เราจะอาเรย์ที่ใช้อยู่ทั้งหมดดังนี้

a แทน string แรก

b แทน string 2

c แทน string 3

d แทน string 4

place[i][j] = ตำแหน่งของตัวที่มีตัวอักษรเดียวกันใน string ที่ i ที่อยู่ก่อนหน้า

dp[i][j][k][l] = จำนวน subsequence เมื่อใช้ a i ตัว,b j ตัว,c k ตัว,d l ตัว

ca[i][j][k][l] =

-ถ้า a[i]=b[j]=c[k]=d[l] จะบันทึกจำนวน subsequence ที่ลงท้ายด้วย a[i](หรือb[j]หรือc[k]หรือd[l]) ของ dp[i][j][k][l]

-ถ้าไม่ใช่ a[i]=b[j]=c[k]=d[l] จะไม่บันทึกอะไร

ก่อนอื่น เราจะมา gen place[i][j] ก่อนโดยทำแบบนี้

```
for(i=0;i<26;i++) num[i]=0;
for(i=1;i<=lena;i++){
    place[0][i]=num[a[i]-'a'];
    num[a[i]-'a']=i;
}
for(i=0;i<26;i++) num[i]=0;
for(i=1;i<=lenb;i++){
    place[1][i]=num[b[i]-'a'];
    num[b[i]-'a']=i;
}
for(i=0;i<26;i++) num[i]=0;
for(i=1;i<=lenc;i++){
    place[2][i]=num[c[i]-'a'];
    num[c[i]-'a']=i;
}
for(i=0;i<26;i++) num[i]=0;
for(i=1;i<=lend;i++){
    place[3][i]=num[d[i]-'a'];
    num[d[i]-'a']=i;
}
```

อธิบายโค้ด ตอนแรกให้ num[i] แทนตำแหน่งล่าสุดของตัวอักษรตัวที่ i(a คือตัวที่ 0)

แล้วทุกครั้งที่เจอตัวอักษรให้อัพเดทตำแหน่งตัวอักษรนั้นเป็นตำแหน่งปัจจุบัน(num[a[i]-‘a’]=i)

อาเรย์ place จะบันทึกตำแหน่งล่าสุดของตัวก่อนหน้าเอาไว้

ต่อมาเราจะวนลูป 4 ชั้น

```
for(i=1;i<=lena;i++){
    for(j=1;j<=lenb;j++){
        for(k=1;k<=lenc;k++){
            for(l=1;l<=lend;l++){
                if(a[i]==b[j] && b[j]==c[k] && c[k]==d[l]){
                    dp[i][j][k][l]=2*dp[i-1][j-1][k-1][l-1]-ca[place[0][i]][place[1][j]][place[2][k]][place[3][l]]+1;
                    ca[i][j][k][l]=dp[i][j][k][l]-dp[i-1][j-1][k-1][l-1]+ca[place[0][i]][place[1][j]][place[2][k]][place[3][l]];

                }else{
                    dp[i][j][k][l]= dp[i][j-1][k-1][l-1]+dp[i-1][j][k-1][l-1]+dp[i-1][j-1][k][l-1]+dp[i-1][j-1][k-1][l]
                                    -dp[i][j][k-1][l-1]-dp[i][j-1][k][l-1]-dp[i][j-1][k-1][l]-dp[i-1][j][k][l-1]-dp[i-1][j][k-1][l
                                    -dp[i-1][j-1][k][l]+dp[i][j][k][l-1]+dp[i][j][k-1][l]+dp[i][j-1][k][l]+dp[i-1][j][k][l]-dp[i-1

                }
            }
        }
    }
}
```

ก่อนอื่น เราลองมาทำความเข้าใจวิธีคิดก่อน

สมมุติว่านะ ตอนนี้เรามี subsequence อยู่แล้วดังนี้

{a,b,c,ab,bb}

แล้วปรากฏว่าต่อมาเราเจอ a[i]=b[j]=c[k]=d[l]='b' ทุกตัวเป็น b เหมือนกัน

เราจะเอากลุ่มของ subsequence เดิมมาเพิ่มด้วย กลุ่มsubsequenceเดิมที่ต่อท้ายด้วย b และ b อีกหนึ่งตัว จะได้ดังนี้ {a,b,c,ab,bb,ab,bb,cb,abb,bbb,b}

