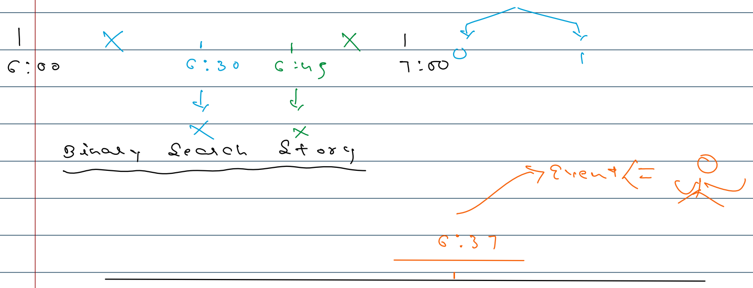
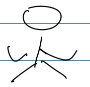
26 July 2025 10:07

Searching algorithms
- → Linear (Sequential)
- → Binary search

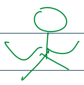


It is an algorithm for finding an element in a sort array by dividing the search range in half.



0	1	2	3	4	5	6	7	8	9	10	11	12	13
2	3	4	5	7	9	11	13	16	18	20	22	24	26

Buffet
↓
Spice level



⇒ Paneer Tikka → (13)

low, low, mid, high, high

0	1	2	3	4	5	6	7	8	9	10	11	12	13
2	3	4	5	7	9	11	13	16	18	20	22	24	26

$mid = \frac{(low + high)}{2}$

low → 0 mid + 1 → 6 + 1 → 7

high → 13 mid - 1 → 10 - 1 → 9 → mid - 1 ⇒ 7

key → (13)

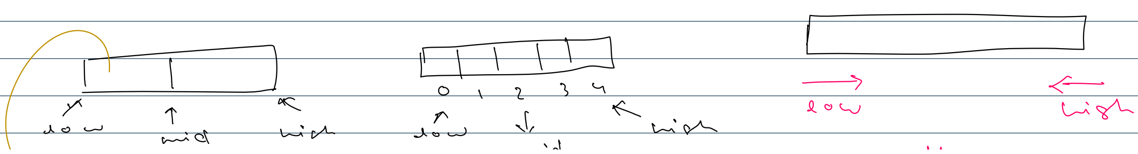
$m = \frac{(0+13)}{2} = 6$ arr[mid] < key   11 < 13

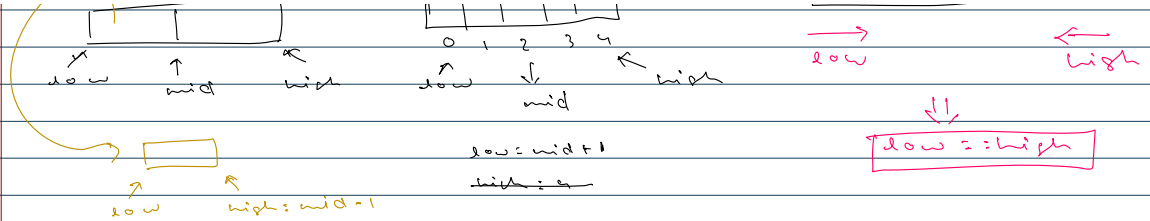
low = 7, high = 13 $m = \frac{(7+13)}{2}$, m = 10   arr[mid] > key   20 > 13

low = 7, high = 9 $m = \frac{(7+9)}{2}$, m = 8   arr[mid] > key   16 > 13

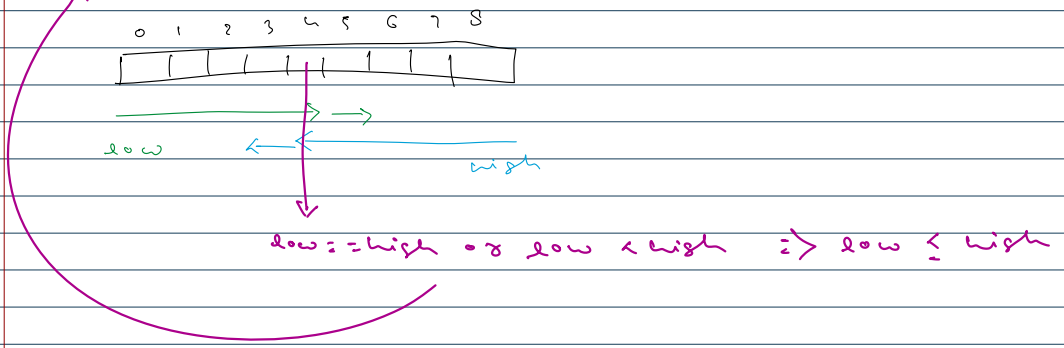
low = 7, high = 7 $m = \frac{(7+7)}{2}$, m = 7   arr[mid] = key   13 = 13

```
if (arr[mid] == key)
    return mid;
else if (arr[mid] > key)    } → 1st array
    high = mid - 1
else if (arr[mid] < key)    } → 2nd array
    low = mid + 1;
```





