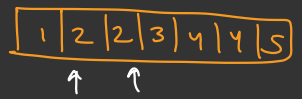
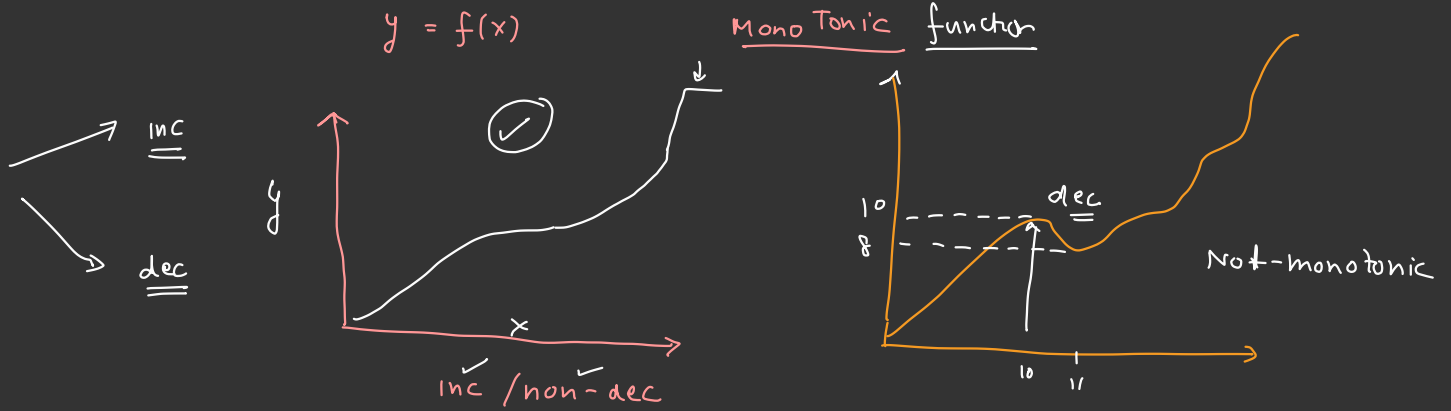
Binary Search

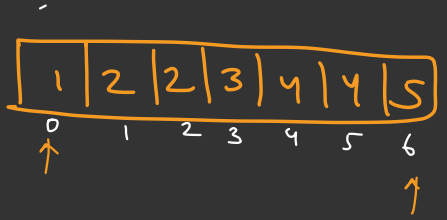
(A) ✓ Binary Search on Sorted Array



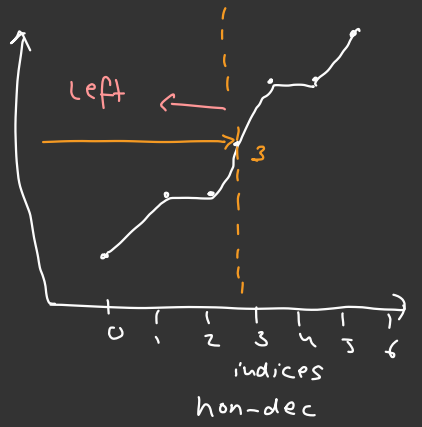
→ search
→ first occ
→ last occ

Today → Monotonic Search Spaces





arr val



Target = 2

Square Root (zomato)

$N = 100 \quad \sqrt{100} = 10$

$N = 50 \quad \sqrt{50} = 7$

$N = 150 \quad \sqrt{150} = 12$

Math.sqrt(100)

Brute force : (Integer Part)

$N = 50$

$i = 0$

while $i^2 <= N$ {

$i++$

}

$ans = i - 1$

print(ans)

Stop $i^2 > N$

$i > \sqrt{N}$

Time → $O(\sqrt{N})$

$N = 10^4$

$\sqrt{N} = 100$

$\Rightarrow O(\log N) < O(\sqrt{N}) < O(N)$

0 1 2 3 4

$0^2 <= 50$

$1^2 <= 50$

$2^2 <= 50$

$3^2 <= 50$

$4^2 <= 50$

$5^2 <= 50$

$6^2 <= 50$

$7^2 <= 50$

$i = 8 \rightarrow 8^2 <= 50$

Linear Search

Binay Search for Square Root

N = 50

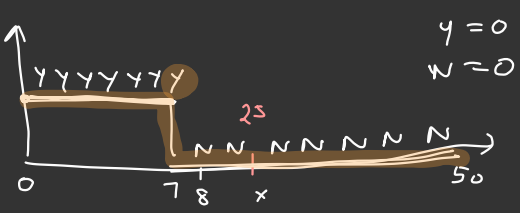
Search Space (0 — N)

