

## Class 30 加强练习 7 (DP 4)

1.

- What's working behind when you enter "[yahoo.com](http://yahoo.com)" in browser?
- When database query is slow, what do you think has problem?
- 2sum,
- OO design a coffee machine.

2.

- validate BST,
- give two arrays  $a[]$  and  $b[]$ . say each entry is sum of  $a[i] + b[i]$ . Find the first K sums in ascending order

3.

- LRU cache.

4.

- Java questions: what is Final. Finally. Finalize?
- What is stringbuffer what is stringbuilder?
- What is binary search?
- How's java's hashMap implemented?
- What is linkedHashMap?
- Find first non duplicate character in a string, do it in one pass.

5.

- What is polymorphism?
- OO design a file system.

**Q1. Longest common substring/subsequence between two strings.**

**Q1.1 Longest common substring (solution 中字母必须连续)**

**Example,** student & sweden, then return "den".

**A[] = sweden;**                      **size = m**

i

**B[] = student;**                      **size = n**

j

**Solution:**

$M[i][j]$  represents the longest common substring between the first  $i$  letters from A and the first  $j$  letters from B (including the  $i$ -th letter from A and the  $j$ -th letter from B.)

**Base case:**

$M[0][0] = 0$

$M[0][j] = 0$  for all  $j$

$M[i][0] = 0$  for all  $i$

**Induction rule:**

$M[i][j] = M[i-1][j-1] + 1$   
0

Case 1: If  $A[i] == B[j]$   
else

ind_j	0	1	2	3	4	5	6	7
i		s	t	u	d	e	n	t
0	0	0	0	0	0	0	0	0
1 s	0	1	0	0	0	0	0	0
2 w	0	0	0	0	0	0	0	0
3 e	0	0	0	0	0	1	0	0
4 d	0	0	0	0	1	0	0	0
5 e	0	0	0	0	0	2	0	0
6 n	0	0	0	0	0	0	3	0

return 3

## Q1.2 Longest common sub-sequence (字母可不连续)

A == studen t w  
i

5

B == Sweden w s  
j

### Solution:

$M[i][j]$  represents the longest common subsequence between the first  $i$  letters from A and the first  $j$  letters from B (might not include the  $i$ -th letter from A and the  $j$ -th letter from B.)

### Base case:

$M[0][0] = 0$

$M[0][j] = 0$  for all  $j$

$M[i][0] = 0$  for all  $i$

### Induction rule:

$M[i][j] = M[i-1][j-1] + 1$   
 $\max(M[i-1][j], M[i][j-1])$

if  $A[i] == B[j]$

if  $A[i] != B[j]$

ind_j	0	1	2	3	4	5	6	7
i		s	t	u	d	e	n	t
0	0	0	0	0	0	0	0	0
1 s	0	1	1	1	1	1	1	1
2 w	0	1	1	1	1	1	1	1
3 e	0	1	1	1	1	2	2	2
4 d	0	1	1	1	2	2	2	2
5 e	0	1	1	1	x	x	x	x
6 n	0	x	x	x	x	x	x	x

## Q2 Longest Increasing Sub-Array vs Sub-Sequence problem

### Q2.1 Longest Ascending Subarray

Given an unsorted array, **find the length** of the longest subarray in which the numbers are in ascending order. For example: If the input array is {7, 2, 3, 1, 5, 8, 9, 6}, the subarray with the most numbers in ascending order is {1, 5, 8, 9} and the expected output is 4.

#### Solution:

来Offer网版权所有，不允许任何组织或个人将本讲义share给除本课注册学生之外的第三方

$M[i]$  represents from the 0-th element to the  $i$ -th element, the value of the longest ascending subarray (including the  $i$ -th element)

#### Base case:

$$M[0] = 1$$

#### Induction rule:

$$M[i] = \begin{matrix} M[i-1] + 1 \\ 1 \end{matrix}$$

if  $\text{input}[i] > \text{input}[i-1]$   
else

## 2.2 Longest Ascending Subsequence

Given an unsorted array, find the length of the longest **subsequence** in which the numbers are in ascending order.

For example,

{1, 2, 4, 3, 7, 6, 4, 5}

longest ascending subsequence is {1, 2, 3, 4, 5}

### Solution:

$M[i]$  represents from the 0-th element to the  $i$ -th element, the value of the longest ascending subsequence (including the  $i$ -th element)

### Base case:

$$M[0] = 1$$

### Induction rule:

$$M[i] = \max_{1 \leq j < i} (M[j]) + 1$$

where  $\text{input}[i] > \text{input}[j]$  for  $0 \leq j < i$   
if there is no such  $j$

input = {1, 2,	4,	3,	7,	6, 4, 5}
M = 1	1+1=2	3	3	4
		max(2+1,	max(2+1,	max(3+1
		1+1)	1+1)	3+1
				2+1
				1+1)

Time =  $O(n^2)$



**Q3.** There is an array of positive integers with no duplicate, in which each integer represents a piece of Pizza's size, you and your friend take turns to pick pizza from the array. Your friend's strategy is pretty simple, he always picks a larger size pizza from either end of the remaining pizzas each time. Your strategy is also to pick a piece of pizza from either end each time. What's the **largest** total sum of all pizza **you can pick assuming you start first**.

