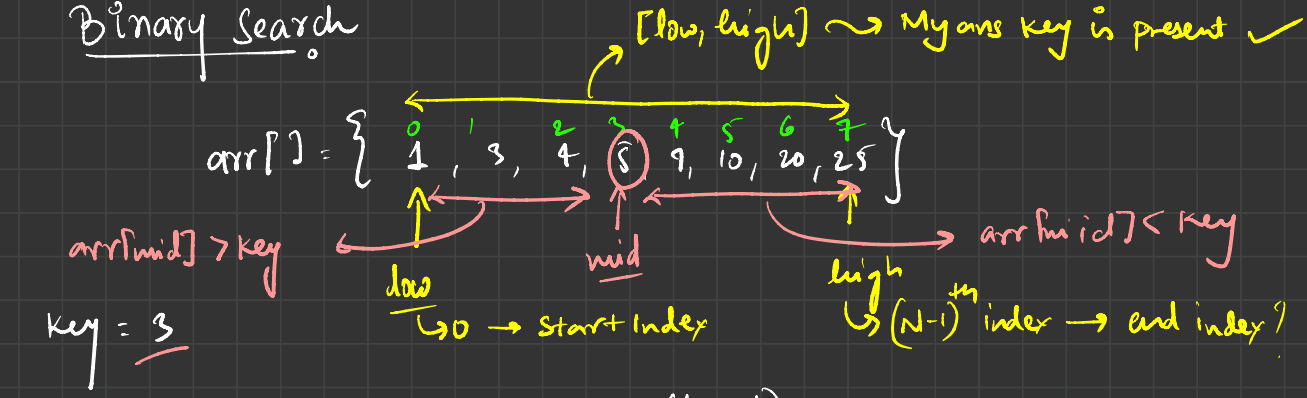
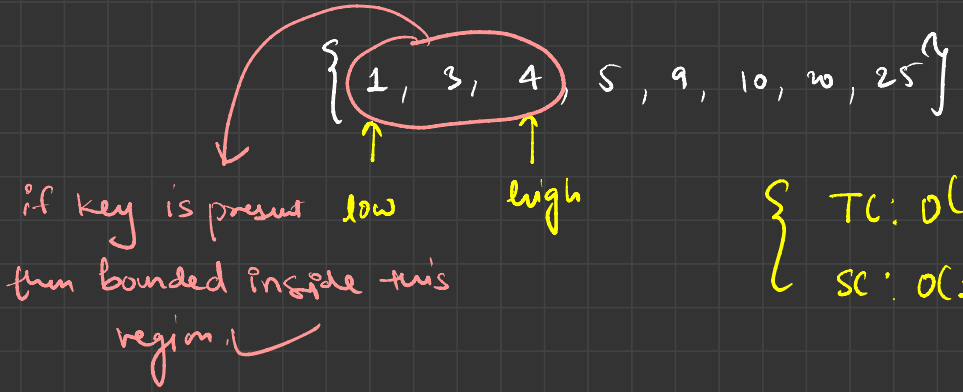
Binary Search



Binary Search → TC: O(logN)
↳ Size of the sorted Array.



{ TC: O(logN)
SC: O(1)

