Todays Content

a. Matrix Multiplication

b. Matrix Chain Multiplication

c. Longest Increasing Subsequence

Start: 9:05PM

Dp =

a. Recursion

b. Overlapping Sub Problems

Con: Solve it once, store it & re-use it

Matrix Multiplication

Rule: $A[3\ 4] * B[4\ 2] = R[3\ 2]$

$A[2\ 5] * B[5\ 3] = R[2\ 3]$

$A[3\ 4] * B[5\ 2] =$ not possible ($4 \neq 5$)

Note: $A_{r_1 * c_1} * B_{r_2 * c_2} = R[r_1\ c_2]$

$==$

We can multiply if $c_1 == r_2$

Cost: $r_1\ c_1 == r_2\ c_2$

$A[3\ 4] * B[4\ 2] = R[3\ 2]$

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array}\begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 2 & 0 & 1 \\ 3 & 2 & 1 & 4 \\ -1 & 0 & 1 & 2 \end{bmatrix} \end{array} * \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{bmatrix} 2 & 1 \\ 1 & 0 \\ -1 & 1 \\ 2 & -1 \end{bmatrix} \end{array} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array}\begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{bmatrix} 6 & 0 \\ 15 & 0 \\ 1 & -2 \end{bmatrix} \end{array}$$

Total no: iterations =

= # Total ele * Iteration for sing ele

$= 3 * 2 * 4$

Total iter $= r_1 * c_2 * r_2$

$R[0][0] = 0^{th}$ row in A * 0^{th} col in B :

$R[0][1] = 0^{th}$ row in A * 1^{th} col in B :

$R[1][0] = 1^{th}$ row in A * 0^{th} col in B :

$R[1][1] = 1^{th}$ row in A * 1^{th} col in B :

$R[2][0] = 2^{th}$ row in A * 0^{th} col in B :

$R[2][1] = 2^{th}$ row in A * 1^{th} col in B :

Obs: $A_{r_1 * c_1} * B_{r_2 * c_2} =$ Total iter $= r_1 * r_2 * c_2$

$c_1 == r_2$ is given

Chain Basics:

$M_1 * M_2 * M_3 = R[3*4]$

$[3*5] \quad [5*7] \quad [7*4]$

Case-I: $[M_1\ M_2]\ M_3$

$\{M_1 * M_2\} = R * M_3 = \text{Final}$

$3*5 \quad 5*7 \quad 3*7 \quad 7*4 \quad 3*4$

Iterations ← Cost = $3*5*7\ (105) + 3*7*4\ (84) = 189.$

Case-II: $M_1\ [M_2\ M_3]$

$M_1 * \{M_2 * M_3\} = M_1\ R = \text{Final}$

$5*7 \quad 7*4 \quad 3*5 \quad 5*4 \quad 3*4$

Cost = $5*7*4\ (140) + 3*5*4\ (60) = 200$

Note: no: of iterations are different. Final result is same

Q. Given N matrices find min iterations to multiply all matrices?

Input:

N=4 : $M_1 \quad M_2 \quad M_3 \quad M_4$

$5*3 \quad 3*6 \quad 6*4 \quad 4*8$

d[5] : 0 1 2 3 4

5 3 6 4 8

$M_1 \quad M_2 \quad M_3 \quad M_4$

N=5 : $M_1 \quad M_2 \quad M_3 \quad M_4 \quad M_5$

$3*6 \quad 6*4 \quad 4*8 \quad 8*2 \quad 2*7$

d[6] : 0 1 2 3 4 5

3 6 4 8 2 7

Note: For N matrices, Input: dimensional size = d[N+1]

Extract Inf Input:

N=5: M_1 M_2 M_3 M_4 M_5

3*6 6*4 4*8 8*2 2*7

d[6]: 0 1 2 3 4 5

3 6 4 8 2 7

M_1 M_2 M_3 M_4 M_5

r * c

$M_1 = d[0] * d[1]$

$M_2 = d[1] * d[2]$

$M_3 = d[2] * d[3]$

$M_4 = d[3] * d[4]$

r c

$M_i = d[i-1] * d[i]$

Qn: N=5 d[6] = [2 3 4 8 6 5]

