classmate
Date
Page

1 2 3 4 5

Insertion sort.

5	8	1	3	2	4

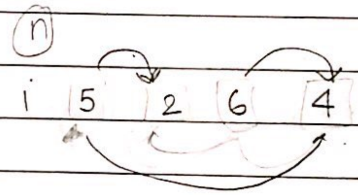
5	2	4	6

```
for (int i = 1 ; i < n ; i++)
{       int k = i-1;    int min = arr[k]
        while ( k >= 0  && arr[k] > min)
                        arr [k+1] = arr [k];
                        k -- ;
}
                arr [k+1] = min;
```

1. Outer loop runs to see which el. is to be inserted.
2. Inner loop checks where the el. is to be inserted.

→ Complexity → (n^2)
Remains same even if binary search is used instead ($n[\log n + n]$)

→ **Inversion**
If $a[i] > a[j]$
& $i < j$



If given an array with k inversions, then the complexity would be $O(n+k)$.
where : n : to check each preceding el.
k : the no: of inversions.

Bubble Sort :
Largest el. bubbles out to the end.

Selection Sort :
Modifying bubble sort by not swapping every el. rather just looking for the largest el.

Binary search : $T(n) = T(\frac{n}{2}) + 2C$.

Quick Sort.

Q: How to avoid multiple iterations in selection sort.
Add a flag condition in the if ( )

→ For a Binary Search

Complexity : $T(n) = T(\frac{n}{2}) + 2C$

C : To find middle el.
C to check with el to search

$= T(\frac{n}{4}) + 2C + 2C$

$\vdots$

$= 2C \cdot \log_2 n$

→ For Ternary search

Comp : $T(n) = T(\frac{n}{3}) + 4C$

$= 4C \log_3 n$

Comparing the two :-

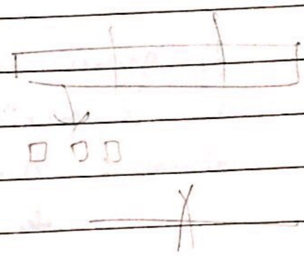
$\log_2 n$ vs $2 \log_3 n$.

$\log_2 n$ vs $\frac{2 \cdot \log_2 n}{\log_2 3}$.

1 vs $\frac{2}{\log_2 3} > 1$.

Q: In Merge sort which one is better? 2-Part or 3-Part?

Selection sort :-



```
for( int i=0 ;  i< n-1;  i++) {
  int min = i ; key = A[i] ;
  for (int j=i+1;  j<n;  j++)
       if ( A[j] < key)
          {    min = j; }
     int temp = A[i];
         A[i] = A[min];
         A[min] = temp;
}
```

Bubble sort :-

```
for (int i=0 ; i<n; i++){
  for (int j=0; j<n-i-1; j++){     int x=0;
        if (A[j] > A[j+1])
        Swap (A[j], A[j+1])
        else   x++;

    if (x == n)
          break;
    }
}
```

Selection sort :-

```
for ( int i=0;  i< n-1 ; i++){.                 1 4 9 5 2 3
    int  min = A[i] ;                           1 2 3 4 5 9
    int  pos  =  i;
    for(int j=i+1 ;  j< n ; j++){   int f=1;
        if ( A[j] < min)
           { pos  =  j  ;                       1 3 2 4 5
             min  = A[j] ;    f=0;              1 2 3 4 5
           }
        else   if(f != 0).
             f = 1;
    }                         A[i] =A[pos]; A[j] = min;
    if  (f = 1)
       x++;
    if ( x == n*n) .
       break;
    }
}
```

CLRS by Cormann.

Quick sort:

Pick a pivot: Hope to place it in its sorted position

If, best case - pivot is the middle el.

& | n/2 | pivot | n/2 |

In this case; no: of steps (logn)

Worst case: pivot itself ie the smallest el.

then no: of steps: = n.

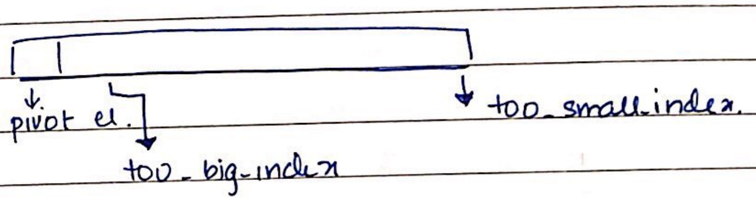
$\because$ | P | | n-1 |

| P | | n-2 |

⋮

n steps.

