读取时用一个map[A][B]数组表示字母AB能变成的字母，由于只有26个字母，可以用一个26位的二进制数表示。进行两次动归：

f[i][j]表示从字符串 从 i 到 j 能变成的字母，同理也是个二进制数，

f[i][j]=f[i][j] |map[c1][c2] 存在 (f[i][k]&char[c1]&&f[k+1][j]&char[c2])

不难得到那几段字符串能变成 ‘S’

再进行一次动归：设g[i]表示前 i 个字符能变成几个 ‘S’

则g[i]=min(g[j]+1) 存在(f[i][j+1]&char['S'])

复杂度O(len3×262)+O(len2)

　　参考程序如下所示：

|  |  |
| --- | --- |
| 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  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64 | //基因序列  #include<algorithm>  #include<iostream>  #include<cmath>  #define LL long long  #define INF 99999999  #define Min(num1,num2) if(num1>num2) num1=num2  #define Max(num1,num2) if(num1<num2) num1=num2  using namespace std;  int n,f[101][101],g[101],num[101],map[101][101];  string s;  void work()  {  memset(f,0,sizeof f);  cin>>s;  int l=s.size();  for(int k=1,i=0;i<l;k++,i++)  f[k][k]=num[s[i]-'A'];  for(int p=1;p<=l;p++)  for(int i=1;i<=l;i++)  {  int j=i+p;  if(j>l)  break;  for(int k=i;k<j;k++)  for(int ci=0;ci<26;ci++)  for(int cj=0;cj<26;cj++)  if((f[i][k]&num[ci])&&(f[k+1][j]&num[cj]))  f[i][j]|=map[ci][cj];  }  int key='S'-'A';  for(int i=1;i<=l;i++)  g[i]=INF;  g[0]=0;  for(int i=1;i<=l;i++)  for(int j=1;j<=i;j++)  if((f[j][i]&num[key])&&g[j-1]!=INF)  Min(g[i],g[j-1]+1);  g[l]==INF ? printf("NIE\n") : printf("%d\n",g[l]);  }  int main()  {  freopen("GEN.in","r",stdin);  freopen("GEN.out","w",stdout);  scanf("%d\n",&n);  num[0]=1;  for(int i=1;i<=26;i++)  num[i]=num[i-1]<<1;  for(int a,b,c,i=1;i<=n;i++)  {  a=getchar()-'A';  b=getchar()-'A';  c=getchar()-'A';  map[b][c]|=num[a];  getchar();  }  int T;  scanf("%d\n",&T);  while(T--)  work();  } |