Intermediate Batch (~2 month)

Introduction to Problem Solving → how to understand 'time complexity'?

↳ Time Limit Exceeded

Warm up Puzzle → Come up with ideas!

10-35

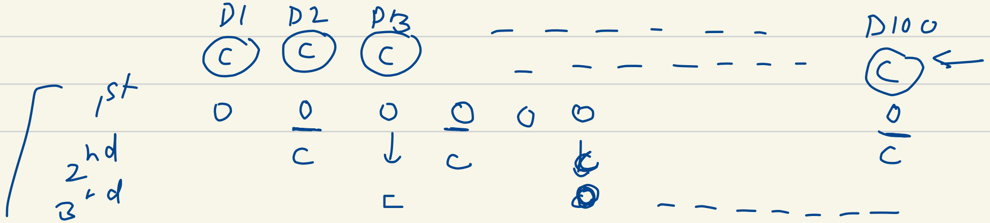
- There is a circular jail with 100 cells numbered from 1 to 100.
- Each cell has an inmate and the door is initially locked. (closed)
- One night the jailer gets drunk and he goes around the jail in circles.
- In 'ith round' he goes to every door which is multiple of I and changes the state of the door. If the door is open, he will close it and vice versa.
- He makes a total of 100 rounds, how many prisoners found their door open after 100 rounds ?

i for 1,2,3---100

every Round

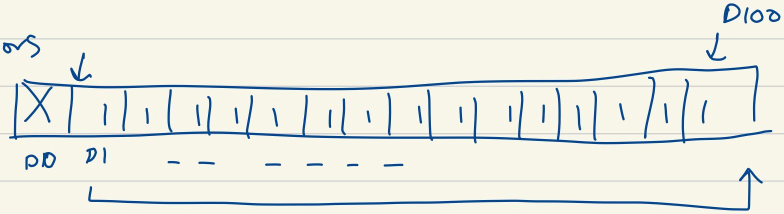
5 Mins

Closed ⇄ open



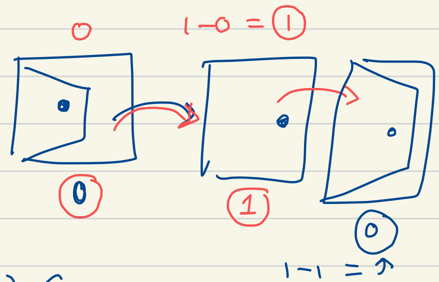
100 Rounds

N ⇒ Rounds = Doors



→ 0 – open

→ 1 – closed



Round

1 → 1, 2, 3, 4 – – 100

2 → 2, 4, 6, 8, – – – – 100

3 → 3, 6, 9, – – – 99

4 → 4, 8, 12, 16 – – – 100

5 → 5, 10 – – – 100

6 → 6, 12, – – – – 96

```
int  doors [101] ;      ARRAY

set (doors , 1, 100);

for ( round = 1 , round <= 100 ; round ++) {
   indoor
      for( i = round ; i <= 100 ; i = i + round){
instruction →   doors[i] = 1 - doors[i]; // Flip /Toggle
                                Flip the state of door
            3
     3
```

99 + 3

102

Loop → to count ~~the~~ dors

N times

```
for (i=1 ———— 100){
        count the no of open doors , dors[i] == 0
}
```

round = 3

i=3 ; 6 9 12 15 ---- 90 93 96 99

0 1 1 0

1 0 0 1

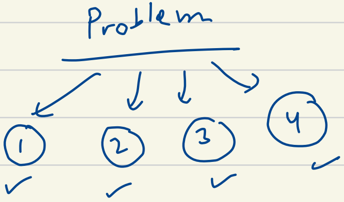
round = 4

⋮

CPU → we don't know how much time it will take?

↑

time ↑ ⇒ optimise ↑



→ code Everything
→ then decide ? > Time waste ×

Analyse → Decide → code

↑
without running code ——→ How much of work CPU is doing.

Time $\propto$ work $\propto$ iterations.