Exit condition => while (low <= high)



low = high or low < high => low <= high

```
// "static void main" must be defined in a public class.
public class Main {
    public static void main(String[] args) {

        int[] arr = {2, 3, 4, 5, 6, 7, 8, 11, 13, 16, 17};
        int item = 13;
        System.out.println(binarySearch(arr, item));
    }

    public static int binarySearch(int[] arr, int item){
        int n = arr.length;
        int low = 0;
        int high = n-1;

        while(low<=high){

            int mid = (low+high)/2;

            if(arr[mid] == item){
                return mid;
            }
            else if(arr[mid] > item){
                high = mid - 1;
            }
            else if(arr[mid] < item){
                low = mid + 1;
            }
        }

        return -1;
    }
}
```

How to identify when to use Binary Search

1. array sorted => ascending

1. array sorted ⇒ asc, desc.
2. data is static
3. lesser insertion/deletion

Find the floor of the kth root

$x^3 \le 149$ ⇒ find largest value of x

$5^3 = 125 \le 149$ ✓ ⇒ x = 5

$6^3 = 216 \le 149$ ✗

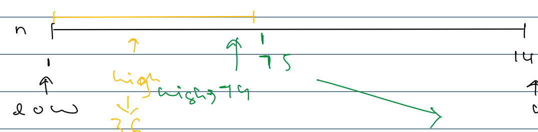
$x^k \le n$

k → power
n → number } → Input
find x ? } → output

```java
public class Main {
    public static void main(String[] args) {

        int n = 149;
        int k = 3;
        System.out.println(kthRootLinear(n, k));
    }

    public static int kthRootLinear(int n, int k){
        int ans = 0;

        for(int i=1; i<=n; i++){
            double power = Math.pow(i, k);

            if(power <= n){
                ans = i;
            }
            else{
                break;
            }
        }
        return ans;

    }
}
```

Binary search solution

$x^3 \le 149$ ⇒ x → ?



