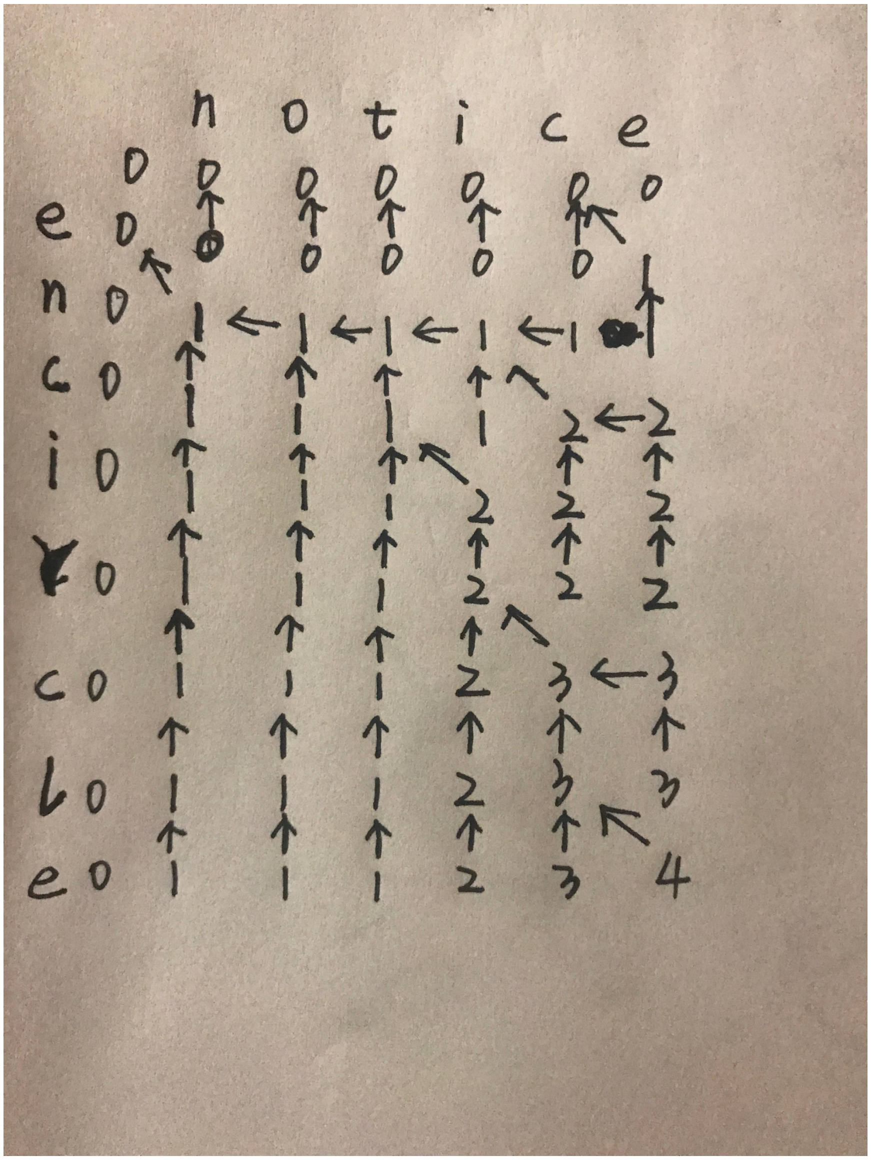
1.

