**NOEL**

* Sub1:
  + Chia dãy 2\*n thành hai dãy con
  + Sử dụng dp hai chiều với f[i][j] là kết quả tính đến vị trí thứ i của dãy một và vị trí j của dãy hai.
  + f[i][j] = f[i – 1][j – 1] + 1 nếu tm = max(f[i – 1][j] , f[i][j – 1])
* Sub2:
  + Chia dãy 2\*n thành hai dãy con
  + Tại các vị trí j của dãy thứ hai, tìm dãy xj là dãy giảm dần các vị trí i thuộc dãy thứ nhất sao cho chênh lệch giá trị của chúng <=d .
  + Ghép các xj theo thứ tự, kết quả bài toán là độ dài dãy con tăng dài nhất.

**BONUS**

* Sử dụng dp hai chiều với f[i][j] là kết quả tính đến vị trí thứ i của dãy một và vị trí j của dãy hai

f[i][j] =

**RLELCS**

* **Chú ý:** Số lượng kí tự dạng chữ trong mỗi xâu khoảng n<=500 kí tự -> ta đưa xâu về dạng pair<char, ll> biểu diễn có ll x kí tự c liền nhau.
* Yêu cầu 1: dãy ko liên tiếp
  + Sử dụng dp hai chiều với f[i][j] là kết quả nếu chọn kí tự thứ x[i] (=y[j]) đưa vào xâu con chung
  + f[i][j] = f[i’][j’] + min( X-số lượng kí tự giống x[i] từ vị trí i’+1 đến i trong xâu x , Y-số lượng kí tự giống y[j] từ vị trí j’+1 đến vị trí j trong xâu y)
  + Với công thức này ta giảm độ phức tạp xuống n^3 thay vì n^4 dựa vào mối quan hệ giữa i’, j’:
  + Nếu như min(X, Y) = X thì ta for i’ và tìm vị trí j’ gần j nhất mà Y>=X

\_ Nếu như min(X, Y) = Y thì ta for j’ và tìm vị trí i’ gần I nhất mà X>=Y

* Yêu cầu 2: dãy liên tiếp: Gọi f[i][j] là đoạn con liên tiếp dài nhất kết thúc tại i, j.