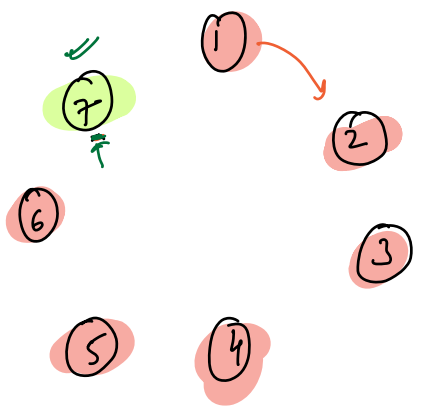
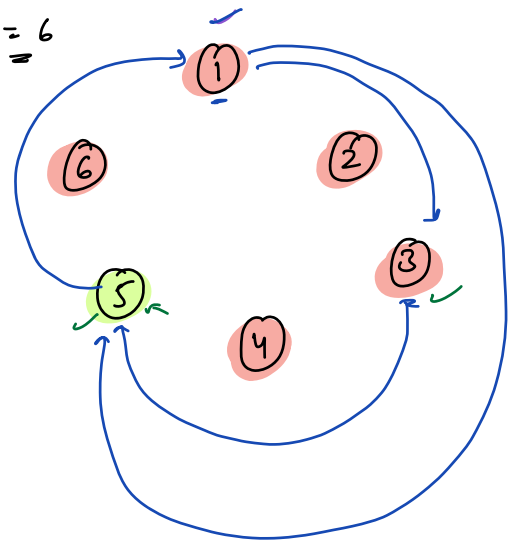
# Josephus Problem

→ N people will stand in circle, Initially person 1 will have knife, Person will will kill person next to her in clowckwise & pass the knife to next clock wise person & Repeat process & find last person standing

N = 7



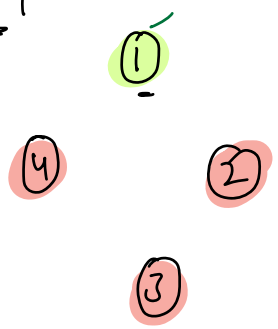

N = 6




observations

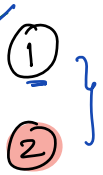
1) N-1 is alive ✗
2) If N is odd : N ✗
   N is even : N-1
3) largest prime number < N ✗