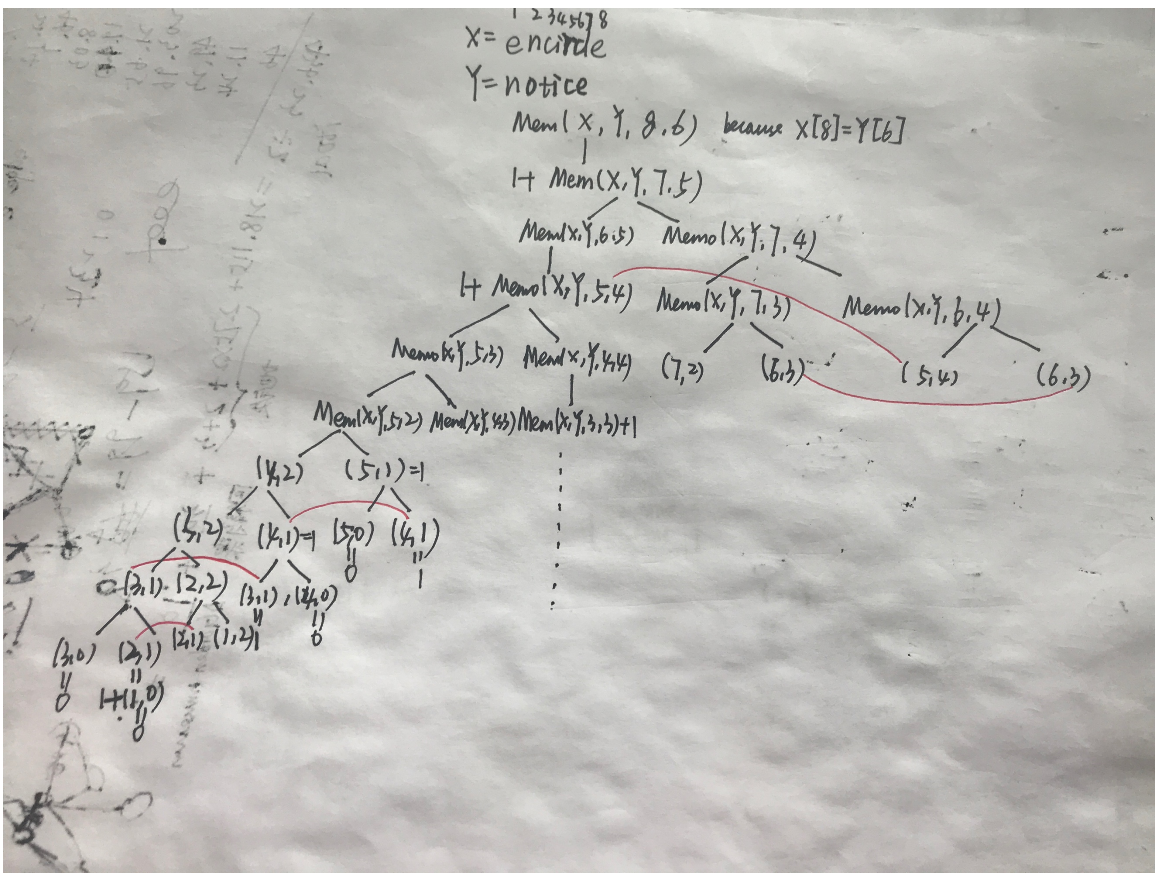


the longest common subsequence is “nice”.

2.def Memoized_LCS_Length(X,Y,m,n)

```
    let c[0..m,0..n] be new array
    for i=0 to m
        c[i, 0]=0
    for j=0 to n
        c[0,j]=0
    return Memoized_LCS_Length_AUX(X,Y,m,n,c)
```

```
def Memoized_LCS_Length_AUX(X,Y,m,n,c)
    if c[m][n]>=0
        return c[m][n]
    if(X[m]==Y[n])
        q=1+ Memoized_LCS_Length_AUX(X,Y,m-1,n-1,c)
    else
        q=max(Memoized_LCS_Length_AUX(X,Y,m-1,n,c),
Memoized_LCS_Length_AUX(X,Y,m,n-1,c))
    c[m][n]=q
  return q
```




3.