Ex-1

```
for (int i=1; i<=N , i++){
    // work
}
```

Time $\propto$ N

Ex-2

```
for (i=1 , i<=N;  i++){
    [ for (j=1;  j<=N; j++){
    [       → // work ←
    [ }
}
```

i=1 → N times

i=2 → N times

⋮

inner loop

i=1	N
i=2	N
i=3	N
⋮	
i=N	N

work

$= N+N+\ldots\ldots$

$\Rightarrow N * N$

Time $\propto N^2$

$N \rightarrow 10N$

$T \rightarrow (10N)^2$

$\rightarrow 100\,N^2$

$\rightarrow 100\,T$

$T \propto N^2$

Ex-3

```
for (i=1 ; i<= N   i++){
    for (j=i; j<=N; j++){
        // work
    }
}
>
```

1 — N		
2 — N	N-1	
3 — N	N-2	
⋮		
N-1	N	2
N	N	1

i=1	N
i=2	N-1
i=3	N-2
⋮	+2
	+1
i=N	

$1+2+3+4+\ldots N$

$\text{work} = \frac{N}{2}(1+N)$

$= \frac{N^2}{2} + \frac{N}{2}$

largest a

$T \propto N^2$

$a = 1$, $d = 1$

$$S_n = (a + a_L)\frac{n}{2}$$

$$= (1 + N)\frac{N}{2}$$

$$= \frac{N^2}{2} + \frac{N}{2}$$

↓ highest degree in polynomial.

$$\boxed{T \propto N^2}$$

Ex-4

```
int i,j;
for (i=1 ; i <= N ; [ i=i+k; ] ) {
    for (j=0, j< k, j++){
        // work
    }
    i = i + j;
}
```

i = i + j → would be k

j → fixed → k

inner loop

* k

outer loop

$\frac{N}{\cancel{2}k}$ times

$= \boxed{N}$

N, K are variable

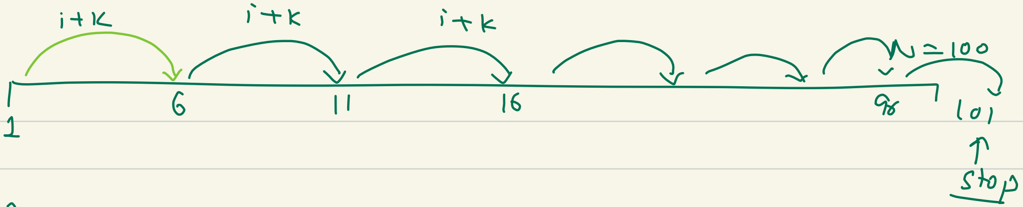
↑ user input

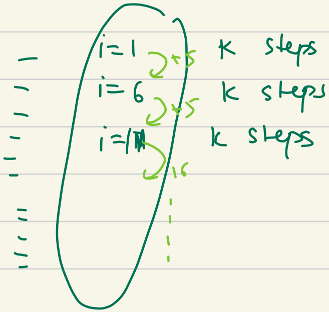
$$\boxed{k < N}$$

Example

N = 100

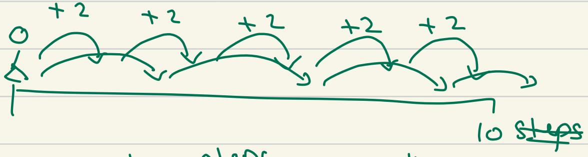
K = 5





$$\underbrace{\frac{N}{\cancel{K}}\ \text{times}}_{\text{outer loop}} * \cancel{K}\ \text{times} \left(\frac{100}{5}\right)\ 20\ \text{steps}$$

$\longrightarrow$ N times



$\frac{10}{2}$ steps

$\frac{10}{3}$ steps

$$\boxed{\text{Time} \propto N}$$

Jailer Problem

```
                                   N
⇒ ✓  for ( r = 1;   r <= 100 ;   r ++ ) {
                                          N
       ⇒ ✓  for ( d = r;  d <= 100,   d = d + r, ) {

              wor  //   doos[d] = 1 - doors[d];

                 3
        3
```

outer loop (N times) work

$N \log N$

→	$r = 1$	N doors ✓	1, 2, 3 ------ 100
→	$r = 2$	N/2 doors	
→	$r = 3$	N/3 doors	3, 6, 9 ------ 100
	⋮	⋮	
	$r = 100$	1 door	100

$$\text{work} = \Rightarrow \overbrace{N + \frac{N}{2} + \frac{N}{3} + \cdots + 1}^{\text{outer loop}} \times N$$