N = 4

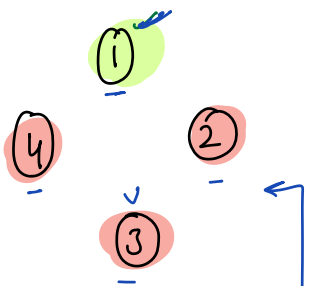


If N=1 : winner = 1

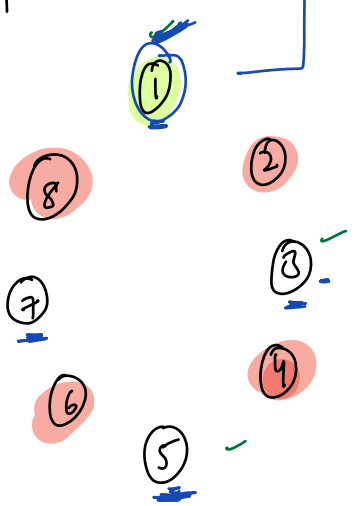
If N=2 : Wi = 1



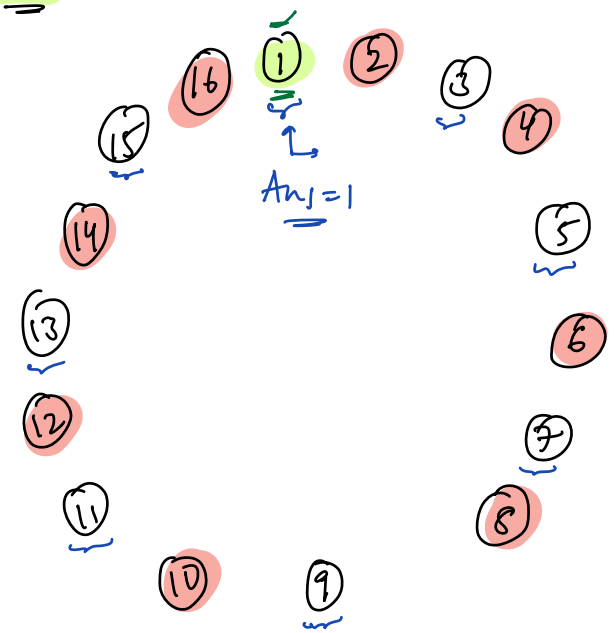
If N=4 : Wi = 1



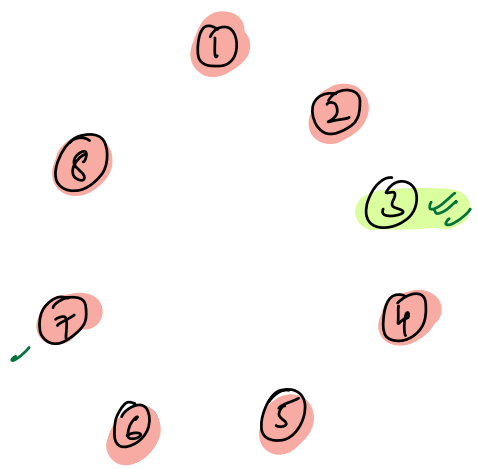
If N=8 :



N=16




N=8 → Starting 3
→ Winner: 3

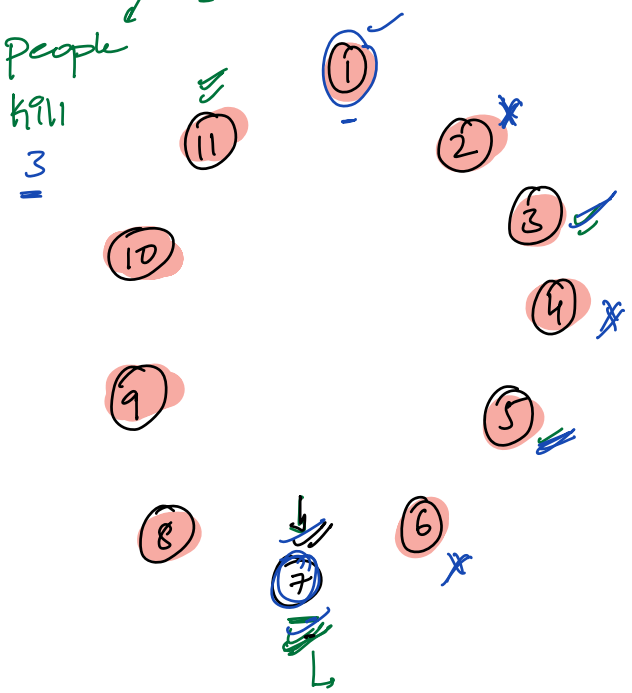


Observations: If there are power of 2 persons, Who ever starts they will be last person standing

Nearest power of 2 = 8

N = 11 → 8

people kill 3



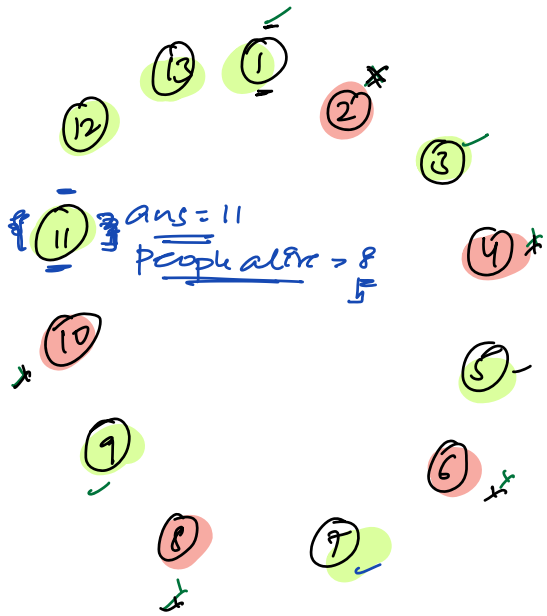
→ How many people are standing

Ans = 7

After killing we at 7.