Algorithm:



Step 1: while (A[too_big_index] <= pivot) ++ too_big_index

2: while (A[too_small_index > pivot) -- too big_index

3: if (too_big_index < too_small_index)
swap.

4. if (too_big_index <= too_small_index) continue

5. else; swap (pivot, too_small index)

Best case: $T(n) = 2T(n/2) + cn. \rightarrow n\log n$.

Worst: $T(n) = T(n-1) + cn \rightarrow n^2$

$\boxed{Q:}$ How to choose the best pivot?

- Find the median of the 1st last & middle el.
- Group no:s of 5 and sort them individually, then sort the medians of all these groups (further group in 5s).
-

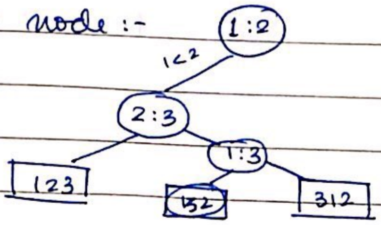
Running time of the algorithm = No: of steps = length of the path.

No. of leaves : n!
At most 2^h leaves ⇒ h is h

$n! \le 2^h$
$\log(n!) \le h$

- Decision tree examples.

Root node :-



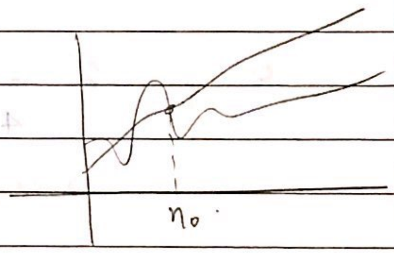
No: of leaves = $n!$
Height of the tree = no: of steps the algorithm can atmost take.

h = # no: of steps
if ht = h ⇒ at most 2^h leaves.

∴ $n! \le \cancel{k}.\,2^h$

$\boxed{\Theta(n\log n) = \log(n!)} \le h$

- Asymptotic Analysis :



O - notation :- (upper bound).

$O(g(n))$ { there exists a c, n_0 such that $0 \le f(n) \le c g(n)$ for $n > n_0$ }.

no unique set of values!

Eg: If $f(n) = 5n^2 + 7n + 8$ (from code... say).
claim : $f(n) = O(n^2)$.

Now, $5n^2 + 7n + 8 \le 20n^2$ for all $n > 1$.
(can take 20, 25 .... > 20) (can take 1, 10 ... > 1)

Eg. $2n^2 = O(n^3)$
∵ By def $2n^2 < 2 \cdot n^3$
↑
c

~ Ω Notation :- (Lower bound).

$\Omega\, g(n)$: { there exists a c, n_0 such that $f(n) \geq c \cdot g(n)$ for $n > n_0$.

~ Θ notation :-

$c_1 g(n) \leq f(n) \leq c_2 (g n)$

Indication :- Θ is approximately the func. f itself

Q: $1/2\, n^2 - 3n = \Theta(?)$

If $g(n) = n^2$. ~~$c_1 g(n)$~~ $c_1 < \frac{1}{2} - \frac{3}{n} < c_2$

Let $c_1 = \frac{1}{4} n^2$. $c_2 = 4n^2$. $n_0 = 7$.

[$c_1 : \frac{1}{14}$, $c_2 = 1$

Q: $2^{n+1} = \Theta(2^n)$.

$c \geq 2$ $\because 2^{n+1} = 2 \cdot 2^n$

Q: $2^{2n} = \Theta(2^n)$.

$4^n \leq c \cdot 2^n$.

Here c depends on n ; contradiction.

Q: Prove: $\max(f, g) = \Theta(f+g)$.

Let f be $\max(f, g)$.

$\therefore$, $c_1(f+g) \leq f \leq c_2(f+g)$.

Take $c_1 = 1/2$ $c_2 = 1$.

Q: Prove: $\log(n!) = \Theta(n \log n)$

$\log n! = \log(n \cdot n-1 \cdot n-2 \ldots) \leq \log(n \cdot n \ldots n)$

$\Rightarrow c_2 = 1$ $\leq \log n^n \leq n \log n$.

