Lecture :- DP Problems

Agenda

- Longest increasing subsequence
- Russian doll envelopes
- Longest palindromic substring
- Palindrome partioning.

LIS

Qu Given arr[n], find length of longest increasing subsequence

Example arr[] = { 10 3 12 7 2 9 11 20 11 3 6 8 }

Ans = { 3, 7, 9, 11, 20 } [5 Ans]

Brute force

Get all subsequences —

for every subsequence —

check whether inc^r or not

TC: $O(2^n)$

Most obvious approach

	0	1	2	3	4	5	6	7	8	9	10	11
arr[] =	10	3	12	7	2	9	11	20	11	3	6	8

lis(0 – 11)

pick

max { 1 + $\boxed{\text{lis}(0-10)}$

{ 3, 7, 9, 11, 20 }

Don't pick

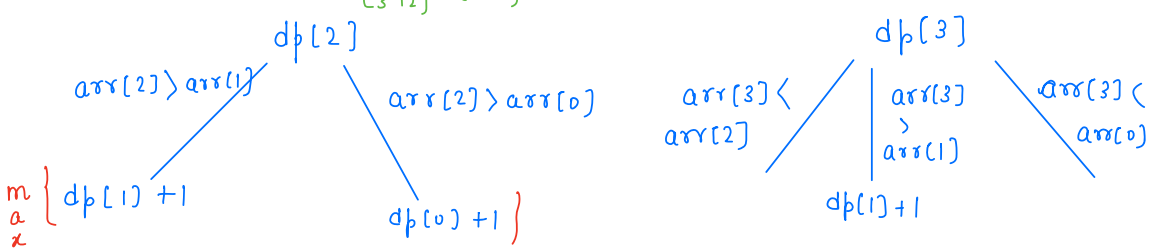
lis (0 – 10) }

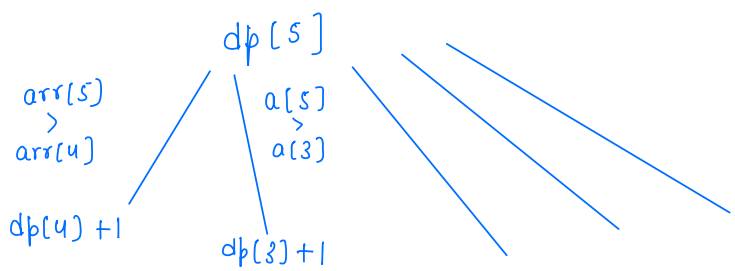
We need to know the elements that we are picking.

Approach

dp[i] = LIS from (0 - i) ending with ith idx.

	0	1	2	3	4	5	6	7	8	9	10	11
arr[] =	10	3	12	7	2	9	11	20	11	3	6	8
dp[] =	1	1	2	2	1	$\boxed{3}$	4	5	4	2	3	4
	[10]	[3]	[3 12]	[3 7]								





Code:

```
int LIS( arr[] ) {

    n = arr.length;

    dp[n];

    dp[0] = 1;

    for( i=1; i<n; i++) {

        for (j=0; j<i; j++) {

            if( arr[i] > arr[j]) {

                dp[i] = max(dp[i], dp[j]+1);

            }

        }

    }

    return {max of dp[]};

}
```

Russian doll envelopes | LC – Hard |

Given n envelopes with their height and width.

Note: An envelope i can fit in another envelope j if :-

1.) $h[i] < h[j]$

2.) $w[i] < w[j]$

3.) Envelope j is empty.

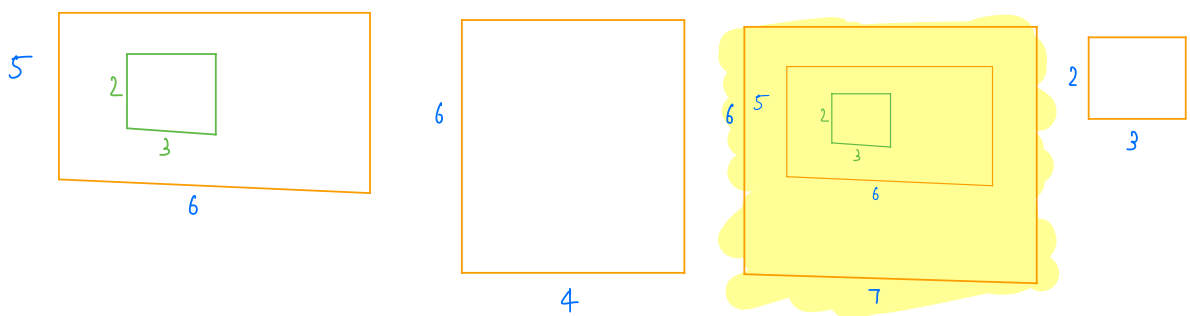
find max envelopes that can fit inside another.