0 1 2 3 4 5

M_1 M_2 M_3 M_4 M_5

Say:

Mul all mat[1-3] res mat size = M_1 M_2 M_3 = R R

2*3 3*4 4*8 = 2*8 d[0]*d[3]

Mul all mat[2-4] res mat size = M_2 M_3 M_4 = R R

3*4 4*8 8*6 3*6 d[1]*d[4]

Mul all mat[1-4] res mat size = M_1 M_2 M_3 M_4 = R R

2*3 3*4 4*8 8*6 = 2*6 d[0]*d[4]

Ques: For N matrices, Find min cost to multiply all mat[1, N]

Given Input dimensional size = d[N+1]

$M_i = d[i-1] * d[i]$

If we mull all mat[i-j] = d[i-1] * d[j]

8. Given N matrices dimensions,
Calculate min iterations to multiply all of them?

Ex: N=5 dim[6] = { 3 1 2 6 4 2 } =
M_1 M_2 M_3 M_4 M_5

Min iterations to multiply all matrices from [1-5]

Mcm(1-5)

[M1]*[M2 M3 M4 M5]
Mcm(1-1) = a_1 : $A_{3\,1}$
Mcm(2-5) = b_1 : $B_{1\,2}$
Total = $a_1 + b_1 + 6$

[M1 M2]*[M3 M4 M5]
Mcm(1-2) = a_2 : $A_{3\,2}$
Mcm(3-5) = b_2 : $B_{2\times2}$
Total = $a_2 + b_2 + 12$

[M1 M2 M3]*[M4 M5]
Mcm(1-3) = a_3 : $A_{3\,6}$
Mcm(4-5) = b_3 : $B_{6\,2}$
Total = $a_3 + b_3 + 36$

[M1 M2 M3 M4] [M5]
Mcm(1-4) = a_4 : $A_{3\times4}$
Mcm(5-5) = b_4 : $B_{4\times2}$
Total = $a_4 + b_4 + 24$

[M2] [M3 M4 M5]
Mcm(2-2) = a_5
Mcm(3-5) = b_5
Total = $a_5 + b_5 + res$

[M2 M3] [M4 M5]
Mcm(2-3) = a_6
Mcm(4-5) = b_6
Total = $a_6 + b_6 + res$

[M2 M3 M4] [M5]
Mcm(2-4) = a_7
Mcm(5-5) = b_7
Total = $a_7 + b_7 + res$

Recursion: Ass: Min cost to mul all mat from i..j {matrix numbers}

```
int Mcm(int d[], int i, int j){
    if(i==j){ return 0;}

    // Sub Problems
```

if we mul all mat[i-j] = d[i-1] * d[j]

$(M_i)(M_{i+1}\ M_{i+2}\ M_{i+3} \cdots M_{j-1}\ M_j) = mcm(i,i) + mcm(i+1,j) + res$

$(M_i\ M_{i+1})(M_{i+2}\ M_{i+3} \cdots M_{j-1}\ M_j) = mcm(i,i+1) + mcm(i+2,j) + res$

$(M_i\ M_{i+1}\ M_{i+2})(M_{i+3} \cdots M_{j-1}\ M_j) = mcm(i,i+2) + mcm(i+3,j) + res$

$\vdots$

$(M_i\ M_{i+1} \cdots M_k)(M_{k+1}\ M_{k+2} \cdots M_j) = mcm(i,k) + mcm(k+1,j) + d[i-1]*d[k]*d[j]$

dim: $d[i-1]*d[k]$ * $d[k]*d[j]$ — result mat cost

$r_1\ c_1\ \ r_2\ c_2$

$\vdots$

$(M_i\ M_{i+1}\ M_{i+2} \cdots M_{j-1})(M_j) = mcm(i,j-1) + mcm(j+j) + res$

```
    long ans = INT_MAX;
    for(int k=i; k<j; k++){
        ans = min(ans, mcm(d,i,k) + mcm(d,k+1,j) + d[i-1]*d[k]*d[j]);
    }
    return ans;
}
```

dpstate: dp(i,j) = Min Cost to mul all mat from i..j

dp(1,N) dp(2 N) ... dp(N N)

```
int dp[N+1][N+1] = -1; invalid
int Mcm(int d[], int i, int j){ TC: O(N^2) * O(N) = O(N^3)
    if(i==j){ return 0;}          SC: O(N^2)
    if(dp[i][j]==-1){             10:40 pm
        long ans = INT_MAX;
        for(int k=i; k<j; k++){
            ans = min(ans, mcm(d,i,k) + mcm(d,k+1,j) + d[i-1]*d[k]*d[j]);
        }
        dp[i][j] = ans;
    }
    return dp[i][j];
}
```