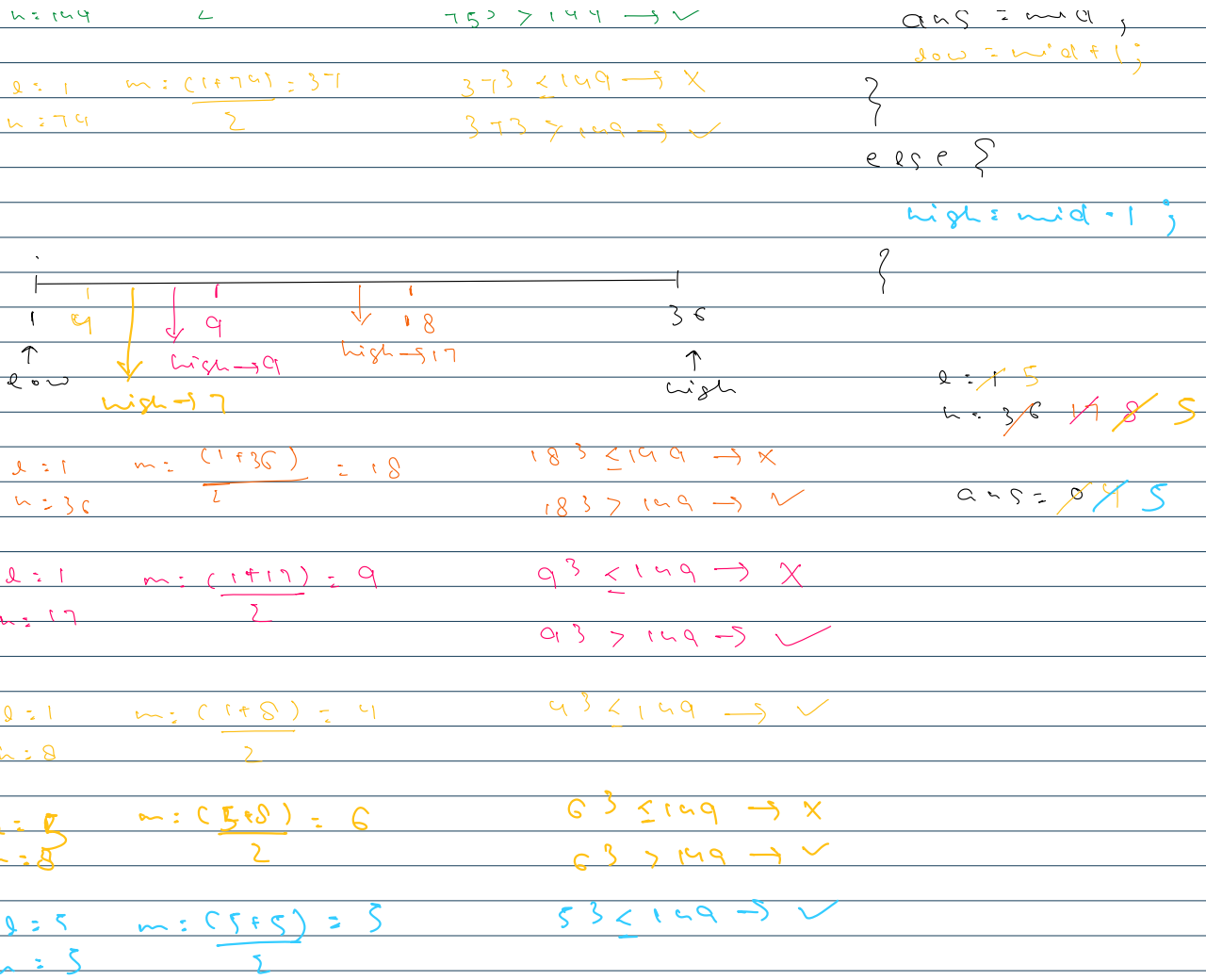
low : 1
high : ~~149~~ ~~74~~ 36

l = 1, h = 149: m = (1+149)/2 = 75 — $75^3 \le 149$ → ✗ ; $75^3 > 149$ → ✓

if (mid^k <= n) {
ans = mid;
low = mid + 1;

l = 1, m = (1+74)/2 = 37 — $37^3 \le 149$ → ✗

?




```
public class Main {
    public static void main(String[] args) {
        int n = 149;
        int k = 3;
        System.out.println(kthRootLinear(n, k));
        System.out.println(kthRootBinary(n, k));

    }

    public static int kthRootLinear(int n, int k){
        int ans = 0;

        for(int i=1; i<=n; i++){
            double power = Math.pow(i, k);

            if(power <= n){
                ans = i;
            }
            else{
                break;
            }
        }
        return ans;

    }

    public static int kthRootBinary(int n, int k){
        int ans = 0;
        int low = 1;
```

```
        int high = n;

        while(low<=high){
            int mid = (low+high)/2;
            if(Math.pow(mid, k) <= n){
                ans = mid;
                low = mid + 1;
            }
            else{
                high = mid - 1;
            }
        }
        return ans;

    }
}
```

278. First Bad Version

Solved

Easy | Topics | Companies

You are a product manager and currently leading a team to develop a new product. Unfortunately, the latest version of your product fails the quality check. Since each version is developed based on the previous version, all the versions after a bad version are also bad.

Suppose you have `n` versions `[1, 2, ..., n]` and you want to find out the first bad one, which causes all the following ones to be bad.

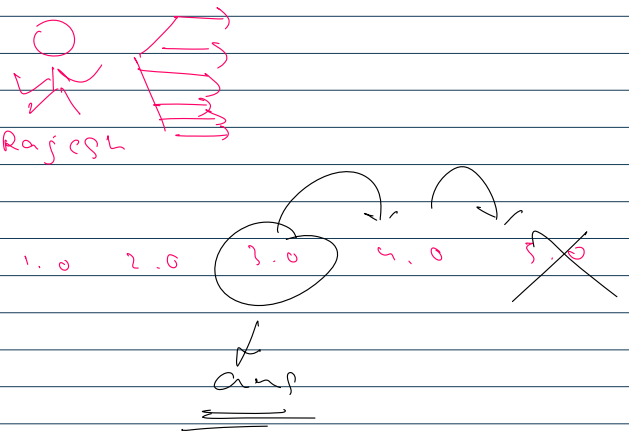
You are given an API `bool isBadVersion(version)` which returns whether `version` is bad. Implement a function to find the first bad version. You should minimize the number of calls to the API.