Example: 2 4 10 3

index 0 1 2 3 4 5 6 7 8 9  
 2 3 4 1 xxxxx 4 6 5

$M[i][j]$  represents [from the i-th pizza to the j-th pizza] the largest sum I can eat when I pick first.

$M[0][9]$  = case 1: if we take the left pizza

$M[2][9] + \text{input}[0]$

$M[1][8] + \text{input}[0]$

if  $\text{input}[1] > \text{input}[9]$

else



$M[1][8] + \text{input}[0]$   
 case 2: if we take the right pizza  
 $M[1][8] + \text{input}[9]$   
 $M[0][7] + \text{input}[9]$

else

if  $\text{input}[0] > \text{input}[8]$   
 else

)

### Base case:

1 piece of pizza:  $M[i][i] = \text{input}[i]$

2 adjacent pieces of pizza:  $M[i][i+1] = \max(\text{input}[i], \text{input}[i+1])$

### Induction rule:

$M[i][j] = \max($

case 1: if i take the left pizza

$M[i+2][j] + \text{input}[i]$

$M[i+1][j-1] + \text{input}[i]$

case 2: if i take the right pizza

$M[i+1][j-1] + \text{input}[j]$

$M[i][j-2] + \text{input}[j]$

if  $\text{input}[i+1] > \text{input}[j]$

else

if  $\text{input}[i] > \text{input}[j-1]$

else

)

Time =  $O(n^2)$

T

## DP 的解题常用方法:

1. 一维的original data (such as a rope, a word, a piece of wood), 求MAX or MIN (cut, merge, etc..)
  - 1.1. if the **weight** of each smallest element in the original data is identical/similar
    - 1.1.1. e.g. **identical**: 1 meter of rope
    - 1.1.2. e.g. **similar**: a letter, a number

Then this kind of problem is usually simple:

**Linear scan and look back to the previous element(s)**

来Offer网版权所有, 不允许任何组织或个人将本讲义share给除本课注册学生之外的第三方



For example:

**Longest Ascending Subarray (when at  $i$ , look back at  $i-1$ )**

**Longest Ascending Subsequence (when at  $i$ , look back at  $1....i-1$ )**

**Cut rope**

**Cut palindrome**

1.2. If the **weight** are not the same:

1.2.1. e.g. DP1 课后题: 沙子归并

1.2.2. e.g. 强化练习题: 切木头

从中心开花,  $[index = 0.1.2.3. N-1]$ , for each  $M[i, j]$ , we usually need to try out all possible  $k$  that  $(i < k < j)$ ,  $M[i, j] = \max (M[i, k] +/- * M[k, j] )$  (for all possible  $k$ )

1.3 Pizza 问题, 两头凑

2. 2D的original data

2.1. Matrix 问题, 大班课基本涵盖

2.2. **Two String** 寻找 Minimum Edit Distance, Longest Common

Substring/Subsequence, 一般解题方法都是  $S1$ 的前 $i$ 个letter 和  $S2$ 的前 $j$ 个letter 比较。Induction rule 一般要看  $M[i][j]$  和  $M[i-1][j]$ ,  $M[i][j-1]$ ,  $M[i-1][j-1]$  之间关系

### Solution 2:

HashMap<key = <a,b>, value = Set<Point>> // normal case

HashMap<key = x-axis, value = Set<Point>> // a = +∞

```
for Pi {  
    for Pj {  
        determine the line  $y = a * x + b$   
        <key = <a, b>, value.add(Point)>  
    }  
}
```

Find the line with the max number of points.

Time =  $O(n^2)$

Corner case: if  $a = +\infty$  which means that the line is perpendicular to the x-axis.

**Q4.2** Given an array of coordinates of points, how to find **the largest number of points** that can form a set such that any pair of points in the set can form a line with positive slope.

**Solution:**

$P1 < x_1, y_1 > P2 < x_2, y_2 >$

slope =  $(y_2 - y_1) / (x_2 - x_1) > 0$

when  $x_1 < x_2$ , we must have  $y_1 < y_2$ .

The question can be converted to the **LONGEST ASCENDING SUB-SEQUENCE** Q(2.2).

Step1: Sort the points according to their x-coordinates.  $\Rightarrow A[N]$

Time =  $O(n \log n)$

Step2:  $A[N] = <x_0, y_0>, <x_1, y_1> \dots <x_n, y_n>$

Find the longest ascending subsequence in  $A[N]$  according to their y-coordinates.

Time =  $O(n^2) \Rightarrow O(n \log n)$

**Q6 (Advanced topic) count-array problem**

*Given an array  $A[N]$  with all positive integers from  $[1...N]$ . How to get an array  $B[N]$  such that  $B[i]$  represents how many elements  $A[j]$  ( $j > i$ ) in array  $A[]$  that are smaller than  $A[i]$ . For example, given  $A[N] = \{ 4, 1, 3, 2 \}$ , we should get  $B[N] = \{ 3, 0, 1, 0 \}$ . Requirement: Time =  $O(n \log n)$ .*

**性质：**在 $\log(n)$ 层中，combine function 能够让每个element 都会和其他所有的元素compare至少一次。总的时间复杂度依然是 $O(n \log n)$