↑ ↑ ↑ ↑ ↑ ↑ N

$N\,(N + N + \cdots)$

$N \times N$

~~$N \times ($~~

$= N + N + N + N + N + - - -$

$= N(1 + 1 + 1 + + - - - - 1)$

$= \underline{N^2}$

$$= \underbrace{\overset{i=1}{N} + \frac{N}{2} + \frac{N}{3} + - - - - - - .}$$

$$= N\left(\underbrace{1 + \frac{1}{2} + \frac{1}{3} + - - - - - \frac{1}{N}}\right)$$

Sum of series ≃ log N

apprx

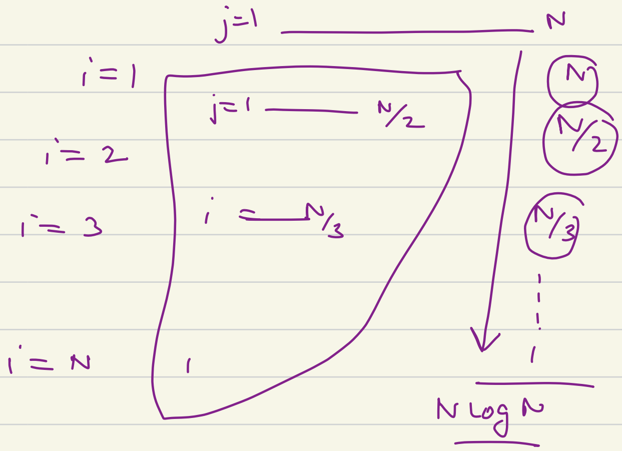
$$\text{Time} = \underline{N \log N} + \underline{N}$$

marking

Count the open doors

$N \log N$

$= O(N \log N)$

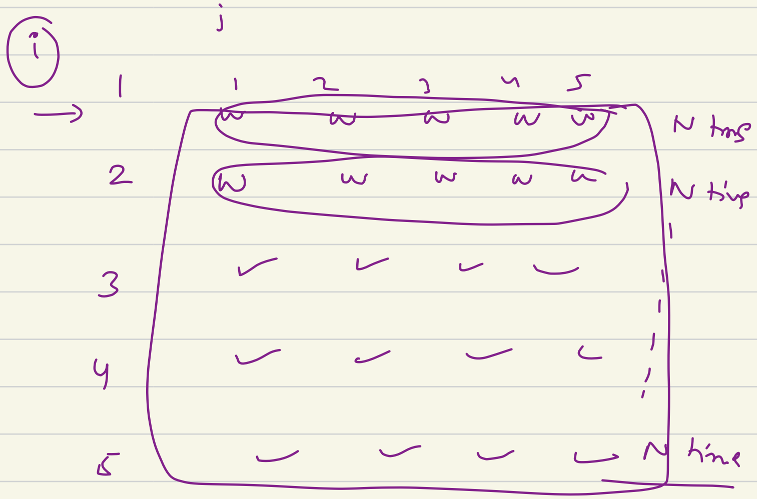


N = 5

```
for (i=1, i<= N, i++){
    for(j=1; j<=N; j++){
        print("W");
    }
}
```

$N * N$

$= \boxed{N^2}$



$N + N + \ldots$

$= N(N)$

$= N^2$

Estimate Time taken by ~~the~~ program

CPU

↳ 10^8 instructions in 1s

1s → 10^8 instructions

$T \propto 10^8$ iterations $\boxed{1\ s}$

$T \propto 10^{10}$ iterations $\boxed{100\ s}$

$T \propto 10^{12}$ iterations $\boxed{10^4 \text{ seconds}}$

$T \propto 10^{18}$ iterations $\boxed{10^{10} \text{ seconds}}$

↳ 317 years

$$\frac{10^{18}}{10^8} = 10^{10}$$

Unitary Method

10^8 ins → 1s

1 ins → $\frac{1}{10^8}$ s

10^{10} ins → $\frac{10^{10}}{10^8} = 100s$

50 yrs

Theoretical analysis's work

Prob $N = 10^6$

- Guy-1
- Guy-2
- Guy-3

				Output
N	One loop	10^6	→	0.01 s
N^2	Two loops	10^{12}	→	10^4 seconds (hours)
N^3	Three loops	10^{10}	→	317 years