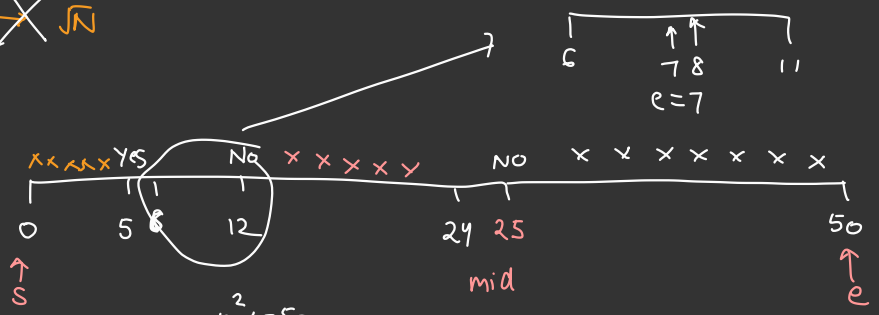
T ✗ √N

x^2 $mid^2 <= N$

f(x)

Yes ↓ No





N → N/2 → N/4 → ... → 1

O(Log N)

$12^2 <= 50$ e = m-1

$25^2 <= 50$

s	e	mid		
s=0	e=50	mid=25	NO	e=m-1
s=0	e=24	m=12	NO	e=m-1
s=0	e=11	m=5	Yes	s=m+1
s=6	e=11	m=8	NO	e=m-1
s=6	e=7	m=6	Yes	s=m+1
s=7	e=7	m=7	Yes	s=m+1
s=8	e=7	Stop	NO	

$5^2 <= 50$

$8^2 <= 50$ ← N

$6^2 <= 50$ ←

$7^2 <= 50$

$mid^2 <= N$

ans

5

6

7

last time you got a Yes will be the final ans

Return ans;

$(NO)^2 <= N$

↑ maximise the no by go Right

$s = mid + 1$

$N = 50$

Code

```
int binarySearch (int n) {
    s=0
    e=N
    while(s<=e){
        mid = (s+e)/2,
        if(mid^2 > N){
            e = mid-1;
        }
        elseif(mid^2 == ans){
            ans = mid;
            s = mid+1
        }
    }
    return ans,
}
```

O(Log N)

O(1) Space

```
elseif(mid^2 == ans){
    return mid,
}
```

(optional)

Decimal Part

N = 3 P = 3

P ≤ 8

$\sqrt{10} = 3.162$

Binary Search $O(\log N)$

Linear Search

10 iters $O(1)$

$p \times 10$

0–9 0–9 0–9

3.1 6 2

$O(\log N + P)$

for loop

$3.0^2 <= 10$

while

3.0 → 0.1 → 3.1

$3.1^2 <= 10$

3.1 → 0.1 → 3.2

$3.2^2 <= 10$

3.1 0 → 0.01 → 3.11 → 3.12 → 3.13 → 3.14 → 3.15 → 3.16 → 3.17

$3.16^2 <= 10$

$3.17^2 <= 10$ (No)

3.160 → 3.161 → 3.162 → 3.163

$3.162^2 <= 10$

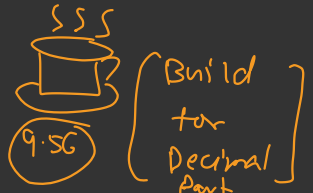
$3.163^2 <= 10$

$3.10^2 <= 10$

$3.11^2 <= 10$

ans = ans + inc

ans ≤



float SquareRoot (int N, int P) { → No of places after decimal

N = 10

log N

float ans = binSearch for Sq Root (N),

ans = 3.0000

float inc = 0.1

for(i=1 ; i<=P, i++){

while(ans * ans <= N){

ans = ans + inc

}

ans = ans - inc,

inc = inc/10; ←

}

return ans,

}

places

one place

10 times

ans = 3.1 → 0.1
= 3.2
= 3.3 → 0.1

P

$3.0^2 <= 10$

$3.1^2 <= 10$

→ $3.2^2 <= 10$ ←

inc = 0.01

ans = 3.1000 0.001

$3.10^2 <= 10$ 0.01

$3.11^2 <= 10$

$3.12^2 <= 10$

0.1
0.01
0.001

→ 3.1619997

String format("%.3f", ans)