$\log n! = \log(\underbrace{\boxed{\cancel{1 \cdot 2 \cdot 3 \ldots}}}_{n/2 \text{ terms}} \; n-1 \cdot n)$

cut

$\therefore \log n! \geq \log\left(\frac{n}{2}+1 \cdot \frac{n}{2}+2 \cdots n\right)$

$\geq \log\left(\frac{n}{2} \cdot \frac{n}{2} \cdot \frac{n}{2} \cdots \frac{n}{2}\right)$

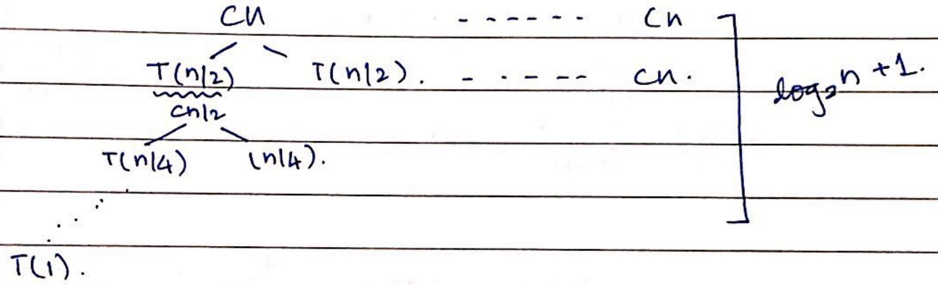
$\geq \log\left(\frac{n}{2}\right)^{n/2}$

$\geq \frac{n}{2}\left(\log n - \log 2\right)$

- **RECURRENCE RELATIONS :**

Eg: $T(n) = 2T(n/2) + cn$

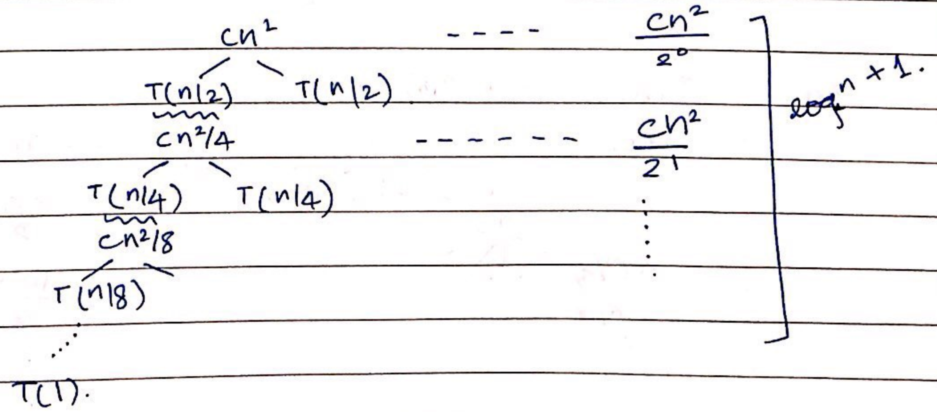
Tree method :-



$\therefore$ comp $= cn(\log_2 n + 1)$

$T(n) = cn\log_2 n + cn$

Eg: $T(n) = 2T(n/2) + cn^2$



Last level :- $2^{\log_2 n} = n$

$\therefore$,

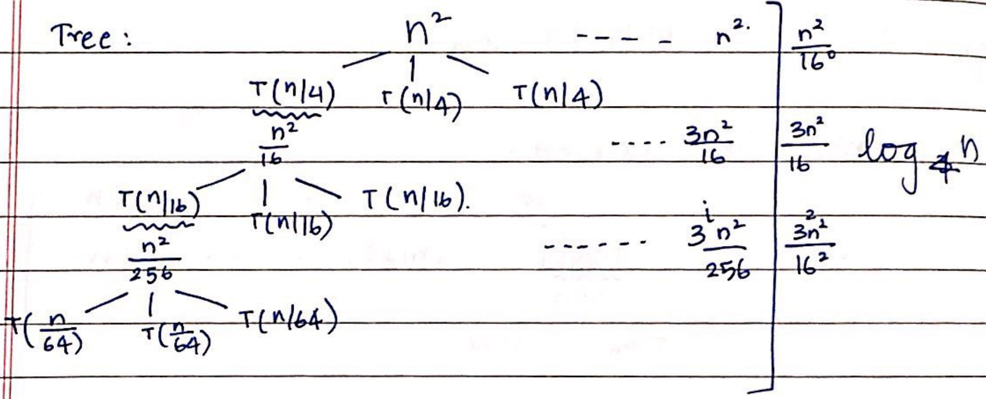
Summation : $T(n) = n + \sum_{i=0}^{\log_2 n - 1} \frac{cn^2}{2^i}$

$= n + cn^2 \underbrace{\sum_{i=0}^{\log_2 (n-1)} \frac{1}{2^i}}_{\text{const.}}$

$= \underline{\underline{O(n^2)}}$.