intel i3

intel i5

Speed

10^8

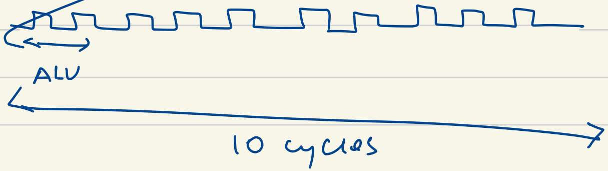
2.5 GHz

~~10^{9}~~

~~10^{10}~~ in 1s

2×10^8 ins → 1 s → **Standard**

$$10^6 \longrightarrow \frac{10^6}{10^8} = 10^{-2} = 0.01\,s$$



fr(____)

[a = b + c;

x64 vs x32

Memory

Puzzle → Circular Jail

↳ Logic → $O(N \log N)$ Brute Force

↓

Better ?

Logic ⇒ What doors would be open

⇒ $\boxed{20}$ → Rounds

Door ×

$\underbrace{1, 2, 4, 5, 10, 20}_{\text{divisors}}$

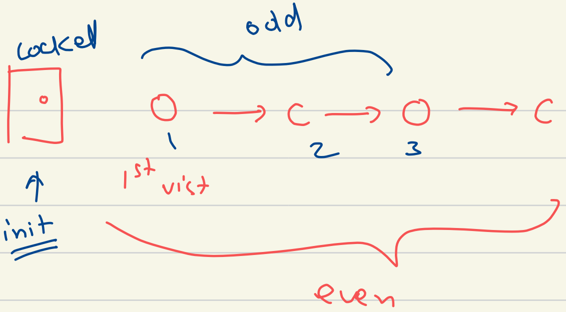
6 times

↪ Even visits → will never change the state of door

↪ odd visit

Door	Rounds	
✓ 1	→ 1	
2	→ 1, 2	
3	→ 1, 3	
✓ 4	→ 1, 2, 4	3 time
5	→ 1, 5	
6	→ 1, 2, 3, 6	
7	→ 1, 7	
8	→ 1, 2, 4, 8	
✓ 9	→ 1, 3, 9	3 times
⋮		

Doors which are perfect sq will be visited odd times



Div

30 → 1, 2, 3, 5, 6, 10, 15, 30 — even divisors

36 → 1, 2, 3, 4, 6, 9, 12, 18, 36 — 2N + 1 odd divisors

Perfect sq. $1^2, 2^2, 3^2, 4^2, 5^2, 6^2, 7^2, 8^2, 9^2, 10^2$

$\Rightarrow \underbrace{1, 4, 9, 16, 25, 36, 49, 64, 81, 100}_{10 \text{ Doors}}$ ← Door No's which would be open

$\sqrt{4} = 2$

$\sqrt{36} = 6$

$\sqrt{~~x~~} =$

Find no of perfect sqares less<= than $\boxed{N}$.

Code-2

Linearly search

```
i = 1
while(i*i <= N){   print(i*i);
        i=i+1;

3
```

$\boxed{N=100}$

1	$1^2 <= 100$
2^2	$2^2 <= 100$
3^2	$9 <= 100$
4^2	$16 <= 100$
⋮	
9^2	$81 <= 100$
10^2	$100 <= 100$
11	stop

N=100 → Perfect squares

↓

10 iterations

$$T \propto \sqrt{N}$$

$$\propto N^{1/2}$$

$N = \boxed{10^9}$

1

$N \log_2 N$

work $= 10^9 \times 30$

$= 3 \times 10^{10}$ Ins

Time $= 3 \times \frac{10^{10}}{10^8} = 300$ seconds

2

$T \propto \sqrt{N}$

work $= \sqrt{10^9} = \sqrt{10^{10}}$

$= 10^5$ (over-estimate)

