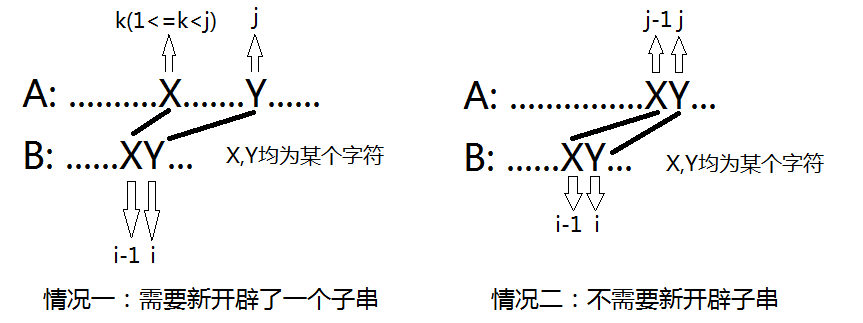
设F[i][j][K]表示B[1~i]已经匹配完了A[1~j]，一共分成了K个字符串的方案数。则分两种情况讨论：



动态转移方程为：F[i][j][K]=f[i-1][j-1][K-1]+f[i-1][j-1][K];

但如果直接开三维数组会爆空间。考虑到F[i]只会由F[i-1]转移过来，可以开一个滚动数组。

再定义pre[i][j]用来储存上一次的情况，在每次判断完后，如果不等就赋为 0，如果相等就与上一次的相加。

参考代码如下所示。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | //子串  #include<bits/stdc++.h>  using namespace std;  long long f[201][201], pre[201][201];  int main()  {  int n,m,K;  cin>>n>>m>>K;  char s1[5050], s2[5050];  cin>>s1>>s2;  f[0][0]=1;  for(int i=1; i<=n; i++)  for(int j=m; j>=1; j--)  for(int k=K; k>=1; k--)  if(s1[i-1]==s2[j-1])  {  pre[j][k]=(pre[j-1][k]+f[j-1][k-1])%1000000007;  f[j][k]=(f[j][k]+pre[j][k])%1000000007;  }  else  pre[j][k] = 0;  cout<<f[m][K]<<endl;  return 0;  } |