Example:

$$h = \begin{array}{cccc} 0 & 1 & 2 & 3 \\ [\,5 & 6 & 6 & 2\,] \end{array}$$

$$w = [\,6 \quad 4 \quad 7 \quad 3\,]$$

ans = 3



$$h = \begin{array}{cccc} 0 & 1 & 2 & 3 \\ [\,2 & 3 & 1 & 3\,] \end{array}$$

$$w = [\,3 \quad 2 \quad 1 \quad 3\,]$$

ans = 2

Observation:

$$h = [\ \overset{0}{2} \quad \overset{1}{3} \quad \overset{2}{1} \quad \overset{3}{3}\ ]$$

$$w = [\ 3 \quad 2 \quad 1 \quad 3\ ]$$

for any envelope i —

It can fit in either right | left

↓

Sort, check for its to fit all the envelopes on left

$$h = [\ \overset{0}{5} \quad \overset{1}{6} \quad \overset{2}{6} \quad \overset{3}{2}\ ]$$

$$w = [\ 6 \quad 4 \quad 7 \quad 3\ ]$$

↓ sort [h | w]

$$h = [\ 2 \quad 5 \quad 6 \quad 6\ ]$$

$$w = [\ 3 \quad 6 \quad 4 \quad 7\ ]$$

1	2	2	3
$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$	$\begin{bmatrix} 5 & 6 \\ 2 & 3 \end{bmatrix}$	$\begin{bmatrix} 6 & 4 \\ 2 & 3 \end{bmatrix}$	$\begin{bmatrix} 6 & 7 \\ 5 & 6 \\ 2 & 3 \end{bmatrix}$

Code:-

```
int russianEnvelopes( h[], w[] ) {
    n = h.length;
    Pair[] envs = new Pair[n];
    // Store h & w in envs
       and sort the envs.

    // Apply LIS.
    dp[n];
    dp[0] = 1;
    for(i=1; i<n; i++) {
        for (j=0; j<i; j++) {
            if( envs[i].h > envs[j].h &&
                envs[i].w > envs[j].w) {
            dp[i] = max(dp[i], dp[j]+1);
            }
        }
    }
}
return max( dp[] );
}
```

TC: $O(n^2)$

SC: $O(n)$

Break: 8:05 - 8:15

Qu: Given a string, find longest palindromic substring.

s = b b d a d b ans = 5

s = a b c c b c ans = 4

Approach

0	1	2	3	4	5
a	b	c	c	b	c

		a (0)	b (1)	c (2)	c (3)	b (4)	c (5)	
a	0	a: t	ab: f	abc: f	abcc: f	abccb: f	abccbc: f	l=6
b	1	X	b: t	bc: f	bcc: f	bccb: t	bccbc: f	l=5
c	2	X	X	c: t	cc: t	ccb: f	ccbc: f	l=4
c	3	X	X	X	c: t	cb: f	cbc: t	l=3
b	4	X	X	X	X	b: t	bc: f	l=2
c	5	X	X	X	X	X	c: t	l=1

dp[i][j] = substring [i-j] is palindrome or not?

Code:

```
boolean[][]  storePalindromes( String s) {
        n = s.length();
        dp[n][n];
        for( len=1; len<=n; len++) {
            i=0;
            j= len-1;
            while( j<n) {
                if( len==1) {
                    dp[i][j] = true;
                }
                else if( len==2) {
                    if( s[i] == s[j] ) {
                        dp[i][j] = true;
                    }
                } else {
                    if( s[i] == s[j] ) {
                        dp[i][j] = dp[i+1][j-1];
                    }
                }
                i++;
                j++;
            }
        }
        return dp;
    }

int longestPalindrome( String s) {
        dp[][] = storePalindrome(s);      ans=0;
        for (i=0; i<n; i++) {
            for (j=0; j<n; j++) {
                if ( dp[i][j] == true) {
                    ans = max (ans, j-i+1);
                }
            }
        }
    return ans;
```

Palindrome partioning

<u>Qu</u> Min palindrome cut such that all broken strings are palindrome.

