a) Brute Force   Least Common multiple = LCM

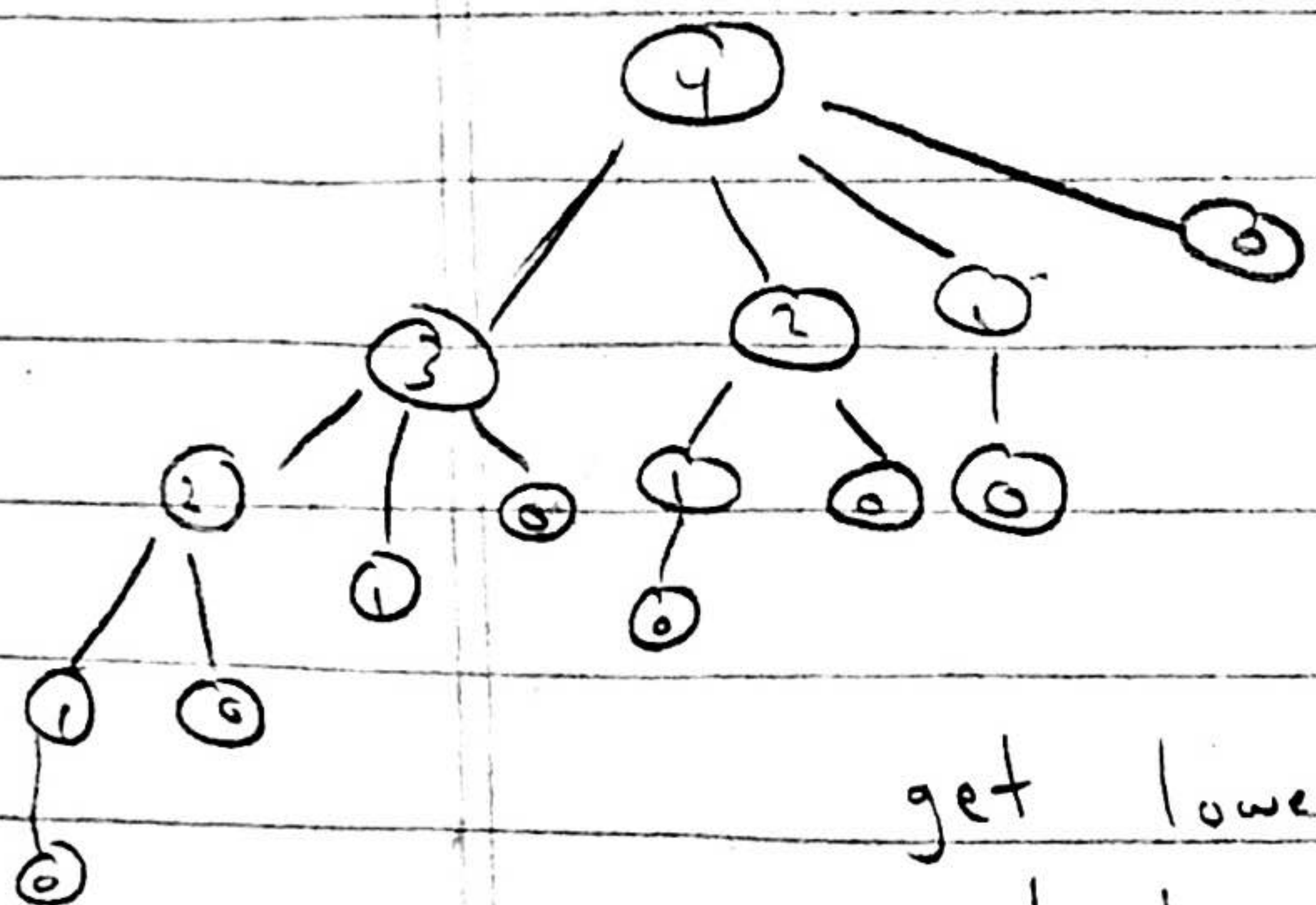
arr = [x_1, x_2, x_3, x_4, ... x_n]

LCM (arr) // get least common multiple of Elements in array

Function 1

hash Datastructure list;

```
for (int i=0; i< arr.size(); i++){
    for(int j=1; j ≤ LCM; j++){
        if( j * arr[i] == LCM){
            list.put key (i+1) value (j)
}
```



get lowest value from list datastructure

~~get lowest key~~

get key

array1 = [key, val]

length   multiple

$\frac{\text{length}}{\text{Length'.key}}$ => get remainder

cost = arr[array1[1]] // gets cost of the length, r > 0   0

# of times

x1 = length.key goes in length target

repeat function with new array length, until remainder is 0

done

add cost in data structure x1 amount of times.

result = [cost, cost, cost, ...]

$\boxed{\sum \text{result}}$

cost

b)

1
2
5

[2, 5, 7, 11] length = 5

	2	5	7	11
	1	2	3	4
0	0	0	0	0
1	2	2	2	2
2	4	5	5	5
3	6	7	7	7
4	8	10	10	11
5	10	12	12	13

→ 4 elements

n → elements in array

5 · 4 = 20 ms

5

L

(4, 1)

max Profit

max

not coming from left

4

if (i ≥ j)

A[j][i] = max(A[j-1][i]), P[j] + A[j][i-1]

else

A[j][i] - A[j-1][i]

index: 4, 1

↓ ↓

cost: 11, 2

$\boxed{\sum \text{cost} = \text{answer}}$

answer = 13

estimated guess

O(n.L)