→ 3.1620000 → 3162

Aggressive Cows (Code)

stalls = [1, 2, 8, 4, 9]

c = 3

2 cows

[1, 2, 4, 8, 9]

① Sort (stalls)

Binary Search

s = 0

e = stalls[n-1] - stalls[0] , ans = 0

while (s <= e) {

mid = (s+e)/2,

if (canPlace cows with mid as minSep) {

ans = mid

s = mid + 1

} else {

e = mid - 1

}

}

log(e-s).N

```
        3
    else {
            e = mid - 1
        3
}

print(ans);
```

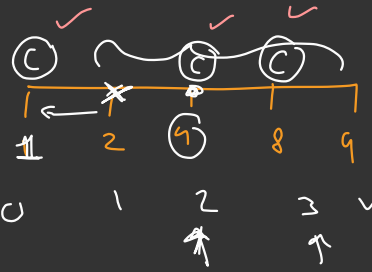
```
                                  //  (3)
boolean canPlace ( stalls[], sep, [C], N ) {
                                   mid  cows  stalls
                                    ↓    ↓     ↓
    lastCow = stalls[0]
    cnt = 1

    for( i=1; i<=N-1, i++ ){
        if ( stalls[i] - lastCow >= sep){
            cnt ++;
```

(2)

(N) → stalls

O(N)



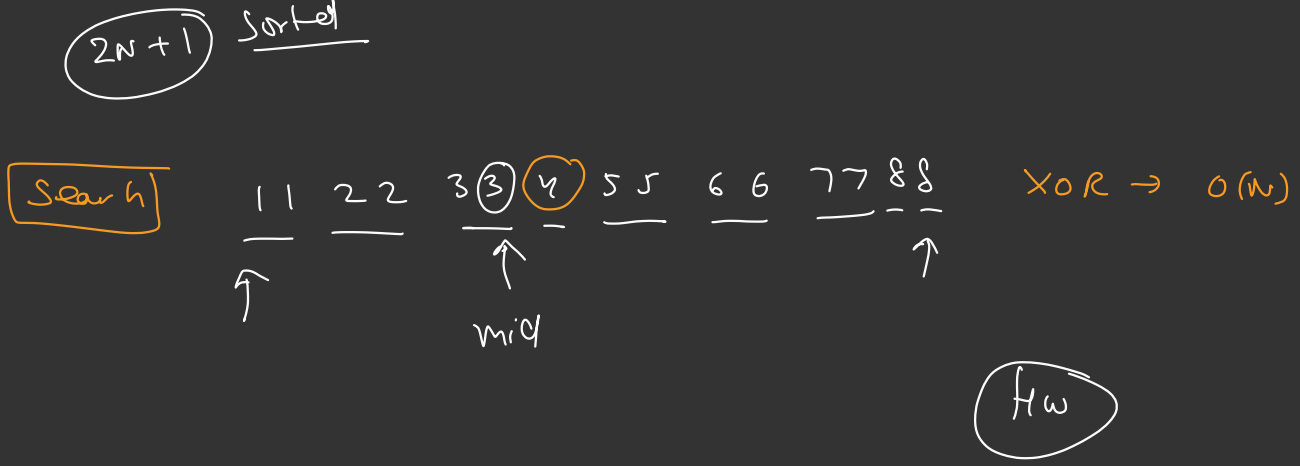
```
        last cow = stalls[i];
        if(cnt == c) { return true; }
    }
    return false;
}
```

2 - 1 >= 3

4 - 1 >= 3

8 - 4 >= 3

Place 3 cows with min sep = sep



Next Class

Find unique element in an array containing 2N + 1 elements, where every element repeats twice. Array is sorted