| w | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0.06 | 0.12 | 0.34 | 0.49 | 0.59 | 0.89 | 1.13 |
| 2 |  | 0 | 0.06 | 0.28 | 0.43 | 0.53 | 0.83 | 1.07 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3 | | | 0 | 0.22 | 0.37 | 0.47 | 0.77 | 1.01 |
| 4 | | | | 0 | 0.15 | 0.25 | 0.55 | 0.79 |
| 5 | | | | | 0 | 0.1 | 0.4 | 0.64 |
| 6 | | | | | | 0 | 0.3 | 0.54 |
| 7 | | | | | | | 0 | 0.24 |
| 8 | | | | | | | | 0 |

| e | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0.06 | 0.18 | 0.52 | 0.82 | 1.12 | 1.91 | 2.49 |
| 2 | | 0 | 0.06 | 0.5 | 0.64 | 0.94 | 1.67 | 2.25 |
| 3 | | | 0 | 0.22 | 0.52 | 0.8 | 1.49 | 2.04 |
| 4 | | | | 0 | 0.15 | 0.35 | 0.9 | 1.38 |
| 5 | | | | | 0 | 0.1 | 0.5 | 0.98 |
| 6 | | | | | | 0 | 0.3 | 0.78 |
| 7 | | | | | | | 0 | 0.24 |
| 8 | | | | | | | | 0 |

| root | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 3 | 3 | 3 | 4 | 6 |
| 2 | | 2 | 3 | 3 | 3 | 4 | 6 |
| 3 | | | 3 | 3 | 4 | 4 | 6 |
| 4 | | | | 4 | 4 | 6 | 6 |
| 5 | | | | | 5 | 6 | 6 |
| 6 | | | | | | 6 | 6 |
| 7 | | | | | | | 7 |

4. I think this new algorithm doesn't work, and previous problem is a counter example. For first six words, the root is the "if", but "if" is not the highest probability among those words.

5. $f(i,j) = \min_{D(i-1)\leq k\leq j}\{f(i-1,k), \max(0, j-k-m)*c\} + h(j-D(i))\ \ for D(j) \leq j \leq D$

The answer is f(n,D)

The running time is O($nD^2$)

6.

$$dp[i][j] = \begin{cases} dp[i-1][j-1] + M(X[i], Y[j]) & \text{if } X[i] = Y[j] \\ \max(dp[i-1][j] + D(X[i]),\ dp[i][j-1] + L(Y[j]), \\ \quad dp[i-1][j-1] + M(X[i], Y[j])) & \end{cases}$$

There are m*n subproblems, the running time is O(mn).

7. I choose 500 to solve this problem.

$$dp[i] = \min_{1 \le j < i}(dp[j] + c_i + \frac{h_i - h_j - 500}{2})$$

dp[i] represents the total cost when they stop in the ith hotel. $c_i$ represents the cost of the ith hotel, $h_i$ represents the distance between the hotel and the beginning.

if they spend the least cost when they arrive in the terminal, they must spend the least cost when they arrive every hotel, so this is the substructure.

Dp[1]=56

Dp[2]=131

Dp[3]=115

Dp[4]=191

Dp[5]=221

Dp[6]=266

Dp[7]=306

Dp[8]=353

So the minimum cost is 353

8.$dp[i] = \min_{m_i - m_j < 300 \wedge j < i} \{dp[j] + 1\}$

O(n)

9.dp[i][j]=min{ $\mathrm{D_{ij}} - \mathrm{dp}[i+1][j], D_{ij} - dp[i][j-1]$ }, $D_{ij}$ is the totoal easiness from problem i to problem j.

```
def function(P)
    dp[1..n][1..n]
    D[1..n][1..n]
    For i=1 to n
        Dp[i][i]=P[i]
        D[i][i]=P[i]
    for i=1 to n
        for j=i+1 to n
            D[i][j]=D[i][j-1]+P[j]
    for length =2 to n
        for i=1 to n-length+1
            dp[i][i+length-1]=min(D[i][j]-dp[i+1][ i+length-1], D[i][j]-dp[i][ i+length-2])
  return dp[1][n]
```