2) Pseudocode

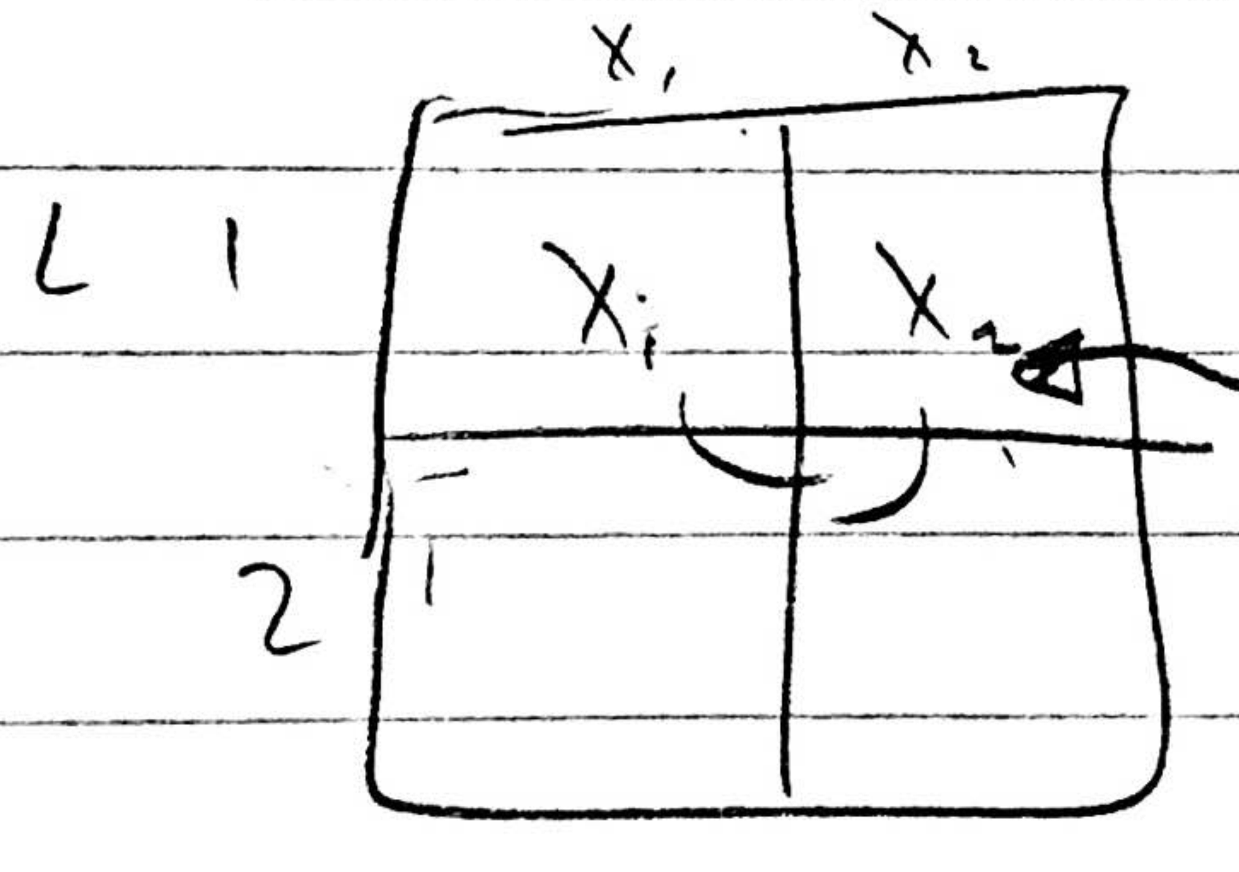
int Price[] = [x_1, x_2, x_3, x_4, X_n]
int length = 5
//the recursive call
function check(int price[], int n) {
    stores maximum profit from rod:
    A[n+1]

    for loop i to n

        max profit to ①

recursive → check(P, n-i)

        take max of last cell and
        Current cell



Compare those

~~A~~ each max values save

get max
from length
to see
max profit.

3) $T(n) = T(n-1) + T(n-2) + T(n-3) + \ldots\ldots + T(1) + 1$

substitution

$T(1) = 1$

$T(2) = T(1) + 1 = 2$

$T(3) = T(2) + T(1) + 1 = 2+1+1$

2 for loops

$\sum_{i=1}^{n} \sum_{j=1}^{n}$

master Theorem

$T(n) = n + n-1 + n-2 + \ldots$

$\frac{n(n+1)}{2} +$

$T(n-1) + n$

$a = 1$

$T(n) = (O n^2)$

$T(n) = \frac{n^2 + n}{2}$

$b = 0$

$p = 0$

$O(n^2)$

$k = 1$

$\frac{3n^2 + n}{2} +$

Case 2

$\log 1 = 1$

$n \log n = n^2$

$\boxed{O(n^2)}$

$T(n) = T(n-1) + T(n-2) + T(n-3) \ldots T(1) + 1$

no coefficient for T so 1

9) Dynamic

$A = [X_1 \quad X_2 \quad X_3]$

$I =$ 1 2 3

Length	1	2	3
0	0	0	0
1	x_1	x_1	x_1
2	$2x_1$	$\max(2x_1, x_2)$	$\max(x_1+x_1)$
3	$3x_1$	$\max(x_1+x_2, x_3)$	$\max(x_1+x_2, x_3, 3x_1)$
4	$4x_1$	$\max(3x_1, 2x_2, x_1+x_2)$	$\max(4x_1, 2x_2, x_3+x_1, x_4)$

ame
lgo
to find
L
basically

W = length $O(n \cdot w)$

$O(n^2)$

, we have the solution for all lengths

Since we have all the previous solutions for the lengths we can tackle any set of numbers.