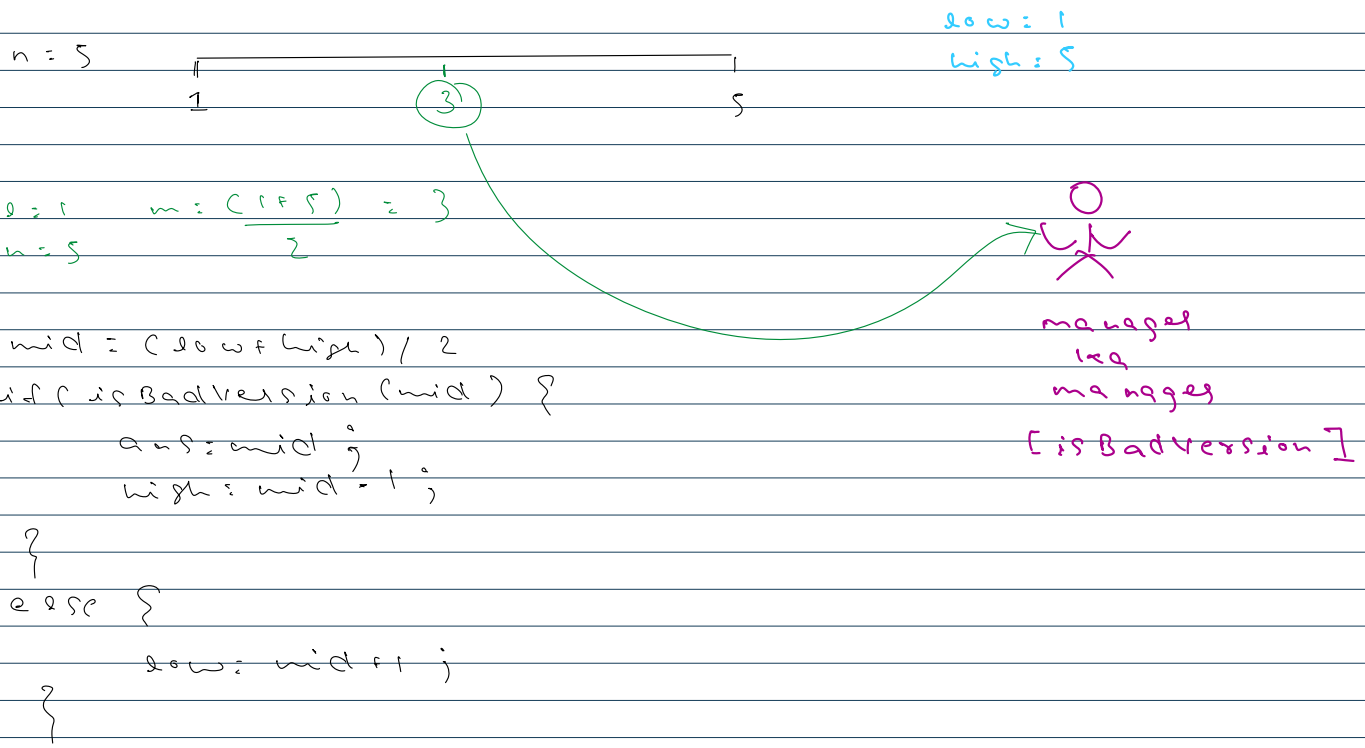
Example 1:

```
Input: n = 5, bad = 4
Output: 4
Explanation:
call isBadVersion(3) -> false
call isBadVersion(5) -> true
call isBadVersion(4) -> true
Then 4 is the first bad version.
```

Example 2:

```
Input: n = 1, bad = 1
Output: 1
```





Constraints

- $1 <= bad <= n <= 2^{31} - 1$

max value of n = $2^{31} - 1$

l : $2^{31} - 6$

h : $2^{31} - 1$

mid : $\frac{(low + high)}{2} = \frac{2^{31} - 6 + 2^{31} - 1}{2}$

$= \frac{2^{31} + 2^{31} - 7}{2}$

$= \frac{2^{32} - 7}{2}$ → out of range

$2^2 + 2^2 = 2^3$

$2 \cdot (2^2) = 2^3$

mid : $low + \frac{(high - low)}{2}$

$\frac{2 \cdot low + high - low}{2} = \frac{low + high}{2}$

```
/* The isBadVersion API is defined in the parent class VersionControl.
      boolean isBadVersion(int version); */
public class Solution extends VersionControl {
    public int firstBadVersion(int n) {
        int low = 1;
        int high = n;

        int ans = -1;
        while(low<=high){
            // int mid = (low+high)/2;
            int mid = low + (high-low)/2;
            // int mid = low/2 + high/2; // -->X
            if(isBadVersion(mid)){
                ans = mid;
                high = mid-1;
            }
            else{
                low = mid + 1;
            }
        }
        return ans;
    }
}
```