Eg: $T(n) = 3T(n/4) + n^2$.

Tree :



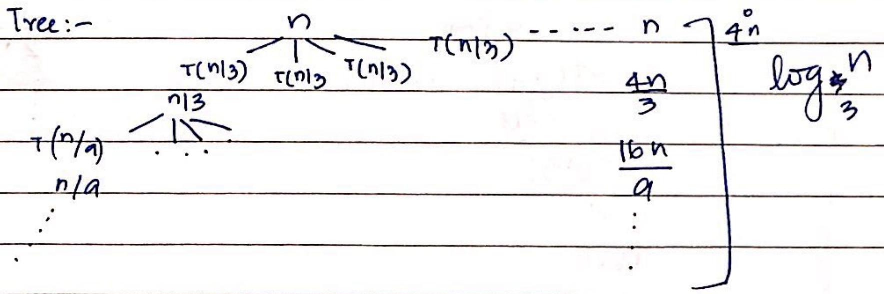
$\therefore$, $T(n) = \cancel{n} + \sum_{i=0}^{\log_4 n - 1} \frac{3^i n^2}{16^i} + 3^{\log_4 n}$

$= \cancel{n} + n^2 . \underbrace{\left(\sum_{i=0}^{\log_3 n - 1} \frac{3^i}{16^i} \right)}_{\text{constant}} + 3^{\log_4 n}$

$= \underline{O(n^2)}$.

Eg: $T(n) = 4T(n/3) + n$.

Tree :-

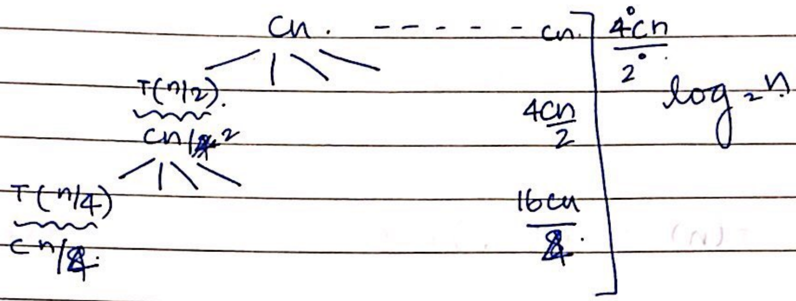


$$4^{\log_3 n} + n\left(\frac{4^{\log_3 n}}{n} - 1\right)$$

$$= 4^{\log_3 n} + \sum_{i=0}^{\log_3 n - 1} \frac{4^i n}{3^i} = 4^{\log_3 n} + n\left(\frac{4}{3}\right)^{\log_3 n} - 1.$$

$$\cancel{= O(n) \cdot 4^{\log_3 n} + n\,\frac{4}{3}\,\frac{\left(\left(\frac{4}{3}\right)^{\log_3 -1} - 1\right)}{4}}$$

Eg: $T(n) = 4(T(n/2) + cn.$



$$\therefore T(n) = 4^{\log_2 n} + \sum_{i=0}^{n-1} 2^i cn.$$

$$= 4^{\log_2 n} + n\sum_{i=0}^{n-1} 2^i c.$$

$$= 2^{\log_2 n^2} + n\sum_{i=0}^{n-1} 2^i c.$$

$$= n^2 +$$

$$= O(n^2).$$

- For general no:- recurrence

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n).$$

$$T(n) = a^{\log_b n}\, T(1) + \sum_{i=0}^{\log_b n - 1} a^i \cdot f\left(\frac{n}{b^i}\right).$$

Q: $T(n) = T(n/3) + T(2n/3) + nc.$

HW solve using correct

- Masters theorem
  → check slides.

- Substitution Method:

Step 1. Guess the form of sol
2. Verify by ind.
3. Solve

Eg: $T(n) = 4T(\frac{n}{2}) + n$ $\qquad T(1) = 1$.

Guess: $T(n) \leq cn^2$.

$T(2) = 4T(1) + 2 \leq c.4$

$\frac{6}{4} \leq c$.