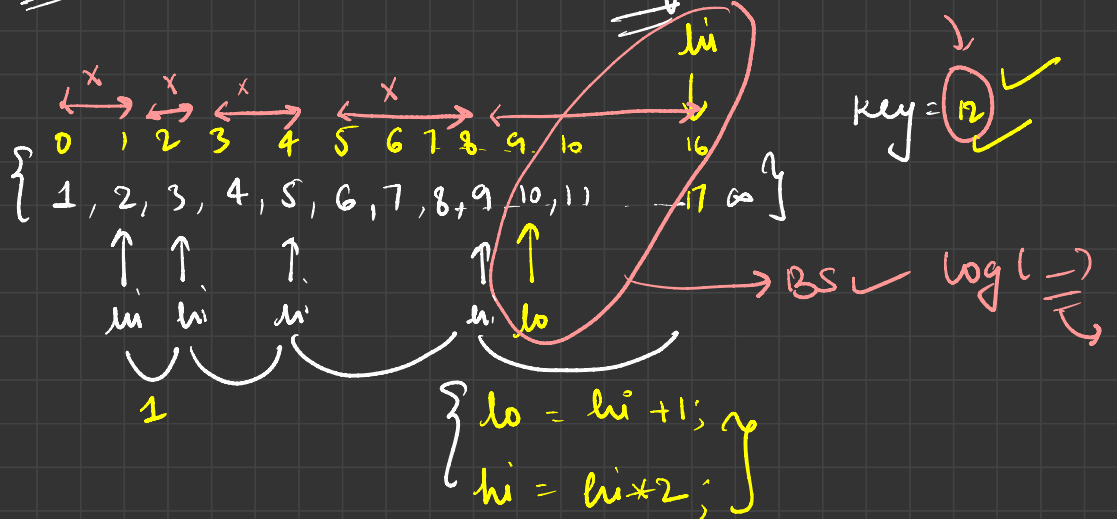
Infinite Sorted Array

arr[] : { 1, 2, 3, 4, 5, ... ∞ }

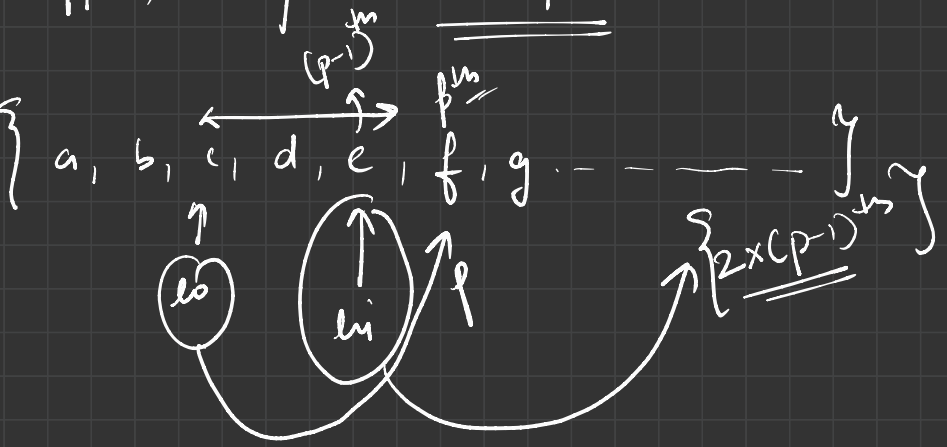
low

key = 100

high?



Suppose, Key → is at p^{th} index



Jump of high

1, 2, 4, 8, 16 ..... 2x(p-1)

Suppose high take (k+1) jumps!