Time $\Rightarrow \frac{10^5}{10^8} = 0.001$ seconds

$T \propto \log_2 N$

work $= \log_2 10^9$

$= 30$ steps

Time $= \frac{30}{10^8} = 3 \times 10^{-7}$ seconds

$\log_2 1024 = \boxed{10}$ $\qquad$ $2^{\boxed{10}} = \underline{1024}$

① $\log_2 1000 = \underline{10}$ $\quad$ (9.xx)

$$\log_2 x^y = y \log_2 x$$

② $\log_2 1000\,000 = \boxed{20}$

$= \log_2 (1000)^2$

$= 2 \log_2 1000$

$= 2 \times 10$

$= \boxed{20}$

③ $\log_2 10^9 = \log_2 (10^3)^3 = 3 \log_2 1000 = \boxed{30}$

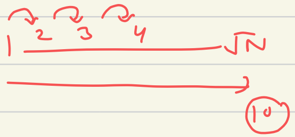
Solution → finding the sqrt ⇒ $\boxed{N}$

Basic way

```
i=0
while (i^2 <= N) {
    i++;
```

$i <= \sqrt{N}$

N = 100



3

code-1 N log N < √N iterations < Code-3

Brute force

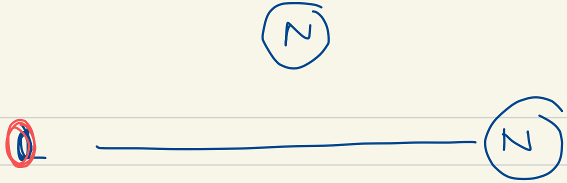
Code-2

better

→ logic → sqrt

Best

~~Math.sqrt(n)~~ ←

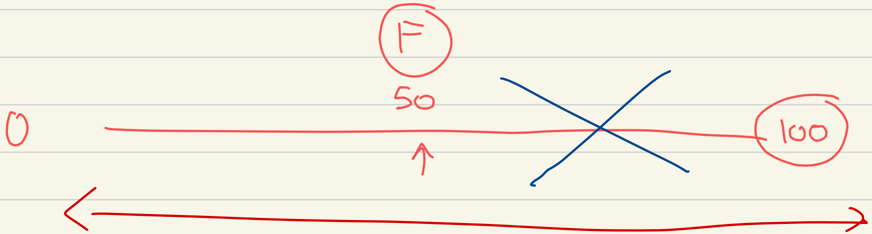


$N = 100$

$N = \textcircled{1}$

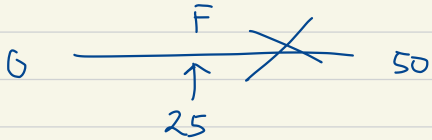
$\text{sqrt}(N) \leq 1$

$\text{Sqrt}(100) \leq 100$



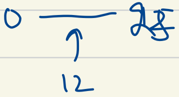
$50^2 <= 100$

N → Range



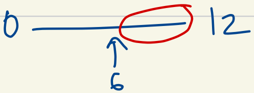
$\frac{N}{2}$ → Range

$25^2 <= 100$



$\frac{N}{4}$

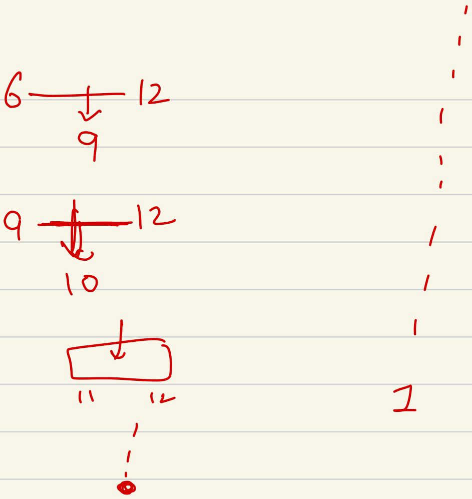
$12^2 <= 100$



$\frac{N}{8}$

$\boxed{6^2 <= 100}$ T

ans = 6



$\boxed{9^2 <= 100}$ T

ans = $\cancel{8}$ 9

$10^2 <= 100$

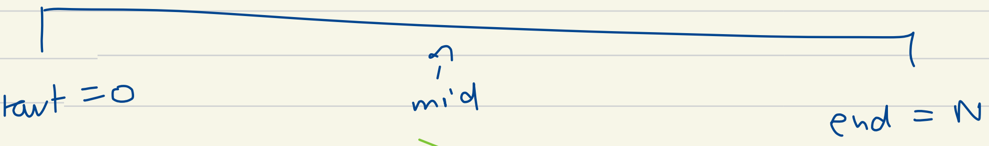
ans = 10

$11^2 <$

Oversets

Binary Search

how to find sqrt efficiently.



