# Pivot in Sorted Rotated

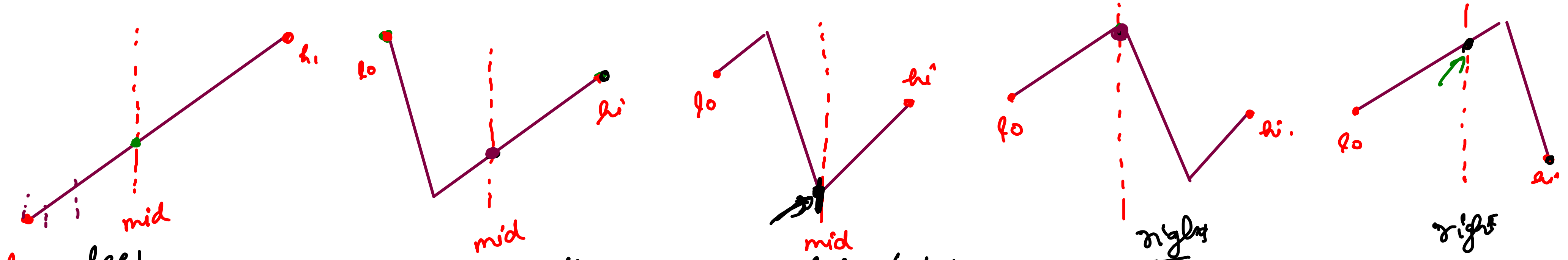
arr → {30 40 50 60 10 20}

pivot - smallest element

allowed complexity - log(n)

arr - {10, 20, 30, 40, 50}

k = 0 | k = 1 | k = 2 | k = 3 | k = 4
---|---|---|---|---
10 20 30 40 50 | 50 10 20 30 40 | 40 50 10 20 30 | 30 40 50 10 20 | 20 30 40 50 10




left | left | left + include mid | right | right
---|---|---|---|---

arr[hi] > arr [mid]

pivot → left mid,

right → mid + 1 to hi

```java
public static int findPivotEle(int[] arr, int lo, int hi) {
    int mid = lo + (hi - lo) / 2;
    int res = 0;
    if(arr[mid] < arr[hi]) {
        // left side -> including mid
        res = findPivotEle(arr, lo, mid);
    } else {
        // right side
        res = findPivotEle(arr, mid + 1, hi);
    }
    return res;
}
```

lo = hi ⇒ pivot element

arr, 4, 4

lefcall

mid = 4

arr, 4, 5

left side.

mid = 5

arr, 4, 6

right call

mid = 3

arr, 0, 6

arr

| 40 | 50 | 60 | 70 | 10 | 20 | 30 |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

lo = = mid return arr[lo];

| Time Compl. | Reoccurence Rel | Example. |
|---|---|---|
| $O(1)$ | — | Arithmatic operation. +, -, =, ==, if |
| $O(\sqrt{n})$ | — | Prime Number |
| $O(\log n)$ | $T(n) = T(n/2) + K$ | Binary Search, smart power |
| $O(n)$ | $T(n) = T(n-1) + K$<br>$T(n) = 2T(n/2) + K$ | Normal power, fake smart power |
| $O(n \log n)$ | $T(n) = 2T(n/2) + n + K$ | MergeSort, quicksort |
| $O(n^2)$ | $T(n) = T(n-1) + n + K$ | Bubble, Selection, Insertion |
| $O(2^n)$ | $T(n) = 2T(n-1) + K$<br>$T(n) = T(n-1) + T(n-2) + K$ | Subseq, fibonacci |

## Analysis for Logn

Solve functional Equation

$n \rightarrow n/2$

$n \rightarrow n/2$

$n \rightarrow n/2$

$$\cancel{T(n)} = \cancel{T(n/2)} + k \text{ ——— } (1)$$

$$\cancel{T(n/2)} = \cancel{T(n/2^2)} + k \text{ ——— } (2)$$

$$\cancel{T(n/2^2)} = \cancel{T(n/2^3)} + k \text{ ——— } (3)$$

$$\cancel{T(n/2^3)} = \cancel{T(n/2^4)} + k \text{ ——— } (4)$$

$$\vdots \qquad \vdots \qquad \vdots$$

$x^{th}$ term.

$$\cancel{T(n/2^{x-1})} = \underbrace{T(n/2^{x})}_{=T(1)} + K$$

add all equations

$$T(n) = T(n/2^x) + Kx \longrightarrow (5)$$

$$\underline{T(1) = 1}$$

$$T(n/2^x) = T(1)$$

$$\Rightarrow \frac{n}{2^x} = 1 \quad \Rightarrow \quad n = 2^x$$

take $\log_2$ both side.