28. Given ar[N] find length of longest strictly increasing subseq?
↳ <
↳ Ordered based on index

	0	1	2	3	4	
qn1 = ar[5] = {	9	2	4	3	10 }	ans = 3

seq1 = { 2 4 } : len 2

seq2 = { 2 3 10 } : len 3

seq3 = { 9 2 3 10 } : ~~7~~

	0	1	2	3	4	5	
qn2 = ar[6] = {	2	-1	6	8	7	9 }	ans = 4.

seq1 = { 6 7 9 } : len 3

seq2 = { 2 3 7 9 } : len 4

seq3 = { -1 6 7 9 } : len 4

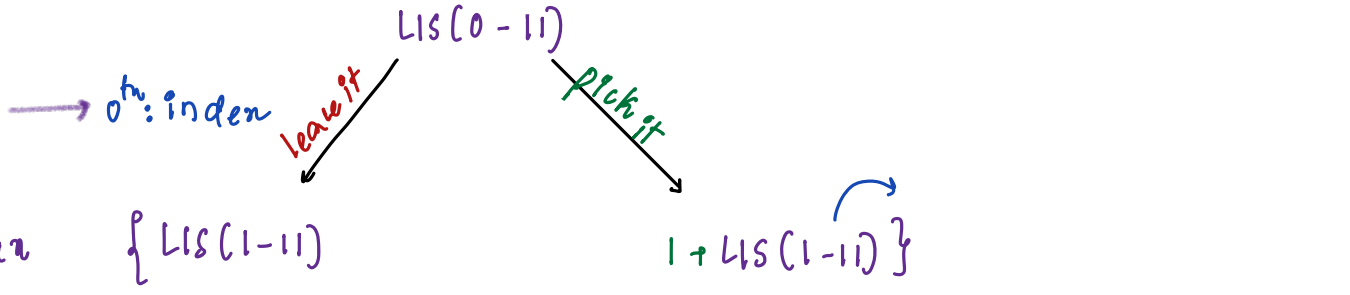
Idea1: Generate all subsequences & calculate len of longest inc subsequence.

TC = $O(2^N) * (N) = O(N * 2^N)$ SC = O(N)

Idea2: Using Recursion:

	0	1	2	3	4	5	6	7	8	9	10	11
ar[12] =	10	3	12	7	2	9	11	20	11	13	6	8

→

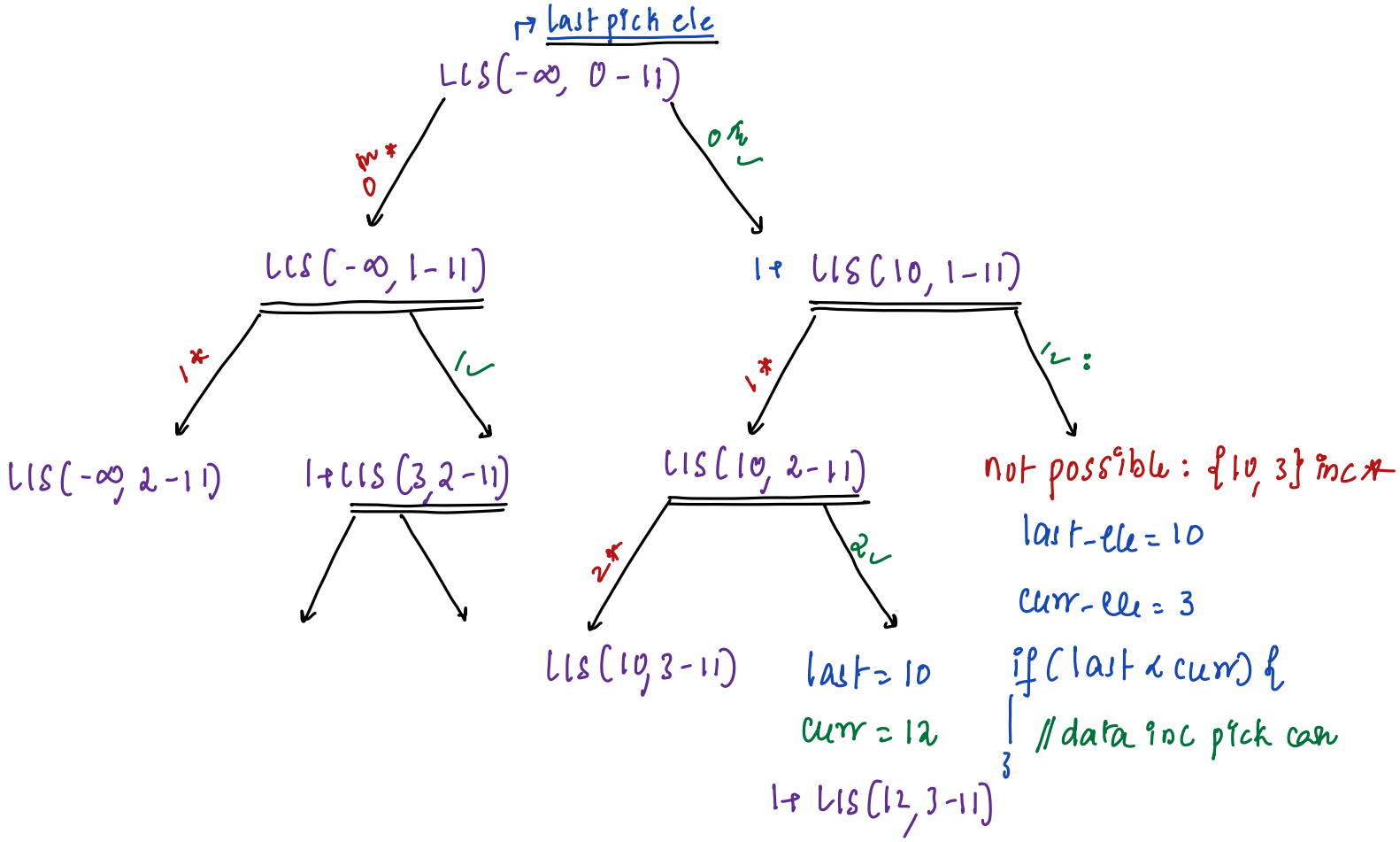


ele 10 + Say subs : { 3 7 9 11 13 } :

Invalid : Subseq is no longer increasing

Idea2: Using recursion with last picked ele

	0	1	2	3	4	5	6	7	8	9	10	11
ar[12] =	10	3	12	7	2	9	11	20	11	13	6	8



Ass: Calculate last inc subsequence from {i ... N-1}

```
int LIS(int ar[], int last, int i) {
    if (i == ar.length) {
        return 0
    }
    int not = LIS(ar, last, i+1);
    if( last < ar[i]) {
        pick = LIS(ar, ar[i], i+1) +1;
    }
    return max(not, pick)
}
```

index: ar element: dp[arelement][index] = not possible.

Note: Instead of storing last picked element, store last_picked ele.

Ass: Calculate last inc subsequence from {i ... N-1}

```
int lis(int ar[], int last, int i) {
    if (i == ar.length) {
        return 0
    }
    int not = lis(ar, last, i+1);
    if(ar[last] < ar[i]) {
        pick = lis(ar, i, i+1) + 1;
    }
    return max(not, pick)
}
```

i → inden:

last → inden: last picked inden:

dp[inden][inden] = not possible.

last picked inden, current inden

RoughSize = dp[N][N]

TC: O(N^2) * O(1) = O(N^2)