$$2^K = 2 \times (p-1)$$

$$\frac{2^K}{2} = (p-1)$$

$$2^{K-1} = (p-1)$$

$$K-1 = \log_2 (p-1)$$

$$K = \log_2 (p-1) + 1$$

$O(\log p)$ → jumps

In worst case,

$lo = p^{th}$

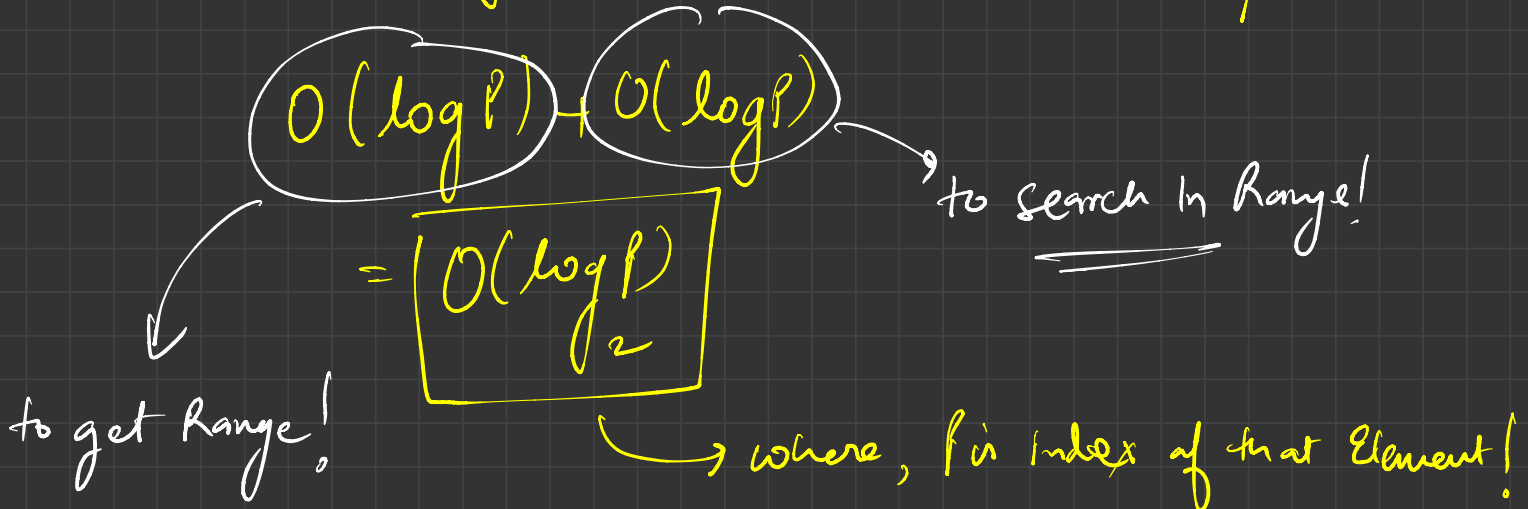
$hi = 2 \times (p-1)^{th}$

$$\text{size} = hi - lo$$
$$= 2 \times (p-1) - p$$
$$= 2p - 2 - p$$
$$= \boxed{p-2}$$

→ BS on this size!

TC : $O(\log p)$

TC: of finding Key on a infinite Sorted Array!



```
int find(int[] arr, int target)
{
    if(arr[0] > target)
        return -1;

    int lo = 0;
    int hi = 1;
    while(arr[hi] < target)
    {
        lo = hi+1;
        hi = 2*hi;
    }
    int idx = BinarySearch(arr, lo, hi, target);
    return idx;
}
```

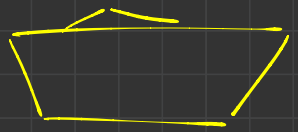
TC: O(log P)

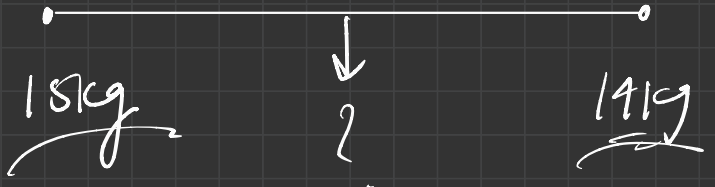
P is index of key in infinite array!

Capacity to ship packages within B days

0	1	2	3	4	5	6	7	8	9
A[] = [	1,	2,	3,	4,	5,	6,	7,	8,	9,	10]

B = 5





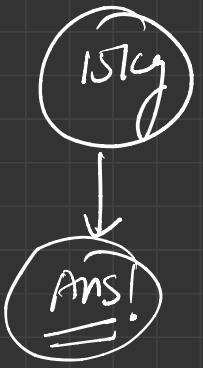
5 days

pAns = ~~32kg~~ ~~20kg~~ ~~17kg~~

0 1 2 3 4 5 6 7 8 9

A[] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

day1 day2 day3 day4 day5



maxCap = 15kg

5 days

Minimum Limit of Balls in A Bag

arr[] = { 2, 4, 8, 2 } maxOpt = 4

{ 3, 1 }

(2, 2) (1, 3) (3, 1)

{ 2, 3, 1, 8, 2 } 3

(1, 7), (2, 6), (3, 5), (4, 4)

