序列自动机

定义

序列自动机是接受且仅接受一个字符串的子序列的自动机。

本文中用 s 代指这个字符串。

状态

若 s 包含 n 个字符，那么序列自动机包含 $n+1$ 个状态。

令 t 是 s 的一个子序列，那么 $\delta(start,t)$ 是 t 在 s 中第一次出现时末端的位置。

也就是说，一个状态 i 表示前缀 $s[1..i]$ 的子序列与前缀 $s[1..i-1]$ 的子序列的差集。

序列自动机上的所有状态都是接受状态。

转移

由状态定义可以得到$\delta(u,c)=\min\{i|i>u,s[i]=c\}$也就是字符 c 下一次出现的位置。

为什么是“下一次”出现的位置呢？因为若 $i>j$后缀 $s[i..|s|]$ 的子序列是后缀 $s[j..|s|]$ 的子序列的子集，一定是选尽量靠前的最优。

构建

从后向前扫描，过程中维护每个字符最前的出现位置 $$ \begin{array}{ll} 1 & \textbf{Input. } \text{A string } S\\ 2 & \textbf{Output. } \text{The state transition of the sequence automaton of }S \\ 3 & \textbf{Method. } \\ 4 & \textbf{for }c\in\Sigma\\ 5 & \qquad next[c]\gets null\\ 6 & \textbf{for }i\gets|S|\textbf{ downto }1\\ 7 & \qquad next[S[i]]\gets i\\ 8 & \qquad \textbf{for }c\in\Sigma\\ 9 & \qquad\qquad \delta(i-1,c)\gets next[c]\\ 10 & \textbf{return }\delta \end{array} $$ 这样构建的复杂度是 $O(n|\sum|)$

例题

计蒜客 Subsequence

题意：给定一个字符串 S输入 N 个字符串 T_i判断 T_i 是否为 S 的子序列。

题解：序列自动机模板题，对文本串 S 求出 $nxt[]$ 数组，对每个 T_i 跑 $nxt[]$ 数组即可，若提前跑出去了则不是子序列。

代码

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+5,INF=0x3f3f3f3f;
int nxt[N][26];
char s[N],t[N];
int main(){
    scanf("%s",s+1);
    int len=strlen(s+1);
    for(int i=0;i<26;i++) nxt[len][i]=nxt[len+1][i]=INF;
    for(int i=len;i>=1;i--){
        for(int j=0;j<26;j++){
            nxt[i-1][j]=nxt[i][j];
        }
        nxt[i-1][s[i]-'a']=i;
    }
    int m;
    scanf("%d",&m);
    while(m--){
        scanf("%s",t+1);
        int lent=strlen(t+1);
        int pos=0;
        bool ok=1;
        for(int i=1;i<=lent;i++){
            pos=nxt[pos][t[i]-'a'];
            if(pos==INF){
                ok=0;break;
            }
        }
        if(ok) puts("YES");
        else puts("NO");
    }
    return 0;
}
```

2020牛客国庆集训派对day4 D Shortest Common Non-Subsequence

题意：求两个字符串最短的公共的非子序列，即该序列既不是 A 的子序列也不是 B 的子序列。

题解：考虑填答案DP□设输入字符串分别为 s□t□答案为 q□

对于最后的答案, 由于它是最短的, 所以把他删掉最后一位之后, 它**或者**是 s 的子序列, **或者** 是 t 的子序列. 从空的答案开始.

如果 q 的第一个位置是 0。那么就会匹配到 s 的第一个 0 和 t 的第一个 0。

否则 q 的第一个位置是 1。那么就会匹配到 s 的第一个 1 和 t 的第一个 1。

然后考虑第二个位置，如果是 0，那么就会匹配到下一个 0。

这样一直匹配下去。直到最后匹配到 s 的末尾的后一位□t 的末尾的后一位，这时候的 q 是 s□t 的公共非子序列。

这题还要一个最小字典序□DP 的时候从后往前遍历□$dp[len(s)+1][len(t)+1]=0$□最后答案即为 $dp[0][0]$□

代码□

```
#include<bits/stdc++.h>
using namespace std;
const int N=4005;
char a[N],b[N];
int nxta[N][2],nxtb[N][2];
int dp[N][N],f[N][N];
int lena,lenb;
vector<int>ans;
int dfs(int x,int y){
    if(x==lena+1&&y==lenb+1) return 0;
    if(dp[x][y]) return dp[x][y];
    int tmp1=dfs(nxta[x][0],nxtb[y][0]);
    int tmp2=dfs(nxta[x][1],nxtb[y][1]);
    if(tmp1<=tmp2) f[x][y]=0;
    else f[x][y]=1;
    return dp[x][y]=min(tmp1,tmp2)+1;
}
void getans(int x,int y){
    if(x==lena+1&&y==lenb+1) return;
    ans.push_back(f[x][y]);
    getans(nxta[x][f[x][y]],nxtb[y][f[x][y]]);
}
int main(){
    scanf("%s%s",a+1,b+1);
    lena=strlen(a+1);
    lenb=strlen(b+1);
    nxta[lena+1][0]=nxta[lena+1][1]=lena+1;
    nxtb[lenb+1][0]=nxtb[lenb+1][1]=lenb+1;
    for(int i=lena;i>=0;i--){
        nxta[i][0]=nxta[i+1][0];
        nxta[i][1]=nxta[i+1][1];
        if(a[i+1]=='0') nxta[i][0]=i+1;
        else nxta[i][1]=i+1;
    }
    for(int i=lenb;i>=0;i--){
        nxtb[i][0]=nxtb[i+1][0];
        nxtb[i][1]=nxtb[i+1][1];
        if(b[i+1]=='0') nxtb[i][0]=i+1;
        else nxtb[i][1]=i+1;
    }
    dfs(0,0);
    getans(0,0);
```

```
    for(int i=0;i<ans.size();i++) printf("%d",ans[i]);
    return 0;
}
```