→ Nearest power of 2 = 8

N = 13 → kill people 5



ans = 11

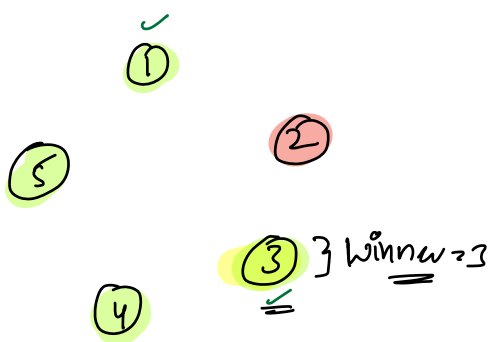
people alive = 8

→ After killing 5 people

we are at = 11

N = 5 → 4

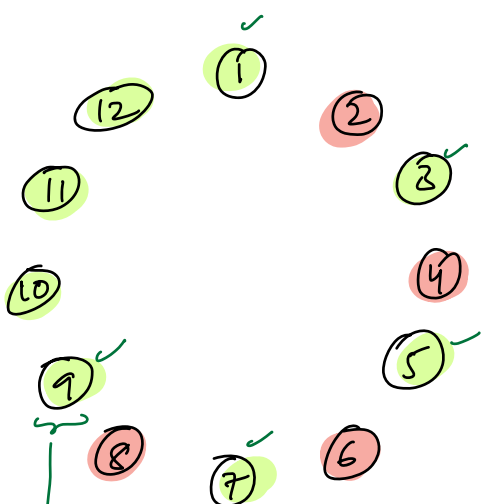
kill 1



3 } winner = 3

After 1 killing were at 3

N = 12 → 8 } kill 4 people → 2*4+1 = 9



9 will winner

N = 100 → Nearest power 2 → 64

↳ people kill : 36

↳ last person standing = 36 * 2 + 1 = 73

N = 70 → Nearest power of 2 → 64

↳ people kill = 6

↳ last person standing = 13

## Pseudo Code

1) Given N

x = Nearest Power of 2 <= N  (→ on your own)

kill = N - x

ans = 2 * (kill) + 1

}

N = 250

| 2 |
|---|
| $2^0 < 50$ |
| $2^1 < 50$ |
| $2^2 < 50$ |
| $2^3 < 50$ |
| $2^4 < 50$ |
| $2^5 < 50$ } nearer 25 |
| $2^6 > 50$ |

If : N = 200 → Nearest power of 2 <= 200 → 128

kill : 72

winner → 72 * 2 + 1 = 145

28) Majority Elements ✓

Given an array of size N, Return if there exists a no with (frequency) > (N/2), without any extra space

SC: O(1)

No: of occurence of that element

Q1): 1 6 1 1 2 1 → N = 6 : freq > N/2 > 3 ≥ 4

ans = 1

Q2) N = 10, freq > N/2 > 10/2 > 5 >= 6

3 4 8 3 7 3 3 8 3 3 ans = 3

Q3) N = 8, freq > N/2 > 8/2 > 4 >= 5

{2 9 6 2 7 6 2 2} ans = 2 ✗ freq(2) = 4 < 5

Ex

No majority Element

Solution 1) For every element iterate & frequency

TC: O(N²) SC: O(1)

2) Sort & get freq of all Elements

TC:

Sorting: (N log N) 3 3 3 3 3 3 4 7 8 8 → O(N)

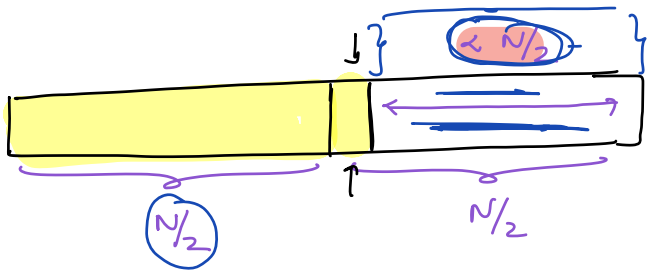
freq of 3 = 6 freq=1 1 freq of 8 = 2

TC: O(N log N + N)

Obs: How many majority Elements can we have =

$freq > N/2$, $freq >= (N/2+1)$

// All Majority Elements we are writing in left side




↳ At max we can only have a single majority element.

