Thursday March 26, 2015 ①

Longest common subsequence (LCS):

Input: strings $X = x_1, \dots, x_n$ & $Y = y_1, \dots, y_m$

Goal: find length of longest string which is a subsequence in both X & Y.

Example: $X = BCDBCDA$
$Y = ABECBA$
answer = 4 for BCBA

Application: used in unix diff for comparing files

First step, define subproblem.

Look at prefixes.

Natural to have a 2-dimensional table since 2 input strings

For i & j where $0 \le i \le n$ & $0 \le j \le m$ let:

$L(i,j)$ = length of LCS of $x_1, \dots, x_i$ with $y_1, \dots, y_j$

Base cases: $L(i,0)=0$
$L(0,j)=0$

For recurrence, two cases: $X_i = Y_j$ & $X_i \neq Y_j$

Suppose $X_i \neq Y_j$:

$X = \boxed{}A \leftarrow X_i$
$Y = \boxed{}B \leftarrow Y_j$

Can't match X_i with Y_j so one or both is not in the solution, so can drop X_i or Y_j.
Try both & take best.

If X_i is dropped then $L(i,j) = L(i-1,j)$
If Y_j is dropped then $L(i,j) = L(i,j-1)$

Therefore if $X_i \neq Y_j$ then:

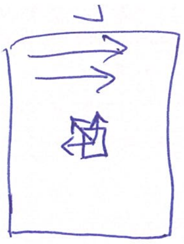
$$L(i,j) = \max\{L(i,j-1), L(i-1,j)\}$$

Suppose $X_i = Y_j$:

Either the LCS ends with $X_i = Y_j$ then
$$L(i,j) = 1 + L(i-1,j-1)$$
or it doesn't use $X_i = Y_j$ then X_i matched with some other character or Y_j matched elsewhere, so
$$L(i,j) = L(i,j-1) \text{ or } L(i-1,j).$$

Therefore if $X_i = Y_j$ then
$$L(i,j) = \max\{1 + L(i-1,j-1), L(i,j-1), L(i-1,j)\}$$

[In fact can argue that: $L(i,j) = 1 + L(i-1,j-1)$ since we might as well match $X_i = Y_j$.]

$L = i$ 

fill the table L row by row
(or column by column)

to get $L(i,j)$ look at ≤ 3 previous neighbors:
$L(i-1,j)$, $L(i,j-1)$, $L(i-1,j-1)$

LCS(X,Y):

for $i = 0 \to n$, $L(i,0) = 0$
for $j = 0 \to m$, $L(0,j) = 0$

for $i = 1 \to n$,
  for $j = 1 \to m$
    if $X_i = Y_j$
      then $L(i,j) = \max\{1 + L(i-1,j-1), L(i,j-1), L(i-1,j)\}$
      else $L(i,j) = \max\{L(i,j-1), L(i-1,j)\}$

Return($L(n,m)$)

Running time: $O(nm)$

Earlier example:

$$X = BCDBCDA$$
$$Y = ABECBA$$

$$L = \begin{array}{c|ccccccc}
 & \emptyset & A & B & E & C & B & A \\
\hline
\emptyset & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
B & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
C & 0 & 0 & 1 & 1 & 2 & \leftarrow 2 & \leftarrow 2 \\
D & 0 & 0 & 1 & 1 & 2^{\uparrow} & 2 & 2 \\
B & 0 & & & & & & \\
C & 0 & & & & & & \\
D & 0 & & & & & & \\
A & 0 & & & & & &
\end{array}$$

Closely related:

Edit distance:

Input: strings $X = x_1, \dots, x_n$ & $Y = y_1, \dots, y_m$

Goal: Min # of edits (Insertion, Deletion, Substitution) to go between X & Y

Example:

X = AAGCTGCCTAA

Y = AACCGCAATA

```
AAGCTGCC-TAA
AACC-GCAATA-
```

edit distance = 5

Variant used in BLAST — widely used in Biology to align DNA sequences

In Biology we also have a 5×5 scoring matrix

$$\delta = \begin{array}{c|c} & - \ A \ G \ C \ T \\ \begin{array}{c} - \\ A \\ G \\ C \\ T \end{array} & \begin{bmatrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} \end{array}$$

when $x_i \neq y_j$ the cost is $\delta(x_i, y_j)$

Knapsack:

Total capacity B
and n objects with:
integer weights $w_1, \ldots, w_n$
& integer values $v_1, \ldots, v_n$

Goal: find subset S of objects that
(a) fits in the backpack
(so total weight is $\leq B$)
& (b) maximizes the total value.

In other words, find subset S of objects where
(a) $\sum_{i \in S} w_i \leq B$
& (b) maximizes $\sum_{i \in S} v_i$

Application: scheduling jobs with limited resources

Two versions:
1) without repetition: one copy of each object
2) with repetition: unlimited supply of each object.

Today: without repetition - one copy of each.

What about a greedy approach?

Example: object: 1 2 3 4

object:	1	2	3	4
Values:	15	10	8	1
weights:	15	12	10	5

$$B=22$$

Sort objects by $r_i = \frac{v_i}{w_i}$ = value per unit of weight

$$r_1 > r_2 > r_3 > r_4$$

Greedy: add object 1 then add object 4

objects 1,4

total value = 16

Optimal: 2&3

total value = 18

Dynamic Programming approach:

First, define the subproblem

Initial try:

$K(j)$ = max value achievable using a subset of objects $1, \ldots, j$

Try to write a recurrence for $K(j)$ in terms of $K(1), \ldots, K(j-1)$

Take $K(j-1)$, how do we know if we can add object j to it?

Need to know how much weight is available.

For each weight b where $0 \leq b \leq B$ want to know the max value achievable.

For b & j where $0 \le b \le B$ & $0 \le j \le n$

let $K(b,j)$ = max value achievable using a subset of objects $1,\dots,j$ & total capacity b.

Goal: compute $K(B,n)$.

For recurrence for $K(b,j)$,

either: - use object j:

then gain value v_j & available capacity is $b-w_j$ so optimal is

$v_j + K(b-w_j, j-1)$

- Don't use object j:

then it's same as optimal for objects $1,\dots,j-1$ with capacity b. So it's

$K(b, j-1)$

Therefore,

if $w_j \le b$ then $K(b,j) = \max\{v_j + K(b-w_j, j-1), K(b,j-1)\}$

else $K(b,j) = K(b,j-1)$

Base cases: $K(b,0) = 0$

$K(0,j) = 0$

Recurrence for $K(\cdot, j)$ uses $K(\cdot, j-1)$
column j column $j-1$

So fill K column by column.

<u>KnapsackNoRepeat</u>$(B, w_1, \dots, w_n, v_1, \dots, v_n)$

for $j = 0 \to n$, $K(0, j) = 0$

for $b = 0 \to B$, $K(b, 0) = 0$

for $j = 1 \to n$

    for $b = 1 \to B$

        if $w_j > b$

            then $K(b, j) = K(b, j-1)$

            else $K(b, j) = \max\{K(b, j-1), v_j + K(b - w_j, j-1)\}$

Return$(K(B, n))$

Running time: $O(nB)$

Is the running time polynomial in the input size?

No, should be Poly$(n, \log B)$ which is unlikely since Knapsack is NP-complete.