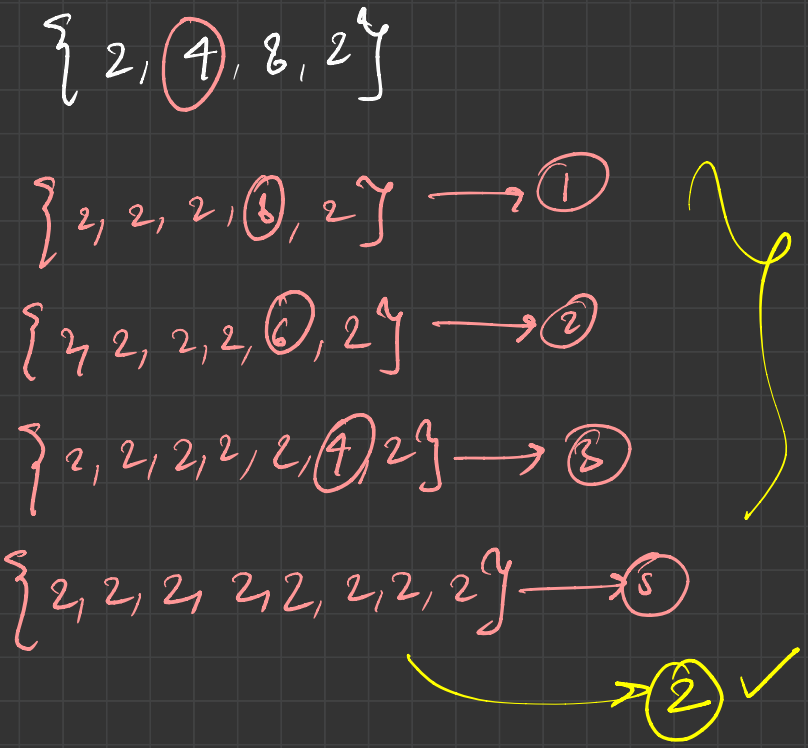
(5, 3) (6, 2) (7, 1)