& $T(3) = 4T(1) + 3 \leq c.9$.

$\frac{7}{9} \leq c$.

Assume $2 \leq k \leq n-1$

$T(k) \leq ck^2$

then $T(n) = 4T(n/2) + n$

$T[n] \leq 4c(\frac{n}{2})^2 + n$

$T(n) \leq cn^2 + n$.

But this creates an issue.

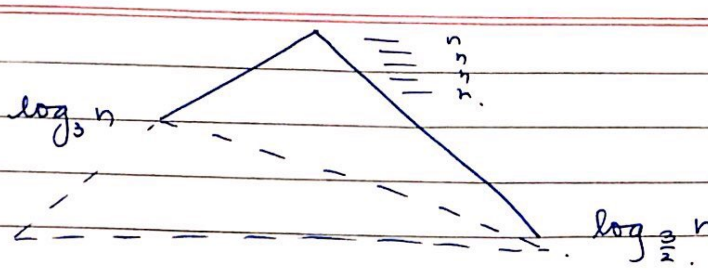
$\because$, $T(n)$ should be $\leq cn^2 - c_2 n$ .... let

$\therefore$, $T(n) \leq 4.\left[\frac{cn^2}{4} - c_2\frac{n}{2}\right] + n$.

$\leq cn^2 - 2c_2 n + n$.

$\leq cn^2$

Q7



∴, $n \log_3 n \le T(n) \le n \log_{\frac{3}{2}} n.$

$$n\left(\frac{\log_2 n}{\log_2 3}\right) \le T(n) \le n \cdot \left(\frac{\log_2 n}{\log_2 3/2}\right).$$

∴, $\underline{\Theta(n \log n)}$.

$\log_3 n$

$\log_{\frac{3}{2}} n$

$\therefore \quad n \log_3 n \le T(n) \le n \log_{\frac{3}{2}} n$

$$n\left(\frac{\log n}{\log_2 3}\right) \le T(n) \le n \cdot \left(\frac{\log_2 n}{\log_2 3/2}\right)$$

$\therefore \quad \underline{\Theta(n \log n)}$

$\boxed{\text{TUTORIAL - 1}}$

Given any array Eg: 3 1 5 2
At each level, an op. is defined as picking any el. and placing it anywhere. What are the min no: of operations?

4 2 3 5 1 6

2 3 4   1 5 6   3+1+1 - 4

10 7 1 3 4 6 2 5 9 8

10 7 1 3 4   6 2 5 9 8   2 5 6 8 9

10 7 1   3 4   6 2 5   9 8   2 6 5 8 9

10 ~~10 7~~ 1   3 4   6 2 5   9 8

10   7   6   2

2 5 6 8 9

7 10 3 4   10   3 4

- Applying masters theorem:

$T(n) = 8T(n/4) + n^{3/2}$.

Here $n^{\log_b a} = n^{\log_4 8} = n^{\frac{\log_2 8}{\log_2 4}} = n^{\frac{3}{2}}$

And $f(n) = n^{3/2}$.

$\therefore \Theta(n) = \boxed{n^{3/2} \log n}$

$\boxed{Q:}$

a) $T(n) = 2T(n/2) + 1$.

b) $T(n) = 4T(n/2) + 100n$.

c) $T(n) = 2T(\sqrt{n}) + \log n$.

d) $T(n) = T(n/3) + T(2n/3) + n$

Ans:

a) $T(n) = 2T(n/2) + 1$.

$a = 2, \quad b = 2, \quad f(n) = 1$.

$n^{\log_b a} = n^{\log_2 2} = n$.

$f(n) = 1$.

$\therefore n^{\log_b a} > f(n) \quad$ for $n > 1$

$\therefore T(n) = \Theta(n)$.

Sub. method:-

Guess: $T(n) \leq c \cdot n$

$\therefore T(n) \leq 2 \cdot \frac{cn}{2} + 1$

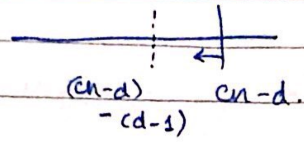
$T(n) \leq cn + 1$. [But $\nleq cn$]

Change: $T(n) \leq cn - d \leq cn$

$\therefore T(n) \leq 2\left(\frac{cn}{2} - d\right) + 1$.

$T(n) \leq cn - 2d + 1$.

$\leq (cn - d) - (d - 1)$.



for $d > 1$.

