**Câu 1.** Tên bài: **STRING.CPP**

Dễ thấy số phép biến đổi phụ thuộc vào vị trí $i$ đang xét của xâu S và vị trí $j$ đang xét của xâu T. Do vậy để cài đặt cho bảng phương án ta sẽ dùng mảng 2 chiều. Gọi $F[i,j]$ là số phép biến đổi ít nhất để biến xâu $S_i$ gồm $i$ kí tự phần đầu của $S$ ($S_i = S[1..i]$) thành xâu $T_j$ gồm $j$ kí tự phần đầu của $T$ ($T_j = T[1..j]$).

Dễ thấy: $F[0, j] = j$ và $F[i, 0] = i$.

Có 2 trường hợp xảy ra:

- Nếu $S[i] = T[j]$: thì ta chỉ phải biến đổi xâu $S_{i-1}$ thành xâu $T_{j-1}$. Do đó :

  $F[i,j] = F[i-1, j-1]$.

- Ngược lại, ta có 3 cách biến đổi:

  + Xoá kí tự $S[i]$: Xâu $S_{i-1}$ thành $T_j$. Khi đó :

    $F[i, j] = F[i-1, j] + 1$ (Cộng 1 là do ta đã dùng 1 phép xóa)

  + Thay thế $S[i]$ bởi $T[j]$: Xâu $S_{i-1}$ thành $T_{j-1}$ Khi đó :

    $F[i, j] = F[i-1, j-1] + 1$.

  + Chèn $T[j]$ vào sau $S[i]$: Xâu $S_i$ thành $T_{j-1}$. Khi đó:

    $F[i, j] = F[i, j-1] + 1$.

Tổng kết lại, ta có công thức QHĐ:

$F[0,j]=j$

$F[i,0]=i$

$F[i,j] = F[i-1,j-1]$ *nếu* $S[i] = T[j]$

$F[i,j] = min(F[i-1,j], F[i,j-1], F[i-1,j-1]) + 1$ nếu $S[i] \neq T[j]$

Kết quả bài toán là $F[n][m]$. Trong đó $n$ là độ dài xâu $S$ và $m$ là độ dài xâu $T$. Chú ý ta chèn thêm 1 dấu cách vào đầu 2 xâu để ta duyệt chỉ số từ 1.

Độ phức tạp của thuật toán là $O(m*n)$. Với $m$, $n$ là độ dài của 2 xâu $S$ và $T$

**Câu 2.** Tên bài: **QBSTR.CPP**

Ta xây dựng hàm mục tiêu là $F(i, j)$ có ý nghĩa là độ dài xâu con chung dài nhất của xâu $X_i$ (xâu gồm $i$ kí tự đầu tiên của xâu $X$ ) và xâu $Y_j$ (xâu gồm $j$ kí tự đầu tiên của xâu $Y$). Tương tự như bài toán 1, ta có thể suy ra công thức quy hoạch động sau:

- $F(0,j) = F(i, 0) = 0.$
- $F(i, j) = F(i-1, j-1) + 1$ *nếu* $X[i] = Y[j]$
- $F(i, j) = max(F(i-1, j), F(i, j-1))$ *nếu* $X[i] \neq Y[j]$.

Kết quả bài toán là $F(m, n)$. Trong đó $m$, $n$ là độ dài 2 xâu $X$ và $Y$. Cộng thêm dấu cách vào đầu 2 xâu để cho chạy chỉ số từ 1.

Độ phức tạp của thuật toán là $O(m*n)$. Trong đó $m$, $n$ là độ dài 2 xâu.

**Câu 3.** Tên bài: **NKPALIN.CPP**

Bài này ta có thể đảo ngược xâu đầu vào và tìm xâu con chung dài nhất của 2 xâu đó. Xâu con chung này chính là xâu đối xứng cần tìm. Công thức tìm xâu con chung dài nhất của 2 xâu A, B:

Gọi F[i][j] là độ dài xâu con chung lớn nhất đến vị trí A[i] và B[j].

+ Nếu A[i] = B[j] thì F[i][j] = F[i−1][j−1]+1.

+ Ngược lại, F[i][j] = Max(F[i−1][j] , F[i][j−1]).

Kết quả của bài toán là F[n][n] với n là độ dài của xâu (xâu được bắt đầu tính từ 1). Sau đó, ta chỉ cần truy vết để tìm ra xâu thỏa mãn.

**Câu 4.** Tên bài: **COUNTPL.CPP**

Đầu tiên, ta sẽ nhận thấy, nếu gọi $F(i)$ là số lượng xâu *palindrome* nhỏ nhất xây được từ vị trí 0 đến vị trí $i$.

Ta dễ dàng nhận thấy $F(0) = 1;$ (vì khi xét 1 kí tự đầu tiên của xâu)

Với một vị trí $j$ bất kì $(1 \leq j \leq i)$, nếu xâu $S_j, S_{j+1}, ..., S_i$ là một xâu đối xứng, thì ta dễ dàng tìm được công thức là:

$$F(i) = min(F(i), F(j-1) + 1)).$$

Tuy vậy, để bài toán chạy nhanh hơn, chúng ta cần phải xét xem xâu $S_j, S_{j+1}, ..., S_i$ là một xâu đối xứng hay không ở ngoài vòng lặp for cuối cùng.

Như thế, ta sẽ tạo 1 mảng *bool* $C(i, j) = True$ nếu xâu từ $i$ đến $j$ là 1 *palindrome* và ngược lại.

Với cách tính mảng $C(i, j)$, ta có :

+ Nếu $i = j$ thì $C(i, j) = 1$.

+ $C(i , j) = false/0$ nếu $Si \neq Sj$.

+ $C(i, j) = C(i + 1, j - 1)$ nếu $Si = Sj$.

Kết quả bài toán là $F(n-1)$. Độ phức tạp của thuật toán là $O(n^2)$