May - 21

21 May 2020 19:58

For Loop I
Time Complexity (Prime Example) → Importance, How to check
Sorting
Recursion

Prime no. check

10

Ram	Sham
$n \to [2 \text{ to } n-1]$	$n \to \sqrt{n} >$
divide by every no.	$[2 \text{ to } \sqrt{n}]$
$\left[\frac{n}{2}, \frac{n}{3} — \frac{n}{n-1}\right]$	↓ (1 computation = 1 unit time)
% → 0 ✗ ✓Prime, % → 0 Not Prime	$\sqrt{n}$ computations
n → computations	

16 → $\sqrt{\cdot}$ → 4

n=10 — 10 — 3
n=100 — 100 — 10
n=10,000 — 10,000 ⟷ 100 ← Efficient

10 unit — 10 sec; 20 unit — 20 sec
10 unit — 20 sec; 20 unit — 40 sec

Time Complexity

1. Atomic/Basic Statements
↳ Unit time

$x+1$ → Addition/Subtraction
$x==0$ → true / false
$x=5$ → Assignment
S.O.P → atomic

$\boxed{x = x+1}$ T.C ?
2 units

$\boxed{i=i+1}$

2. Loops

$\boxed{\text{for(int } i=0;\ \underline{i<5};\ i{+}{+})}$
{
S.O.P(5);

i=0 ← ①

```
for(int i=0 ;         )                     1
{                                       ←
   S.O.P(5);                         i=0
                                     i<5   ← 1
                                     S.O.P ← 1
}     3
      4  [ 1  [  i++   ← 2
                 i<5   ← 1
                 S.O.P ← 1
22        2
units
      4
      4   3
      4   4'
      3 ←        i++  ← 2
                 i<5     1
```

3 Notations

1 Big-Oh Notation 'O'

Worst Case Complexity

Constant time → O(1)

10 units → O(1)

```
for(int i=0; i<n; i++)
{     S.O.P("Hello");
}
```

```
   i=0
   i<n       ] → 3
   S.O
2 ← i= i+1
1 ← i<n      ] → 4 → (n-1)
1 ← S op
   i++
   i<n       ] → 3
```