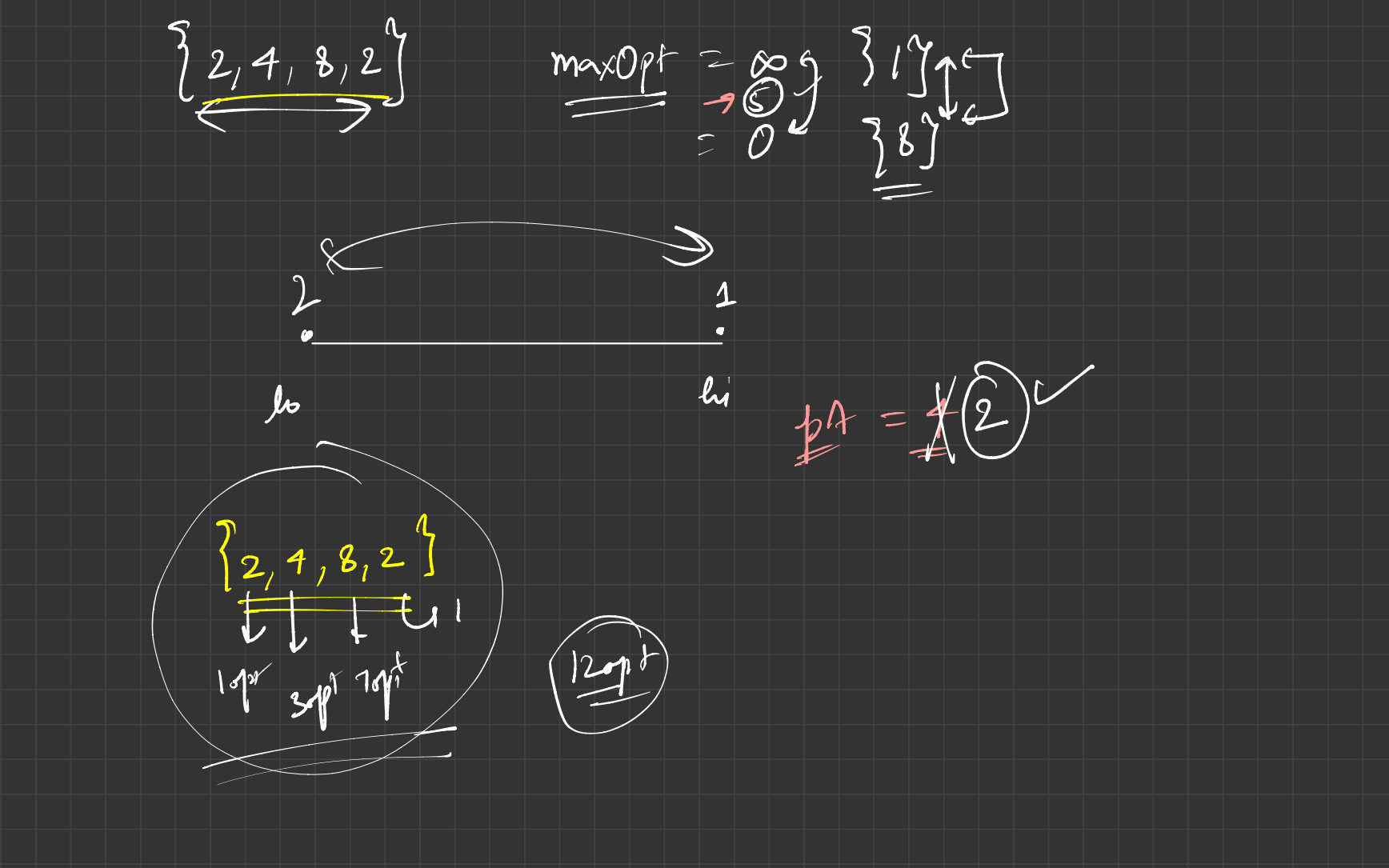
{ 2, 3, 1, 4, 4, 2 } 2

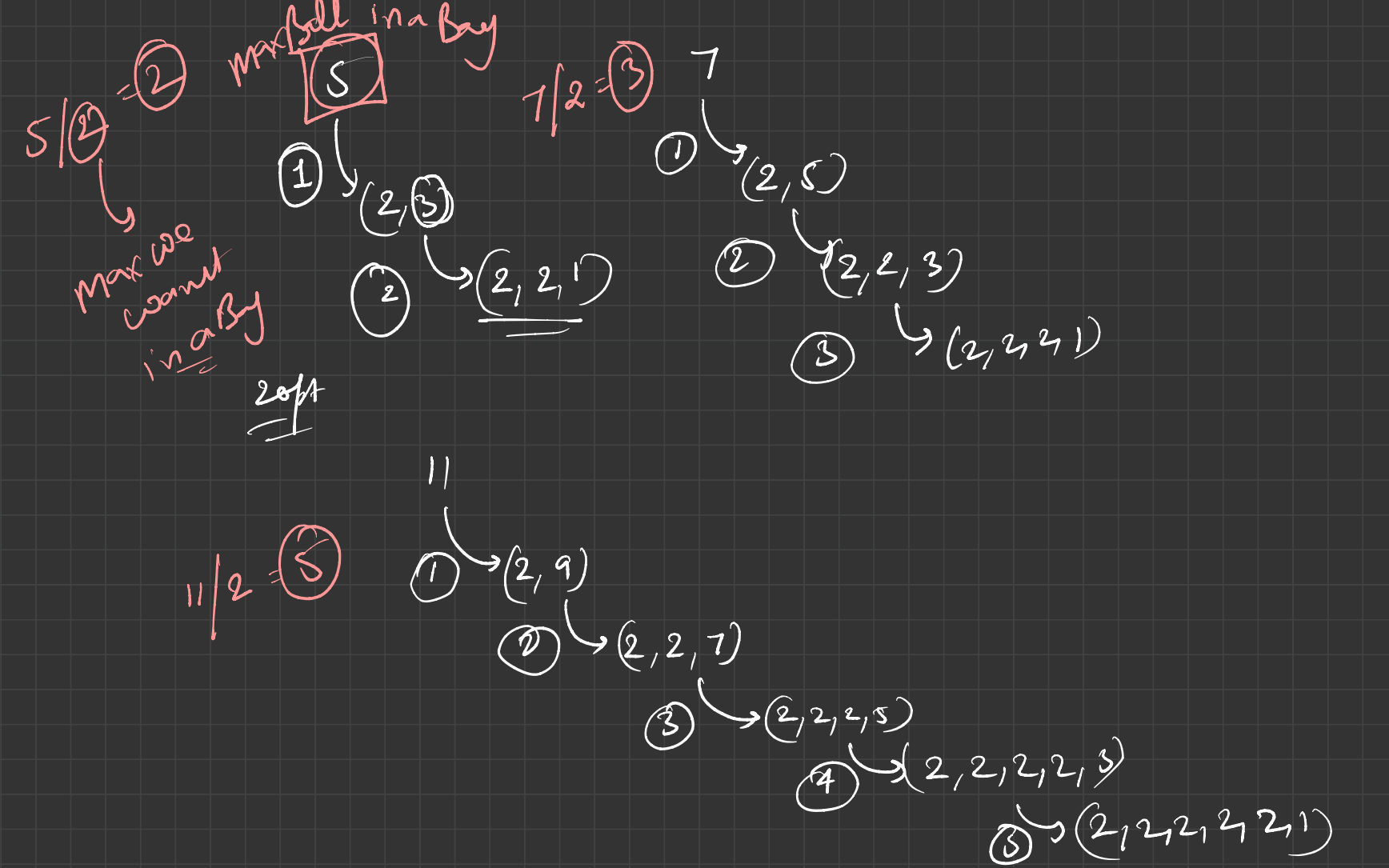
{ 2, 3, 1, 3, 1, 4, 2 } 1

→ { 2, 3, 1, 3, 1, 2, 2, 2 } 0

3


{2, 4, 8, 2}
{2, 2, 2, 8, 2} → 1
{2, 2, 2, 2, 6, 2} → 2
{2, 2, 2, 2, 2, 4, 2} → 3
{2, 2, 2, 2, 2, 2, 2, 2} → 5
2

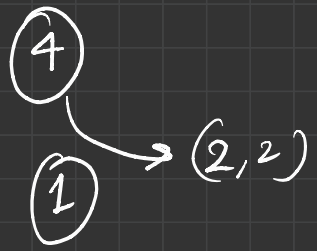

{2, 4, 8, 2}
maxOpt = ∞
5
0
1
8
2
1
lo
hi
ans = 2