$\therefore$, $T(n) \leq cn-d$.

$T(n) \leq cn$.

Base case ~~T(n) = T(1)~~.

$T(2) = 2 \cdot T(1) + 1 = 3$.

$T(2) \leq 2c - 1$.

$3 \leq 2c - 1$.

$2 \leq c$ for $n_0 \geq 2$.

b). $T(n) = 4T(n/2) + 100n$.

$a = 4$, $b = 2$. $n^{\log_b a}$ $n^{\log_2 4} = n^2$.

Guess $O(n) = n^2$.

ie: $T(n) \leq cn^2 - dn$.

$T(n) \leq 4c\frac{n^2}{4} + 100n$. $T(n) \leq 4\left(c\frac{n^2}{4} - dn\right) + 100n$

$\leq cn^2 - 4d + 100n$

$T(n) \leq cn^2 + 100n$. $\leq cn^2 - (4d - 100)n$

$\therefore$, $\leq cn^2 - (4d - 100)n$ $\therefore$, $d \geq 25$.

$n_0 \geq 2$

$4c\left(\frac{n}{2}\right)^2 - d\left(\frac{4n}{2} + 100n\right)$

$\leq cn^2 - 2dn - 100dn$

$\leq cn^2 - dn - n(d - 100)$ $d \geq 100$.

$T(n) = 4T(n/2) + 100n.$

$\leq 4\left[c\left(\frac{n}{2}\right)^2 - \frac{dn}{2}\right] + 100n$

$\leq cn^2 - 2dn + 100n$

$\leq (cn^2 - dn) - n(d - 100) \qquad d \geq 100.$

Base: $T(2) = 4T(1) + 100(2) = 204$

$c(2)^2 - d(2) = 4c^2 - 200.$

$204 \leq 4c^2 - 200.$

$404 \leq 4c^2.$

$c^2 \geq 101$

c). $T(n) = 2T(\sqrt{n}) + \log n.$

Let $\log n = m.$

$n = 2^m.$

$\therefore T(2^m) = 2T(2^{m/2}) + m.$

$S(m) = 2S(m/2) + m \quad = m \log m$

$T(2^m) = S(m) = m \log m = \underline{\underline{\log n(\log(\log n))}}$

d). $T(n) = T(\lfloor n/3 \rfloor) + T(\lfloor \frac{2n}{3} \rfloor) + n$

Guess :- $T(n) \leq c.n \log n$

$T(0) = T(1) = 1.$

Base case checking :-

$T(0) = 1 \qquad 1 \leq 0 \qquad \times$

$T(1) = 1 \qquad 1 \leq 0 \qquad \times$

$T(2) = 1 + 1 + 2 = 4 \leq c.2 \qquad 2 \leq c.$

$T(3) = 1 + 4 + 3 = 8 \leq c.3 \log_2 3$

$\frac{8}{3 \log 3} \leq c$

Scanned with CamScanner

Step 1: Assume $\left.\begin{array}{l} 2 \le k \le n-1 \\ T(k) \le ck\log k \end{array}\right]$

$T(n) \le n\log n \leftarrow$ To prove.

$$T(n) \le c\frac{n}{3}\log\frac{n}{3} + c\frac{2n}{3}\log\frac{2n}{3} + n.$$

$$\le \frac{cn}{3}\log n - \frac{cn}{3}\log 3 + \frac{2nc}{3}(\log 2 + \log n - \log 3) + n.$$

$$\le \cancel{\frac{cn}{3}\log n}\; cn\log n - cn\log 3 + \frac{2cn}{3}\log 2 + n$$

$$\le cn\log n - cn\log 3 + \frac{2cn}{3} + n$$

$$\le cn\log n - cn\log 3 + n\left[\frac{2c}{3} + 1\right].$$

$$\le cn\log n - n\left(c\log 3 - \frac{2c}{3} - 1\right)$$

$$c\log 3 - \frac{2c}{3} - 1 \ge 0$$

$$c \ge \frac{1}{\left(\log 3 - \frac{2}{3}\right)}.$$

Check all base cases.

And claim a value of $c \ge \max(\text{all})$.

QUEUE :

Enqueue : Adding an element

Dequeue : Deleting an el.

Determining the size : $(N-f+r) \bmod N$.

```
void enq
    if ( size == N-1)
            full.
  else      Q[r] ← 0.
            r ← (r+1) mod N.
```