Times ∝ $\log_2 N$

till single element

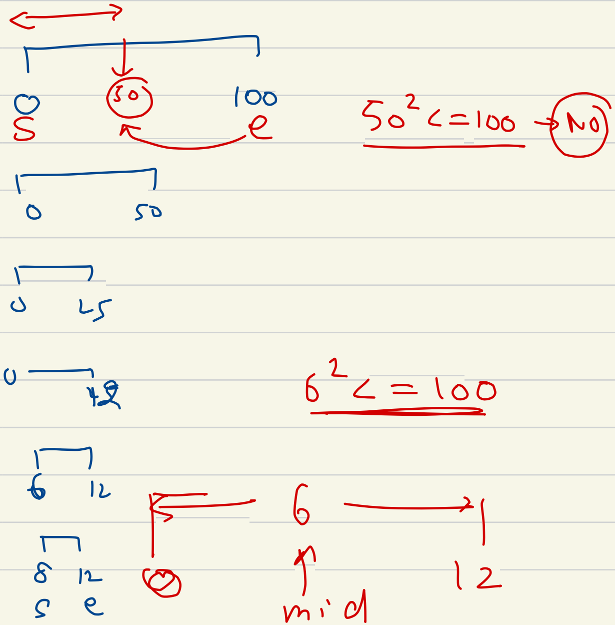
```
s = 0
e = N
int ans = 0;
while (s<=e) {
    mid = (s+e)/2,
    if (mid * mid <= N) {
        ans = mid;
        s = mid + 1;
    }
    else {
        e = mid - 1
    }
}
print(ans);
```

$S > e$ → stop

Times →

→ work

$7^2 <= 50$



$50^2 <= 100$ → NO

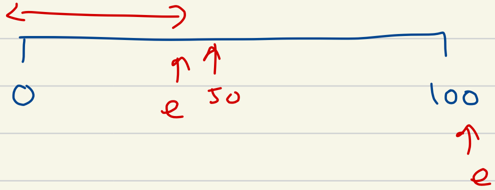
$6^2 <= 100$

ans = $\boxed{\cancel{0}}$ $\boxed{\cancel{5}}$ $\boxed{\cancel{8}}$ $\boxed{10}$

N = 100 $\sqrt{N}$ = ?