Obs2: If we can majority element is at center

↳ TC: $O(N \log N)$ ↳ $O(1)$

↳ Note: What's the guarantee that center element is majority

{ 2 9 6 2 7 6 2 2

Sorting: 2 2 2 2 6 6 7 9

Is 2 majority x

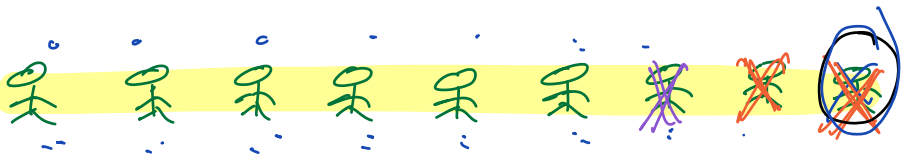
Is 6 majority x

Soln: Sort, get center, check majority or not.

Moore's Voting Algorithm

Elections: 15 MLA Seats

Satya : 9



Jatin : 3

13 MLA > 13/2 > 6 >= 7

Renu : 3

11 MLA > 11/2 > 5 >= 6

9 MLA > 9/2 > 4 >= 5

// Cancelled 2 MLA Seats All together

// Cancelled 2 MLA Seats

// Cancelled 2 MLA

Obs: Take any 2 diffent party seats remove them

Majority still wont change

N seats

Majority //

> N/2 = N/2 + 1 //

// 2 different seats

N-2

// N/2 seats //

To form goverment > (N-2)/2 >= (N-2)/2 + 1

>= N/2

Issue: Deleting 2 Distinct Elements

3 4 8 3 7 3 3 8 3 3

Maj : ✗ ✗ ✗ ✗ 3 ← final majority element

freq: * * * * 2

N = 12:

6 6 8 4 4 7 4 4 4 4 4 6

Maj = ✗ NULL 4 3 → Majority Element

freq = x ∅ x x ∅ 4 5 4

N = 8:

3 2 4 7 2 8 2 2

Maj ✗ NULL ✗ NULL ✗ NULL {2} → probable Candidate

freq: x ∅ x ∅ x ∅ x 2

{ Iterate & check if it's majority or not }

TC: ( O(N) + O(N) ) } 10:50 break

SC: O(1) }

Eg: 3 3 3 3 4 5 6 7 8 3

Maj: 8 NULL

Freq: 2 2 1 1 1 1

1) Majority Elem: $> N/2$

Steps:

1) Iterate & get Majority Element

2) Check if Majority Element is correct or Not

3 3 3 3 4 4 6

Maj: 3 → probable candidate = 3

Freq: 1 2 2 1

Is 3 majority majority = ✓

308) Majority Element 2

An Element is said to be majority element if

if frequency > N/3

N = 8

ar[8]: 4 5 3 2 5 2 4 2 : > 8/3 >= 3

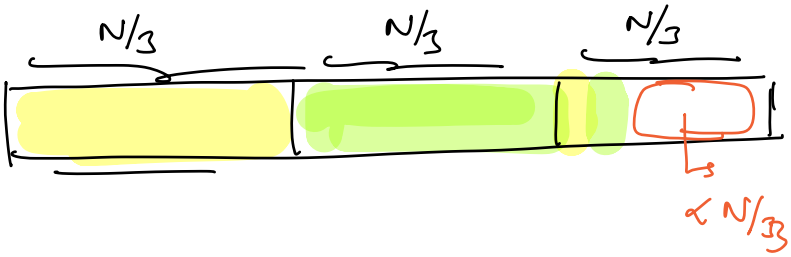
freq(2) >= 3

ar[8]: 3 2 -1 6 2 6 7 -1 : No majority

ar[10]: -2 4 3 6 3 4 7 3 10 3 : > 10/3 >= 4

freq(3) >= 4

At max how many majority Elements can we have >= 2 ?



obs: Take any 3 different data delete Then majority won't change

N Elements → 3 delete → N-3 Elements

Majority freq > $N/3$ >= $(N/3 + 1)$

= $(N/3 + 1)$

$N/3$

Majority freq > $\frac{(N-3)}{3}$ >= $\frac{(N-3)}{3} + 1$ = $N/3$

ar[]: 4 5 3 2 5 2 4 2

Maj1 : ~~NULL~~ ~~4~~ NULL 2

freq: ~~0~~ ~~1~~ 0, ~~0~~ ~~1~~ 2

Maj2: ~~NULL~~ ~~5~~ NULL, ~~3~~ NULL

Freq: ~~0~~ ~~1~~ 0, ~~0~~ ~~1~~ 0

Is 2 majority Element yes

Iterate & check 2

Deleted Elem:

3, 4, 5, 4, 2, 5

arr[13]:

3 x 6 7 6 5 6 6 3 1 3 6 3

Maj1 : 6

freq1 : * 2

Is 6 majority ✓

Is 3 majority ✓

Maj2 : 3

freq2 : * 2

If $N_i > N/4$ :

→ Delete 4 distinct element majority won't change

→ At 3 majority Elements

If $N_i$ freq $> N/k$

→ Delete k distinct elements majority won't change

→ At max k-1 majority elements can here.

O(k) *

Funny Concept, int range $\approx 10^9$, long range $\approx 10^{18}$

int a = $10^6$, b = $10^5$

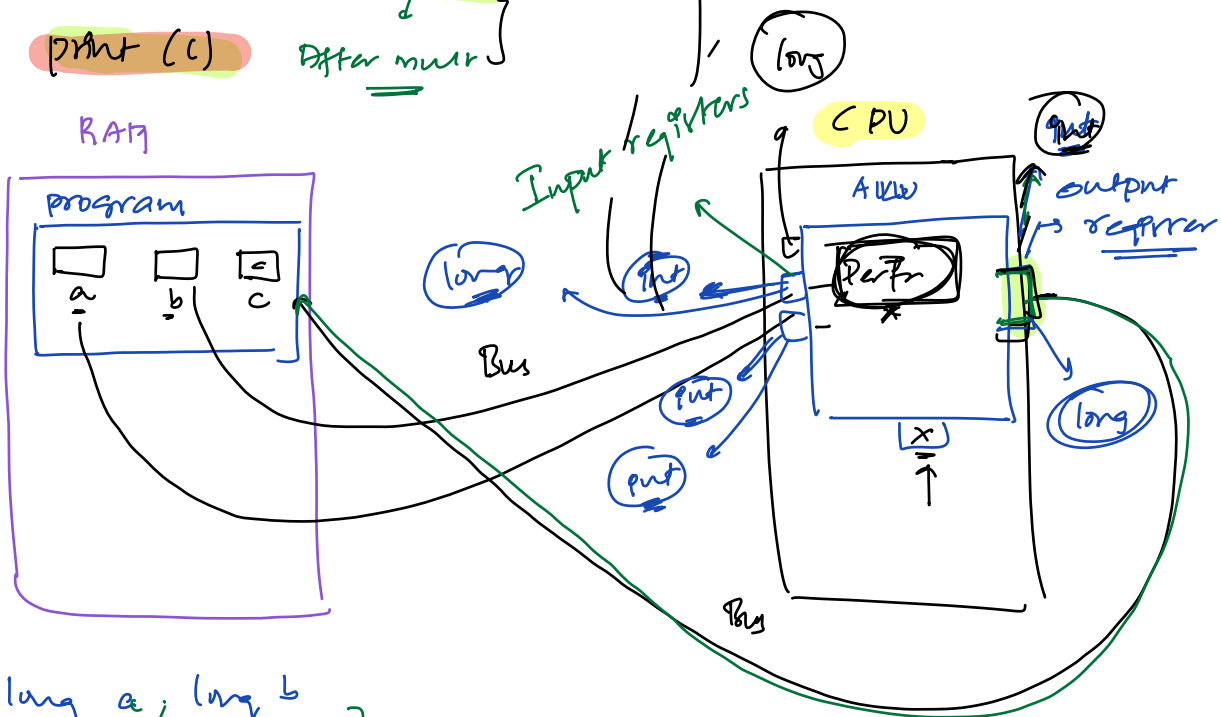
```
int c = a * b
print (c)
```
} overflow ✗

```
long c = a * b
print (c)
```
} overflow ✗

```
long c = long(a * b)
print (c)
```
} overflow

Diffr mult

What happens when we multiply 2 integers




```
{ long a; long b
  long c = a * b }
```

{1 L 3}