$$\log_2 n = \log_2 2^x$$

$$\Rightarrow \boxed{x = \log_2 n}$$

put value of 'x' in eq (5)

$$T(n) = \underbrace{T(n/2^x)}_{T(1)} + kx.$$

$$T(n) = 1 + k \log_2 n.$$

$$\boxed{T(n) = O(\log_2 n)}$$

Properties of log

- $\log_a a = 1$
- $\log_a a^b = b.$
- $\log(m \times n) = \log m + \log n.$
- $\log(m/n) = \log m - \log n.$
- $a^{\log_a b} = b$

# Power

## Normal Power

```
if(n==0)
    return 1;

xn1 = Power(x,n-1);
xn = xn1 * x;
return xn;
```

$T(n) = T(n-1) + K$

## Smart Power

$T(n) = T(n/2) + K$

```
if(n==0) return 1;

halfP = Power(x, n/2);

xn = halfP * halfP;
if(n%2==1) {
    xn *= x;
}
return xn;
```

$O(\log n)$

## Fake Smart

$T(n) = T(n/2) + T(n/2) + K$

```
if(n==0) return 1;

half1 = Power(x, n/2);
half2 = Power(x, n/2);
xn = half1 * half2;
if(n%2 == 1) {
    xn *= x;
}
return xn
```

## Normal Power

$$T(n) = \cancel{T(n-1)} + K \quad \text{——— } 1$$

$n \to n-1 \qquad \cancel{T(n-1)} = \cancel{T(n-2)} + K \quad \text{——— } 2$

$n \to n-1 \qquad \cancel{T(n-2)} = \cancel{T(n-3)} + K \quad \text{——— } 3$

$n \to n-1 \qquad \cancel{T(n-3)} = \cancel{T(n-4)} + K$

$\vdots \qquad \vdots \qquad \vdots \qquad \Big\}\ x$ time

$$\cancel{T(n-(x-1))} = T(\underbrace{n-x}_{\to T(1)}) + K$$

add all $eq^n$.

$$T(n) = T(n-x) + Kx \text{ — } eq\ 5$$

$T(n-x) = T(1)$

$\Rightarrow \quad n - x = 1$

$\boxed{x = n-1}$ put $x$ in $eq\ 5$

$$T(n) = T(n-x) + kx.$$

$$T(n) = 1 + (n-1)K$$

$$T(n) = nK + K' \quad \Rightarrow \quad \boxed{T(n) = O(n)}$$

# Fake Smart Power →

$$T(n) = T(n/2) + T(n/2) + K$$

$$T(n) = \cancel{2T(n/2)} + k \quad \text{——— } 1$$

eq① $n \to n/2$, eq×2 $\quad \cancel{2T(n/2)} = \cancel{2^2 T(n/2^2)} + 2k \quad \text{←—— } 2$

eq② $n \to n/2$, eq×2 $\quad \cancel{2^2 T(n/2^2)} = \cancel{2^3 T(n/2^3)} + 2^2 k \quad \text{——— } 3$

eq③ $n \to n/2$, eq×2 $\quad \cancel{2^3 T(n/2^3)} = \cancel{2^4 T(n/2^4)} + 2^3 k \quad \text{——— } 4$

$\vdots \qquad \vdots \qquad \vdots$ $\quad$ x times.

$$\cancel{2^{x-1} T(n/2^{x-1})} = 2^x T(n/2^x) + 2^{x-1} k$$

add all eq$^n$

$$T(n) = 2^x \underbrace{T(n/2^x)}_{T(1) = 1} + k + 2k + 2^2 k + \dots 2^{x-1} k$$

$T(n/2^x) = T(1) \Rightarrow \frac{n}{2^x} = 1$, $\quad n = 2^x$

take log. both side,

$\log n = \log 2^x$

$$\boxed{x = \log n}$$

$$T(n) = 2^{\log_2 n} T(1) + k(1 + 2 + 2^2 + 2^3 + \dots + 2^{x-1})$$

$a = 1$, $r = 2$

G.P → $S_n = a + ar + ar^2 + ar^3 + \dots + ar^{n-1}$

$S_n = \frac{a(r^n - 1)}{r - 1}$ $\quad$ sum of G.P.

$$T(n) = n \times 1 + k\left[\frac{1(2^x - 1)}{2 - 1}\right]$$

$$= n + k(2^{\log_2 n} - 1)$$

$$= n + k(n - 1)$$

$T(n) = n(k+1) - k \Rightarrow$ $\quad \boxed{T(n) = O(n)}$

$T(n) = nk^1 + c$

# Complexity Analysis for MergeSort – n log n

$T(n) = T(n/2) + T(n/2) + n + k$

$\Rightarrow \quad T(n) = \cancel{2^1 T(n/2)} + n + k$ ——— eq (1)

