p=rank[q];
        sum[p]=0;
        cnt[p]=1;
        for(int i=0;i<10;i++) if( ch[p][i] != -1 )
            if(p==0&&i==0) continue;
            sum[p] = ( sum[p] + sum[ch[p][i]] + cnt[ch[p][i]]*i ) % mod;
            cnt[p] = ( cnt[p] + cnt[ch[p][i]]*10 ) % mod;
        }
    }
    return sum[0];
}
```

一般地，对于一个字符串 S，和 S 中第 i 个字符 x，定义子串 T=S(i..j)为一个关于 x 的识别子串，当且仅当:

1. i<=x<=j
2. T 在 S 中只出现一次

比如，对于 banana 的第 5 个字符，“nana”，“anan”，“anana”，“nan”，“banan”和“banana”都是关于它的识别子串。

请你写一个程序，计算出对于一个字符串 S，关于 S 的每一位的最短识别子串的长度。

包含x且只出现一次的字符串→|right|==1（叶子）

对于SAM上的一个点v，若|right(v)|==1，则是可识别串

点v能包含的字符串长度：记作[L+1,R]

对于以i∈[1,R−L]为开头的串，合法串的长度是R−i+1（以i开始以R结尾）

对于以i∈[R−L+1,R]为开头的串，合法串的长度是L+1（以L开始以R结尾）

两棵线段树维护

总结：后缀自动机

每一个点v代表多个字符串
这些字符串结束点都在maxlen(v)
个数为maxlen(v)-maxlen(p[v])
maxlen从大到小排序→拓扑序
Parent树：本质是表示Right集合包含关系的树

广义SAM：Trie+SAM或者暴力SAM（每个串重置last）+判重

常见题目：

1. 一个/多个模板串的子串
2. 子串计数问题（DP）
3. 前缀/后缀串问题

完全匹配：AC自动机
子串匹配：后缀自动机