{2, 4, 8, 2}
1opt
3opt
7opt
12opt

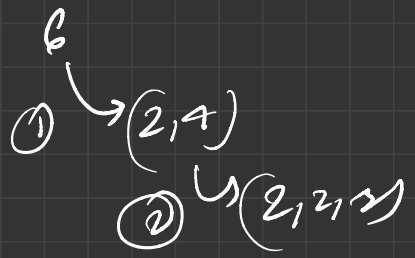

5/2 = 2
Max we could in a Bag
Max Ball in a Bag
5
1
(2,3)
2
(2,2,1)
7/2 = 3
7
1
(2,5)
2
(2,2,3)
3
(2,2,2,1)
20pt
11/2 = 5
11
1
(2,9)
2
(2,2,7)
3
(2,2,2,5)
4
(2,2,2,2,3)
5
(2,2,2,2,2,1)

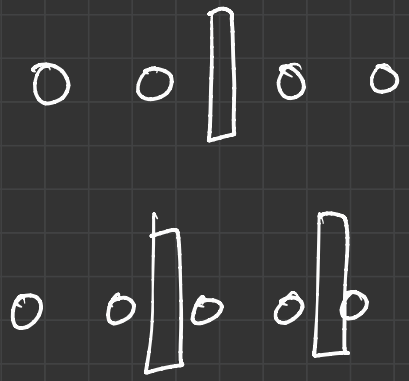
Num of opt = $\frac{\text{Max}^m \text{ in A Bag}}{\text{Max}^m \text{ we want}}$