$\cancel{2T(n/2)} = \cancel{2^2 T(n/2^2)} + 2n/2 + 2k$ ——— eq (2)

$\cancel{2^2 T(n/2^2)} = \cancel{2^3 T(n/2^3)} + 2^2 n/2^2 + 2^2 k$ ——— eq (3)

$\cancel{2^3 T(n/2^3)} = \cancel{2^4 T(n/2^4)} + 2^3 n/2^3 + 2^3 k$ ——— eq (4)

$\vdots \quad \vdots \quad \vdots \quad \vdots$ } $x$ times.

$\cancel{2^{x-1} T(n/2^{x-1})} = 2^x T(n/2^x) + 2^{x-1} n/2^{x-1} + 2^{x-1} k$

$T(n) = 2^x T(n/2^x) + \{ n + \frac{\cancel{2}n}{\cancel{2}} + \frac{\cancel{2^2}n}{\cancel{2^2}} + \frac{\cancel{2^3}n}{\cancel{2^3}} + \cdots \frac{\cancel{2^{x-1}}n}{\cancel{2^{x-1}}} \}$

$+ k(1 + 2 + 2^2 + 2^3 + \cdots + 2^{x-1})$ ← Sum of G.P.

$T(n/2^x) = T(1)$

$\Rightarrow \frac{n}{2^x} = 1 \Rightarrow n = 2^x$

$\boxed{x = \log n}$

$T(n) = 2^x \times 1 + nx + k\frac{(2^x - 1)}{2-1}$

$T(n) = 2^{\log n} + n \log n + k(2^{\log n} - 1)$

$T(n) = n + n\log n + kn - k$

$T(n) = n \log n + k'n + k''$

$$\boxed{T(n) = O(n \log n)}$$

quickSort → $T(n) = T(n/2) + T(n/2) + n + k$

$\Rightarrow \boxed{T(n) = 2T(n/2) + n + k} \Rightarrow O(n \log n)$

↑ for partition index

# fibonacci Recursive code + complexity Analysis ||

```
int fib(int n) {
    if( n==0 || n==1) {
        return n;
    }
    int fibnm1 = fib(n-1);   T(n-1)
    int fibnm2 = fib(n-2);   T(n-2)
    int fibn = fibnm1 + fibnm2;
    return fibn;
}
```

$$\underbrace{T(n) = T(n-1) + T(n-2)} + K$$

$$T(n) < T(n-1) + T(n-1) + K \rightarrow 1$$

$$T(n-2) + T(n-2) + K < T(n) \rightarrow 2$$

Combine eq 1 & 2

$$T(n-2) + T(n-2) + K < T(n) < T(n-1) + T(n-1) + K$$

$$2^{n-2}(K+1) - K < T(n) < 2^{n-1}(K+1) - K$$

$$\Rightarrow \quad T(n) = O(2^n)$$

# Subsequence

$T(n) = T(n-1) + T(n-1) + k$

$T(n) = \cancel{2T(n-1)} + k$

$n \to n-1 \}\, 2$ $\quad \cancel{2T(n-1)} = \cancel{2^2 T(n-2)} + 2k$

$\cancel{2^2 T(n-2)} = \cancel{2^3 T(n-3)} + 2^2 k$

$\cancel{2^3 T(n-3)} = \cancel{2^4 T(n-4)} + 2^3 k$

$\vdots \qquad \vdots \qquad \vdots \Big\}$ x times.

$\cancel{2^{x-1} T}(n-(x-1)) = 2^x T(n-x) + 2^{x-1} k$

---

$T(n) = 2^x T(n-x) + k(1 + 2 + 2^2 + \cdots + 2^{x-1})$

$T(n-x) = T(1)$

$\Rightarrow n - x = 1 \Rightarrow \boxed{x = n-1}$

$T(n) = 2^x \times 1 + k\left(\frac{2^x - 1}{2-1}\right) = 2^x + k(2^x - 1)$

$T(n) = 2^{n-1} + k(2^{n-1} - 1)$

$T(n) = \frac{2^n}{2}\underbrace{(k+1)}_{k'} \underbrace{-k}_{k''} \Rightarrow T(n) = 2^n k' + k''$

$\Rightarrow T(n) = O(2^n)$

$T(n) = T(n-1) + T(n-1) + k$

overall complexity of subseq

for yes call

for no call

# Selection Sort on Recursion. →

30 70 40 90 60 50

T(n)

```
SelectionSort(int[] arr, int indx) {
    if(indx == arr.length) return;
```