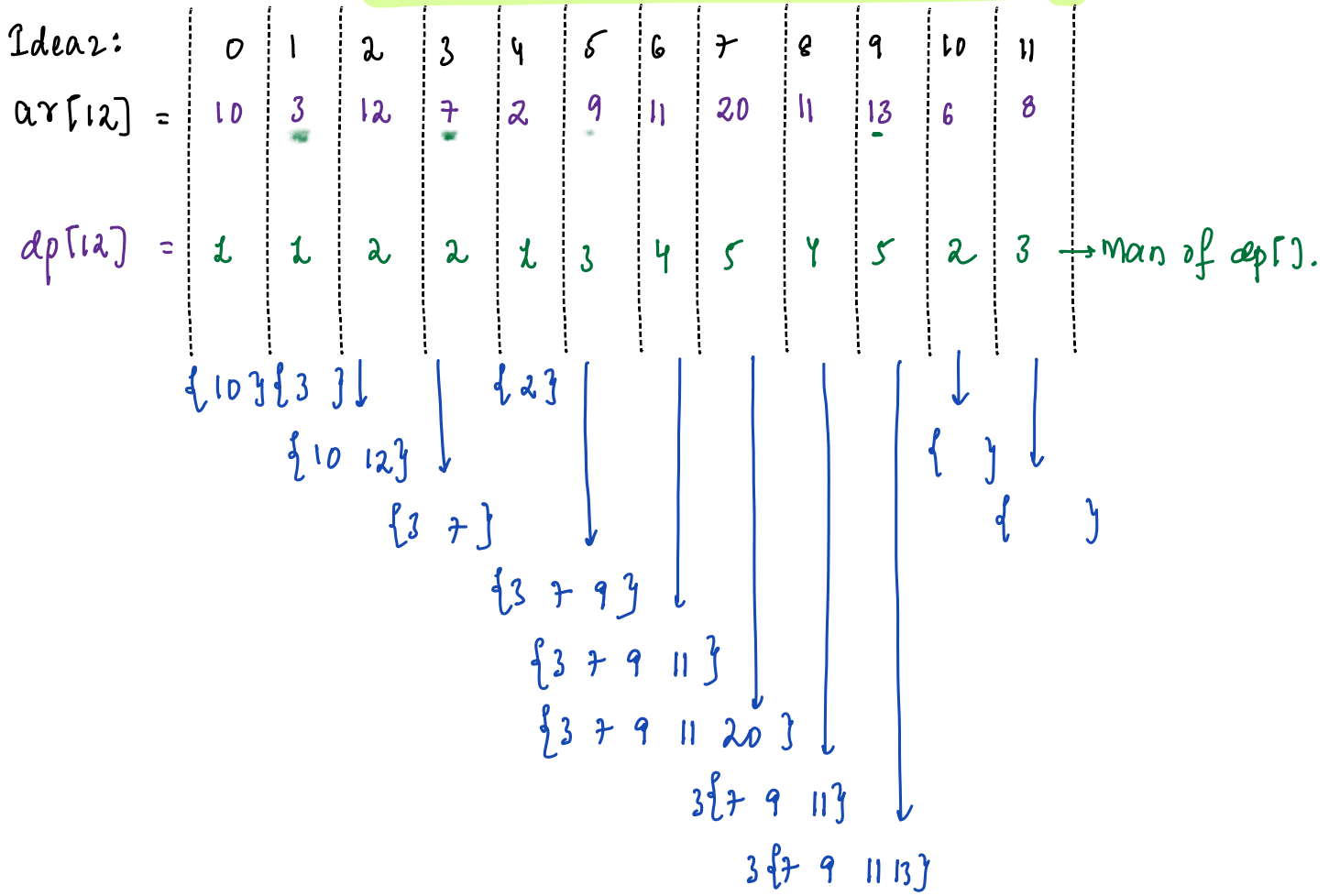
SC: O(N^2)

TODO: Write dp code.

Using dp:

dp[N], dp[i] = length of longest increasing ending at i

Note: Subsequence should contain ending at i.




```
int LIC(int ar[]){ TC: O(N^2)   SC: O(N)
    int dp[N];
    dp[0]=1;
    for(int i=1; i<N; i++){
        // dp[i] = length of longest increasing sub end at i
                { ..... ar[i] }
        int len=0;
        for(int j=0; j<i; j++){
            if(ar[j]<ar[i]){ len = max(len, dp[j]+1) }
        }
        dp[i]=len;                    // go from ar[j] → ar[i]
    }
    return max of dp[].
}
```