s = 0   e = 100   mid = $\underline{50}$ $50^2 <= 100$

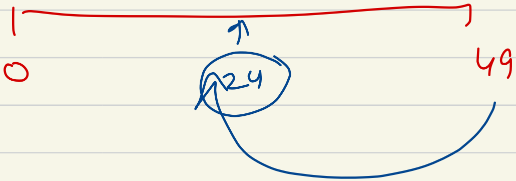
↳ go left   e = mid − 1



S = 0   e = 49   mid = 24 $\underline{24^2 <= 100}$

↳ go left

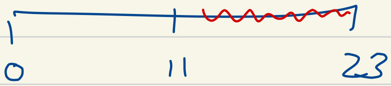
e =



24

S = 0   e = 23   mid = 11 $11^2 <= 100$

↳ go left $\underline{e = mid - 1}$

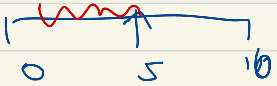


$\underline{S = 0}$ e = 10   mid $5^2 <= 100$

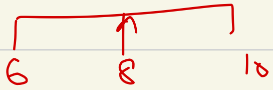
Yes

↳ go Right

S = mid + 1



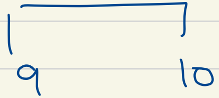
$s = 6$ ~~mid~~ $e = 10$ mid $= 8$



$8^2 <= 100$

↳ Right

$s = 9$ $e = 10$ mid $= 9$



↳ Right $9^2 <= 100$

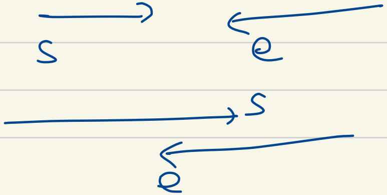
$S = 10$

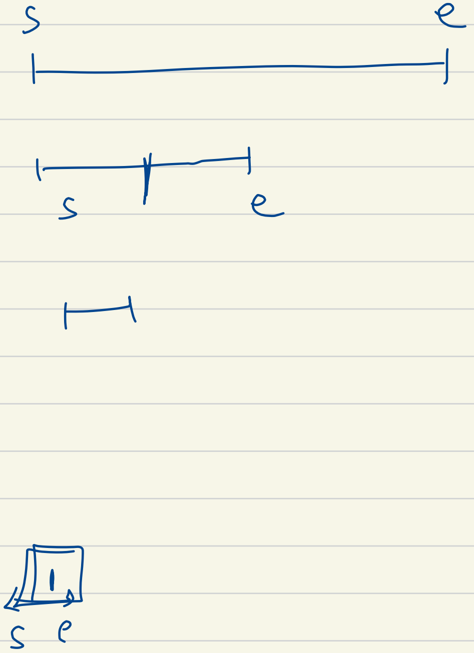
$s = 10$ $e = 10$ mid $= 10$

↳ go Right $10^2 <= 100$

$s = 11$, $e = 10$

$\boxed{s > e}$ → Stop





steps.

0 step → N

1nd step → $\frac{N}{2} = \frac{N}{2^1}$

2nd → $\frac{N}{4} = \frac{N}{2^2}$

3th step → $\frac{N}{8} = \frac{N}{2^3}$

⋮

⇒ k steps → $\boxed{1 = \frac{N}{2^k}}$

$$N = 2^k$$

$$\Rightarrow \log_2 N = \log_2 2^k$$

$$\Rightarrow \log_2 N = K \underset{\substack{\| \\ 1}}{\boxed{\log_2 2}}$$

$$\Rightarrow \boxed{K = \log_2 N}$$

$$\Rightarrow \log_2 \underline{1000} = 10 \qquad \log_2 X^Y = Y \log X$$

$$\begin{tikzcd}
N \arrow[r, bend left] & N/2 \arrow[r, bend left] & N/4 \arrow[r, bend left] & N/8 \arrow[r, bend left] & {} \arrow[r, bend left] & {} \arrow[r, bend left] & 1
\end{tikzcd}$$

$$\xrightarrow[\log N \text{ steps}]{\hspace{6cm}}$$

$$N = 16 \qquad \begin{tikzcd}
16 \arrow[r, bend left] & 8 \arrow[r, bend left] & 4 \arrow[r, bend left] & 2 \arrow[r, bend left] & 1
\end{tikzcd} \Rightarrow 4 \text{ steps} \qquad \log_2 16 = 4$$

$$\begin{tikzcd} 100 \arrow[r, bend left] & 50 \arrow[r, bend left] & 25 \arrow[r, bend left] & 12 \arrow[r, bend left] & 6 \arrow[r, bend left] & 3 \arrow[r, bend left] & 1 \end{tikzcd}$$ ⇒ 6 steps

$\log_2 100 = \boxed{6}.\cancel{xx}$

$$\begin{tikzcd} N \arrow[r, bend left] & N/3 \arrow[r, bend left] & N/9 \arrow[r, bend left] & \cdots 1 \end{tikzcd}$$

$\boxed{\log_3 N}$

$\log 1000000$ $\quad \log_2 (1000)^2 = 2(\log_2 1000)$ $\nearrow 10$

$N \log N$	$\sqrt{N}$	$\log N$
$N = 10^6$ $10^6 \times 20$ $= 2 \times 10^7$ steps	$=$ 1000 steps	$=$ 20 steps 2×10^{-7} seconds.

Homework ✯ ✯

↳ Bruteforce

↳ Linear search

↳ Binary search

Code it

Notes 20mins

Expect

Expect

↳ Regular Live ↑

70 hours

↳ 2.30 hours

2 months.

→ Time & Space Complexity (2)

→ Arrays (6)

→ Bit manipulation (2-3)

↳ Maths (2)

↳ Strings (1)

↳ Sorting (1)

↳ Hashing (2) ✓

↳ Rec. (2) ✓

Dataset

→ Linked

→ Stack

→ Queue

→ Trees

4 lec.

fundamentals

→ Subset & Subseq.

4 months.

Adv

→ greedy
→ DP
→ graphs
→ sortig
→ heaps
→ hashtable