(n) → Min Indx → Swap } done by myself

```
T(n-1) Selection Sort(arr, indx+1);
}
```

Expectation → Selection Sort(arr, 0) → 0 to end shift & position it well

faith → SelectionSort(arr, 1);

Merging → 0th index, shifting by me.

bubbleSort(arr,

## Selection

$$T(n) = \cancel{T(n-1)} + n + k \Big\} \Rightarrow n \rightarrow n-1$$

$$\cancel{T(n-1)} = \cancel{T(n-2)} + n-1 + k$$

$$\cancel{T(n-2)} = \cancel{T(n-3)} + n-2 + k$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$\cancel{T(n-(x-1))} = T(n-x) + n-(x-1) + k$$

| | | | |
|---|---|---|---|
| 0 | 4 | } – 4 | $n$ |
| 1 | 4 | } 3 | $n-1$ |
| 2 | 4 | } 2 | $n-2$ |
| 3 | | } 1 | $\vdots$ |
| 4 | | } 0 | 1 |

add all eq$^n$

$$T(n) = T(n-x) + n + n-1 + \cdots + n-(x-1) + kx.$$

$T(n-x) = T(1)$

$n - x = 1$

$\boxed{x = n-1}$

$$T(n) = 1 + n + n-1 + \cdots + \cancel{n}-(\cancel{n}-1-1) + k(n-1)$$

$$= 1 + n + n-1 + \cdots + 2 + k(n-1)$$

$$= 1 + 2 + 3 + \cdots + n + k(n-1)$$

$$= \frac{n(n+1)}{2} + kn - k$$

$$T(n) = \frac{n^2}{2} + n\left(k + \frac{1}{2}\right) - k.$$

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

$$T(n) = An^2 + Bn + C$$

$$\boxed{T(n) = O(n^2)}$$

## Insertion

1

2

3

$\vdots$

$n$

$$1 + 2 + 3 + n \cdots + n = \frac{n(n+1)}{2} = O(n^2)$$

## Bubble Sort

0 — $n$    $n \rightarrow n$

0 — $n-1$    $n-1$

0 — $n-2$    $n-2$

$\vdots$    $\vdots$    $O(n^2)$

$0 \rightarrow 0$    1

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

Loop 1 for(i=1; i<n; i++) ——— O(n)

Loop 2 for(int i=1; i*i <= n; i++) ⟶ $O(\sqrt{n})$

$i \le \sqrt{n}$

loop 3 for(int i= 1; i<=n; i*=2)

$1 \to 2 \to 4 \to 8 \cdots \cdots 2^x \to 2^{x+1}$

$2^{x+1} \ge n$

$n \to n/2 \to n/4 \to n/8 \cdots\cdots 1$

⟶ Log(n)

Loop 4 for(int i=n; i>1; i=i/2) ⟶ log n

Two loop → O(n)

Loop 5 for(int i=1; i<n; i+=m){

for(int j=1; j<m; j++){

}

O(n)

i=1 ⟷ m } → m times.

i=1+m ⟷ m

i=1+2m ⟷ m

i<n

i=1+ ...

$\frac{n}{m}$

single itr = m times.

n/m itr = $\frac{n}{\cancel{m}} \times \cancel{m}$

= n

i<8 ; i+=2

2, 4, 6, 8

n/m

$\boxed{O(n)}$

i, m

i+m, m

i+2m, m

i+3m, n

n

loop 6 $j=0$

```
for(int i=1; i<=n; ){
    if(j==i){
        j=0;
        i++;
    }
    j++;
}
```

$n=5$

iteration of $j$

$i=1$ • → 1

$i=2$ • • → 2

$i=3$ • • • → 3

$i=4$ • • • • → 4

⋮

$i=5$ • • • • • → n

$$\text{Sum} = 1+2+\dots+n$$
$$= \frac{n(n+1)}{2} = O(n^2)$$

loop 7

```
int i=1; s=1;
while(s<=n){
    i++;
    s+=i;
}
```

$$S = i_1 + i_2 + i_3 + \dots + x$$
$$S = 1+2+3+\dots+x$$
$$S = \frac{x(x+1)}{2} \le n$$
$$x^2 + x - 2n \le 0$$
$$x^2 + x - 2n = 0$$

$$ax^2 + bx + c = 0$$
$$x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$$
$$x = \frac{-1 \pm \sqrt{1+8n}}{2}$$
$$x = \frac{-1 + \sqrt{1+8n}}{2}$$
$$x \propto \sqrt{n} \Rightarrow x = O(\sqrt{n})$$