

# \$\text{dp}\$ 套 \$\text{dp}\$

## 算法简介

把 \$\text{dp}\$ 当作内层状态进行转移。

## 算法例题

### 例题一

[HDU4899](#)

#### 题意

给定一个长度为 \$n\$ 的字符串 \$S\$ 对 \$i=0\sim n\$ 求有多少长度为 \$m\$ 的 \$T\$ 满足 \$\text{LCS}(S,T)=i\$ 其中 \$S,T\$ 只含 \$\text{ATGC}\$

#### 题解

首先 \$\text{LCS}\$ 的经典算法，设 \$\text{dp}(i,j)=\text{LCS}(T[1\sim i],S[1\sim j])\$ 于是有

$$\text{dp}(i,j) = \begin{cases} \text{dp}(i-1,j-1)+1, & T[i] = S[j] \\ \max(\text{dp}(i-1,j),\text{dp}(i,j-1)), & T[i] \neq S[j] \end{cases}$$

固定 \$S\$ 不难发现 \$\text{dp}(i,\ast)\$ 的结果只与 \$\text{dp}(i-1,\ast)\$ 以及 \$T[i]\$ 相关。

不妨考虑状态表示 \$\text{dp}(i,\ast)\$ 数组。这是一个一维数组，且 \$0 \le \text{dp}(i,j) - \text{dp}(i,j-1) \le 1\$ 因此可以通过差分用一个二进制数表示状态。

考虑处理出 \$\text{dp}(i-1,\ast) \to \text{dp}(i,\ast)\$ 其中转移边为 \$T[i]=\text{ATGC}\$ 这是一个自动机。

设 \$f(i,j)\$ 表示所有长度为 \$i\$ 状态为 \$j\$ 的 \$T\$ 的数量，利用自动机进行状态转移即可。总时间复杂度 \$O(|\sum|2^{nm})\$

```
const int MAXB=15,MAXN=20,mod=1e9+7;
int a[MAXN],b[MAXN],c[MAXN],nxt[1<<MAXB][4],dp[2][1<<MAXB],ans[MAXN];
char s[MAXN];
void decode(int p,int n){
    _rep(i,1,n){
        b[i]=p&1;
        b[i]+=b[i-1];
        p>>=1;
    }
}
```

```
}  
int encode(int n){  
    int p=0;  
    for(int i=n;i;i--){  
        c[i]-=c[i-1];  
        p<<=1;  
        p+=c[i];  
    }  
    return p;  
}  
void add(int &a,int b){  
    a+=b;  
    a%=mod;  
}  
void solve(){  
    scanf("%s",s+1);  
    int n=strlen(s+1),m=read_int();  
    _rep(i,1,n){  
        if(s[i]=='A')  
            a[i]=0;  
        else if(s[i]=='G')  
            a[i]=1;  
        else if(s[i]=='T')  
            a[i]=2;  
        else  
            a[i]=3;  
    }  
    int S=1<<n;  
    _for(i,0,S){  
        decode(i,n);  
        _for(j,0,4){  
            _rep(k,1,n){  
                if(a[k]==j)  
                    c[k]=b[k-1]+1;  
                else  
                    c[k]=max(c[k-1],b[k]);  
            }  
            nxt[i][j]=encode(n);  
        }  
    }  
    int pos=0;  
    mem(dp[pos],0);  
    dp[pos][0]=1;  
    _for(k,0,m){  
        pos=!pos;  
        mem(dp[pos],0);  
        _for(i,0,S){  
            _for(j,0,4)  
                add(dp[pos][nxt[i][j]],dp[!pos][i]);  
        }  
    }  
}
```

```
    }
    mem(ans,0);
    _for(i,0,S){
        decode(i,n);
        add(ans[b[n]],dp[pos][i]);
    }
    _rep(i,0,n)
    enter(ans[i]);
}
int main()
{
    int T=read_int();
    while(T--)
    solve();
    return 0;
}
```

From:

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

Permanent link:

[https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:jxm2001:dp%E5%A5%97dp&rev=1629897328](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:dp%E5%A5%97dp&rev=1629897328) 

Last update: 2021/08/25 21:15