1) a n a | o n o | a a ans = 2

2) a | b c b | a b a a b a | a ans = 3

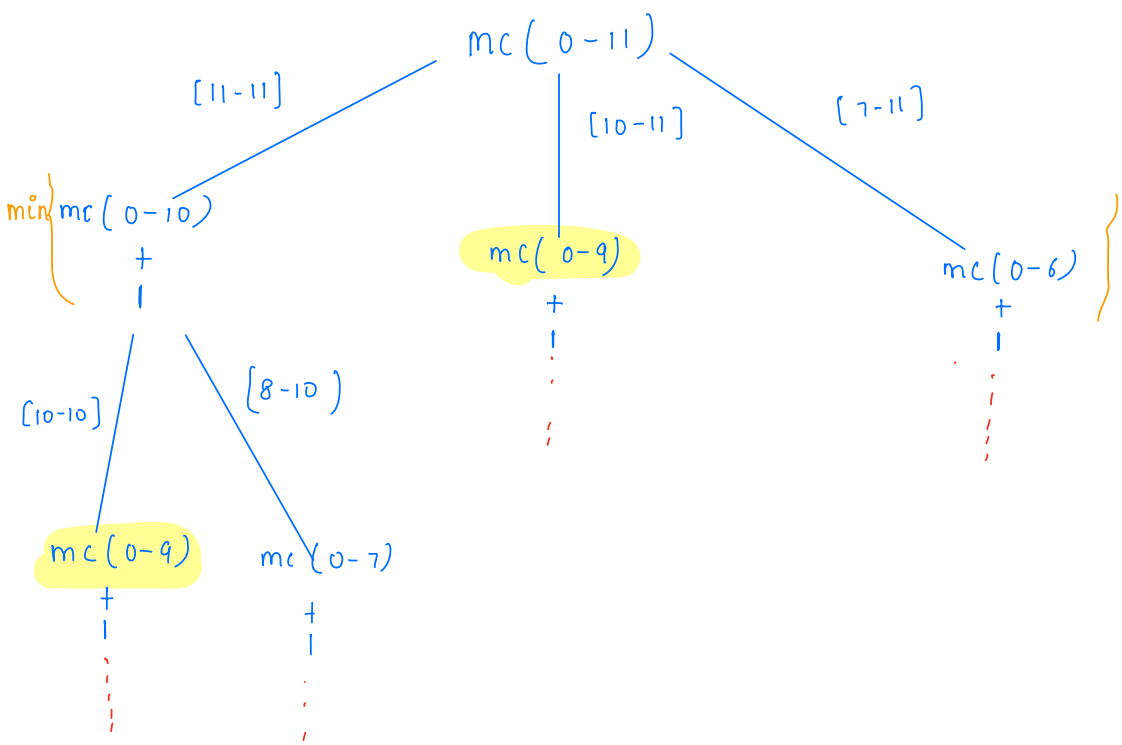
3) a b c b a | b a a b | a a ans = 2

4) a | b | c | d | e ans = 4

5) a b c b a ans = 0

Dry run:

0	1	2	3	4	5	6	7	8	9	10	11
a	b	c	b	a	b	a	a	a	b	a	a



Overlapping subproblem
Optimal substructure } DP

Changing factors

end idx [dp[n]]

dp[i] = Min cut required to make all broken strings from (0-i) palindrome

ans ⇒ dp[n-1]

Code:

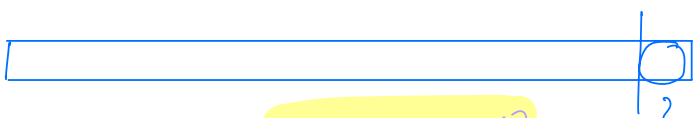
```
int mincut(string str){
    n = str.length();
    dp[n];
    dp[0] = 0;
    pal[][] = storePalindrome(str);
    for(i=1; i<n; i++){
        ans = ∞;
        if(pal[0][i] == true){
            ans = 0;
        } else {
            for(j=i; j>=0; j--){
                if(pal[j][i]){
                    ans = min(ans, dp[j-1]) + 1;
                }
            }
        }
        dp[i] = ans;
    }
    return dp[n-1];
}
```

TC: $O(n^2)$
SC: $O(n^2)$

Dry run:

0	1	2	3	4	5	6	7	8	9	10	11
a	b	c	b	a	b	a	a	a	b	a	a
							j	j	j	j	i

```
if(pal[j][i]){
    1 + dp[   ];
}
```



Thankyou ☺