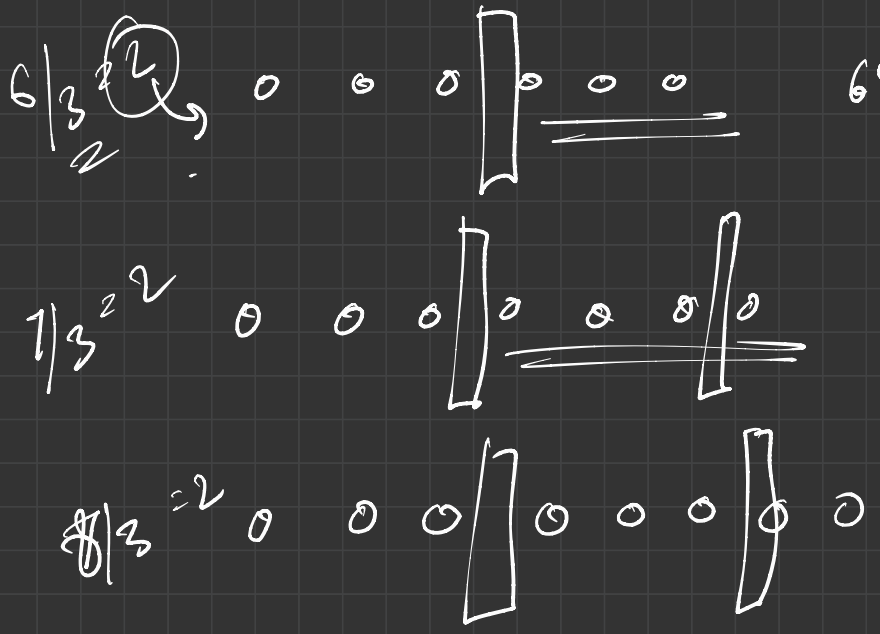
$4/2 = 2$

$6/2 = 3$








6/3 = 2
6%3 = 20
decrease
1 separator

```java
class Solution {
    static boolean isPossible(int n, int[] arr, int limit, int maxOpt) {
        int currOpt = 0;
        for (int i = 0; i < n; i++) {
            int currBalls = arr[i];
            int divider = currBalls / limit;
            if (currBalls % limit == 0) {
                divider--;
            }
            currOpt += divider;
        }

        return currOpt <= maxOpt;
    }

    public static int solve(int n, int m, int arr[]) {
        // Write your code here

        // if infinite operations, you can divide all balls to sepreate bags, hence max
        // in a bag will be 1
        int lo = 1;

        // if you have zero opt, then max in a bag in max in the array of bags
        int hi = 0;
        for (int balls : arr) {
            hi = Math.max(balls, hi);
        }

        int potetialAns = -1;
        while (lo <= hi) {
            // mid -> max balls you can in a bag after division
            int mid = (lo + hi) / 2;

            if (isPossible(n, arr, mid, m) == true) {
                potetialAns = mid;
                hi = mid - 1;
            } else {
                lo = mid + 1;
            }
        }

        return potetialAns;
    }
}
```

arr[] = {2, 4, 8, 2} m = 5

lo = 1 hi = 8 mid = 1

pAns = 2

8/1 = 8

2/1 = 2 (1)