ซึ่งสังเกตว่า จะมี subsequence ซ้ำกันเกิดขึ้น ได้แก่ b,ab,bb

และสังเกตว่า subsequence ที่ซ้ำ คือ subsequence ที่ลงท้ายด้วย b(หรือตัวที่ลงท้ายด้วยตัวที่กำลังจะเติม) ที่มีอยู่ก่อนแล้ว

ดังนั้น เราต้องเก็บ subsequence ที่ลงท้ายด้วย b เอาไว้ แล้ว**หักออก**

จะได้ดังนี้ {a,c,ab,bb,cb,abb,bbb,b}

จากตัวอย่างข้างต้น เราจะสรุปคร่าวๆดังนี้

-ถ้าเจอ a[i]=b[j]=c[k]=d[l] เราจะเอา**เคสเดิม**มาเติมด้วย**เคสเดิมที่ต่อท้ายด้วยa[i](หรือb[j]หรือc[k]หรือd[l])** และเพิ่ม a[i](หรือb[j]หรือc[k]หรือd[l])เข้าไปอีกหนึ่งตัว แล้ว**ลบออกด้วยเคสที่ลงท้ายด้วย a[i](หรือb[j]หรือc[k]หรือd[l])**

จะได้สมการดังนี้

Dp[i][j][k][l]=(จำนวนเคสเดิม+จำนวนเคสเดิมที่ต่อท้ายด้วยa[i](หรือb[j]หรือc[k]หรือd[l]))+1-เคสที่ลงท้ายด้วย a[i](หรือb[j]หรือc[k]หรือd[l])

=(dp[i-1][j-1][k-1][l-1]+dp[i-1][j-1][k-1][l-1])+1-ca[place[1][i]][place[2][j][place[3][k]][place[4][l]]

=2*dp[i-1][j-1][k-1][l-1]+1-ca[place[0][i]][place[1][j]][place[2][k]][place[3][l]];

เหตุผลที่ใช้ ca[place[1][i]][place[2][j][place[3][k]][place[4][l]] เพราะช่อง place[1][i],place[2][j],place[3][k],place[4][l]) คือช่องที่ a[i]=b[j]=c[k]=d[l] ที่มีตัวอักษรเหมือนกับตอนนี้ที่อยู่ก่อนหน้า เราเลยเรียกช่องนั้นมาใช้เลย

ต่อมา หลังจากได้ subsequence ใหม่แล้ว เราจะบันทึกว่ามีตัวที่ลงท้ายด้วยตัวที่เพิ่งเติมไป(a[i]หรือb[j]หรือc[k]หรือd[l])กี่ตัว

ซึ่งมันจะเท่ากับ = จำนวนsubsequenceที่ลงท้ายด้วย(a[i]หรือb[j]หรือc[k]หรือd[l])เดิม+(จำนวนsubsequenceใหม่-จำนวนsubsequenceเดิม)

= ca[place[1][i]][place[2][j][place[3][k]][place[4][l]]+(dp[i][j][k][l]-dp[i-1][j-1][k-1][l-1]);

เหตุผลที่เพิ่ม (จำนวนsubsequenceใหม่-จำนวนsubsequenceเดิม) เพราะสังเกตว่า **subsequence ที่ได้เพิ่มมา ทุกอันจะลงท้ายด้วย(a[i]หรือb[j]หรือc[k]หรือd[l])หมด (ในตัวอย่างคือ cb,abb,bbb)**

สรุปคร่าวๆที่ผ่านมาทั้งหมด เราจะได้ว่า

Dp[i][j][k][l]=2*dp[i-1][j-1][k-1][l-1]+1-ca[place[0][i]][place[1][j]][place[2][k]][place[3][l]];

Ca[i][j][k][l]=ca[place[1][i]][place[2][j][place[3][k]][place[4][l]]+(dp[i][j][k][l]-dp[i-1][j-1][k-1][l-1]);