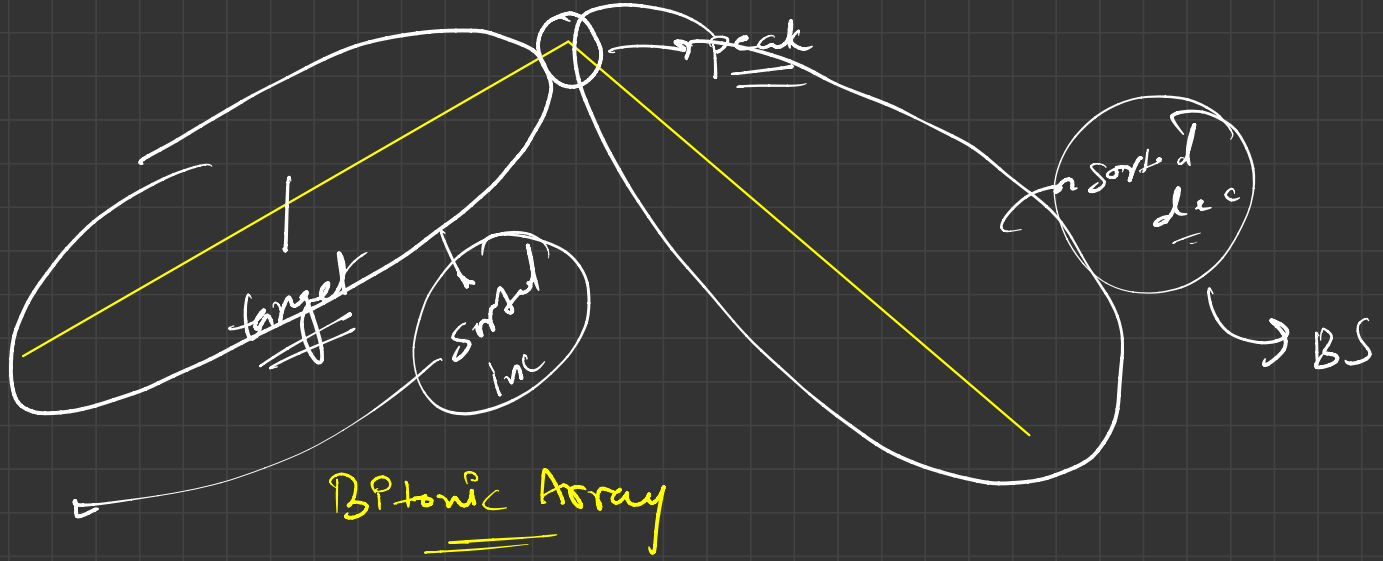
2, 1 = 20

4/1 = 4 3 2 1 = 4

40/1 = 40

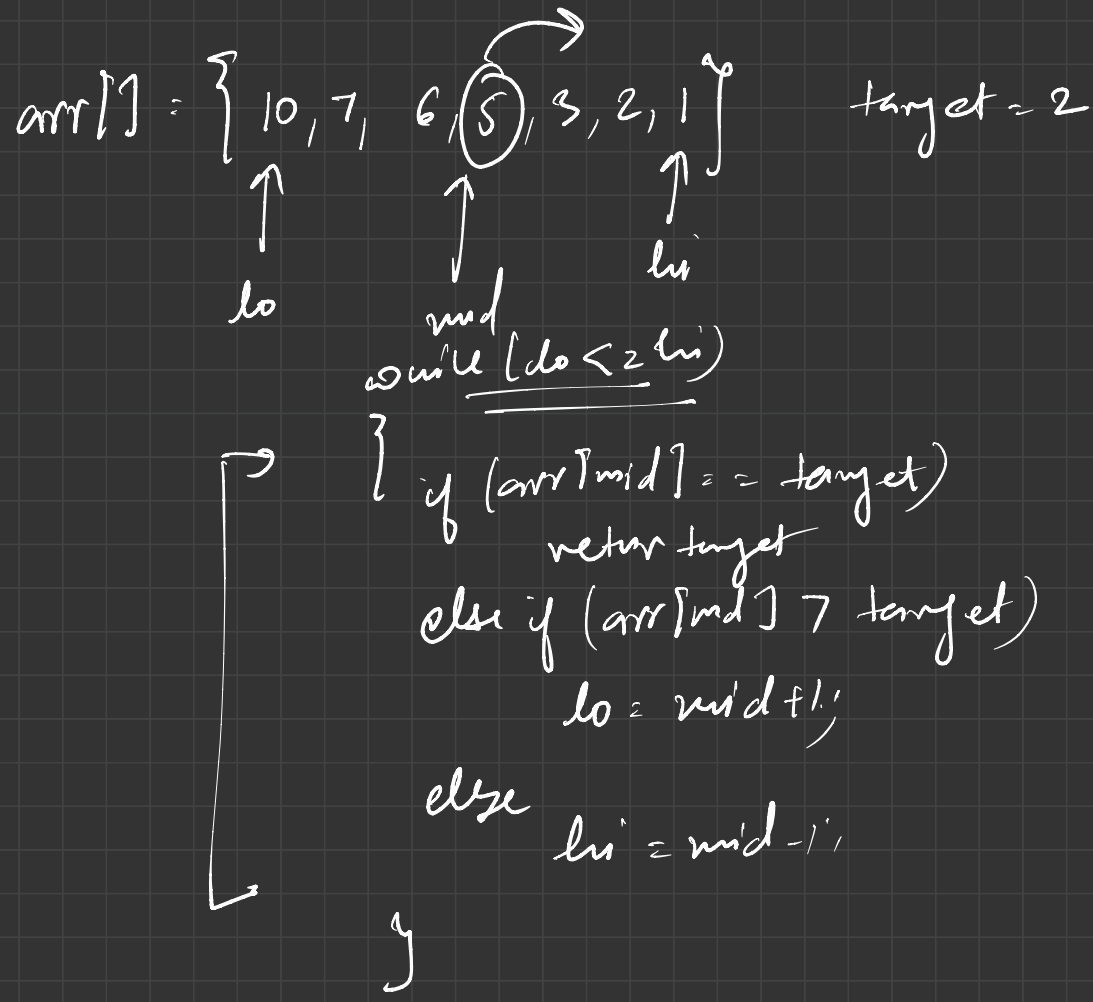
currOpt = 20 (12)

Binary Search is Completed!



Bitonic Array

BS


arr[] = { 10, 7, 6, 5, 3, 2, 1 }
target = 2
lo
mid
hi
while (lo <= hi)
{
if (arr[mid] == target)
return target
else if (arr[mid] > target)
lo = mid + 1;
else
hi = mid - 1;
}