ที่เราพูดมาทั้งหมดนี้คือ กรณีที่ a[i]=b[j]=c[k]=d[l]

ที่นี้เราจะมาพูดถึง กรณีที่ไม่ใช่บ้าง

ถ้าไม่เหมือน เราจะเอา subsequence ของอันก่อนๆหน้ามาต่อกันให้ได้ subsequence ของอันนี้โดยใช้หลักการของ set ดังนี้

Formula for Probability of Union of Four Sets

The reason for why the formula for the probability of the union of four sets has its form is similar to the reasoning for the formula for three sets. As the number of sets increase, the number of pairs, triples and so on increase as well. With four sets there are six pairwise intersections that must be subtracted, four triple intersections to add back in, and now a quadruple intersection that needs to be subtracted. Given four sets A, B, C and D, the formula for the union of these sets is as follows:

$$P(A \cup B \cup C \cup D) = P(A) + P(B) + P(C) + P(D) - P(A \cap B) - P(A \cap C) - P(A \cap D) - P(B \cap C) - P(B \cap D) - P(C \cap D) + P(A \cap B \cap C) + P(A \cap B \cap D) + P(A \cap C \cap D) + P(B \cap C \cap D) - P(A \cap B \cap C \cap D).$$

ซึ่งมันจะทำให้เราได้สมการ

dp[i][j][k][l]= dp[i][j-1][k-1][l-1]+dp[i-1][j][k-1][l-1]+dp[i-1][j-1][k][l-1]+dp[i-1][j-1][k-1][l]-dp[i][j][k-1][l-1]-dp[i][j-1][k][l-1]-dp[i][j-1][k-1][l]-dp[i-1][j][k][l-1]-dp[i-1][j][k-1][l]

-dp[i-1][j-1][k][l]+dp[i][j][k][l-1]+dp[i][j][k-1][l]+dp[i][j-1][k][l]+dp[i-1][j][k][l]-dp[i-1][j-1][k-1][l-1];

```
}else{
    dp[i][j][k][l]= dp[i][j-1][k-1][l-1]+dp[i-1][j][k-1][l-1]+dp[i-1][j-1][k][l-1]+dp[i-1][j-1][k-1][l]
                    -dp[i][j][k-1][l-1]-dp[i][j-1][k][l-1]-dp[i][j-1][k-1][l]-dp[i-1][j][k][l-1]-dp[i-1][j][k-1][l]
                    -dp[i-1][j-1][k][l]+dp[i][j][k][l-1]+dp[i][j][k-1][l]+dp[i][j-1][k][l]+dp[i-1][j][k][l]-dp[i-1][j-1][k-1][l-1];

}
```

และเราก็จะได้ดังรูป

```
for(i=1;i<=lena;i++){
    for(j=1;j<=lenb;j++){
        for(k=1;k<=lenc;k++){
            for(l=1;l<=lend;l++){
                if(a[i]==b[j] && b[j]==c[k] && c[k]==d[l]){
                    dp[i][j][k][l]=2*dp[i-1][j-1][k-1][l-1]-ca[place[0][i]][place[1][j]][place[2][k]][place[3][l]]+1;
                    ca[i][j][k][l]=dp[i][j][k][l]-dp[i-1][j-1][k-1][l-1]+ca[place[0][i]][place[1][j]][place[2][k]][place[3][l]];

                }else{
                    dp[i][j][k][l]= dp[i][j-1][k-1][l-1]+dp[i-1][j][k-1][l-1]+dp[i-1][j-1][k][l-1]+dp[i-1][j-1][k-1][l]
                                    -dp[i][j][k-1][l-1]-dp[i][j-1][k][l-1]-dp[i][j-1][k-1][l]-dp[i-1][j][k][l-1]-dp[i-1][j][k-1][l
                                    -dp[i-1][j-1][k][l]+dp[i][j][k][l-1]+dp[i][j][k-1][l]+dp[i][j-1][k][l]+dp[i-1][j][k][l]-dp[i-1

                }
            }
        }
    }
}
```

แล้วเราก็ตอบ dp[lena][lenb][lenc][lend] นั่นเอง

ปล. Long long ด้วยนะ