$\rightarrow 3 + 4(n-1) + 3$

$\Rightarrow 3 + 4n - 4 + 3 = 4n + 6 - 4$

$= \boxed{4n+2}$

$\Rightarrow 3+4n-4+3 = 4n+6-4$
$= \boxed{4n+2}$

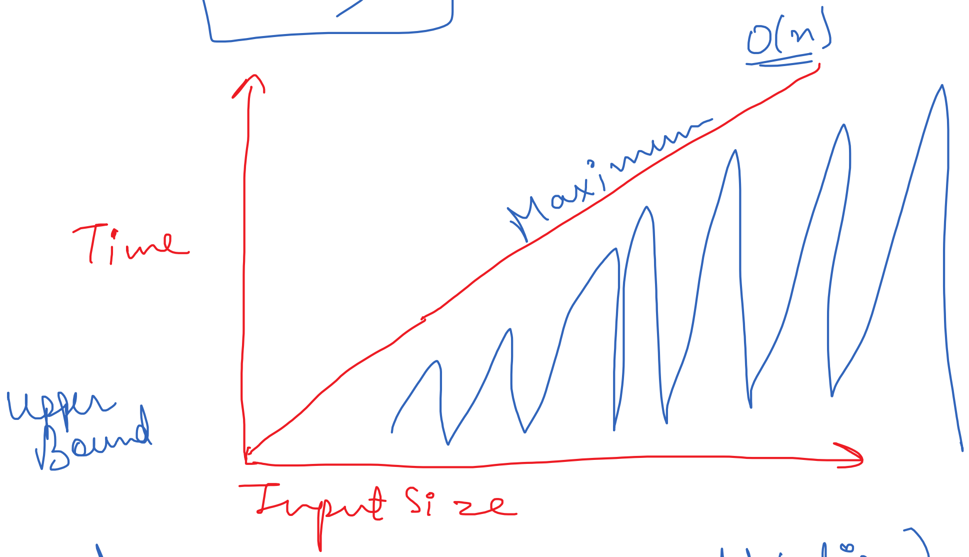
$4n+2 \Rightarrow O(4n+2)$

$O \rightarrow$ Worst Case

$\boxed{n \rightarrow \infty}$

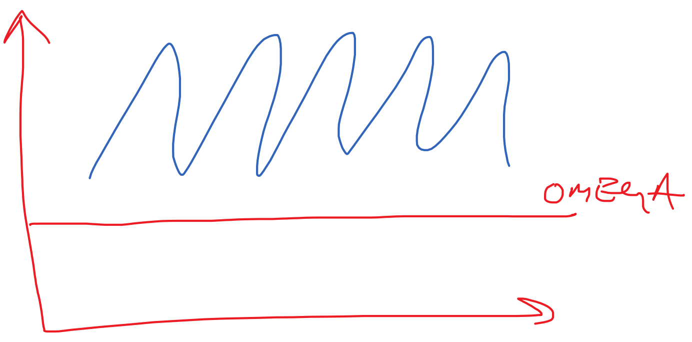
$\underline{4n} \; \cancel{+2}$

$\boxed{O(n)}$ $\boxed{\text{Code} \propto \text{size of input}}$



2nd Ω (OMEGA Notation)
(Lower Bound)

$\boxed{f(n) > C \times g(n)} \rightarrow$ OMEGA Notation

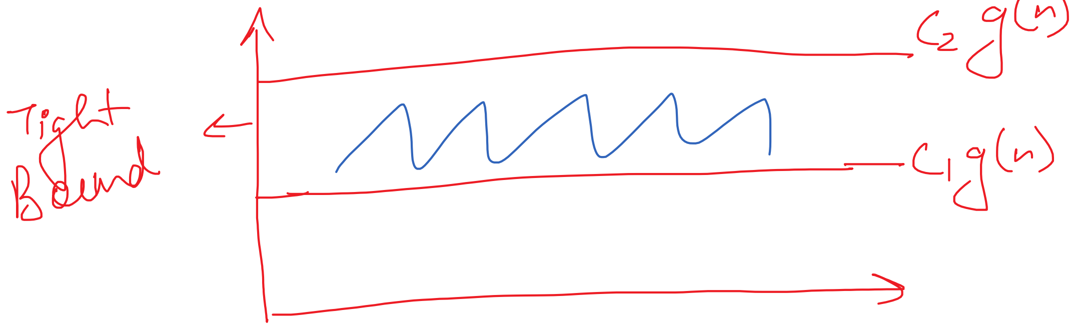


3 Θ-Notation

$c_1 \times g(n) < f(n) < c_2 \times g(n)$

$g(n)$

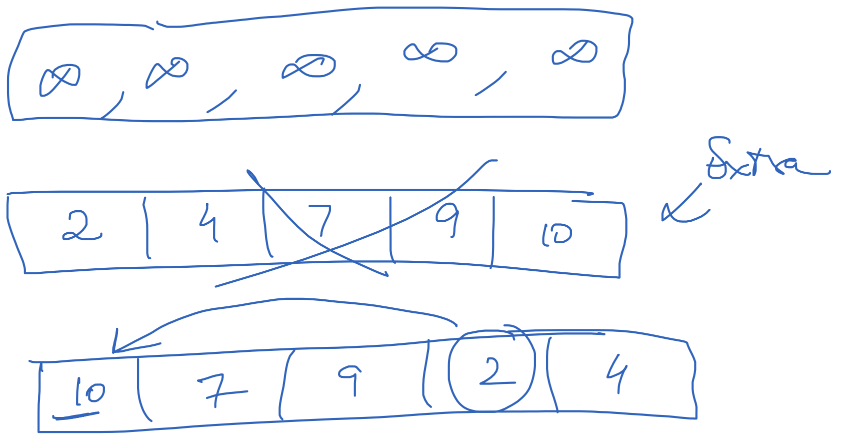
$c_1 \times g(n)$



Sorting → Time Complexity
→ Space Complexity → In place?

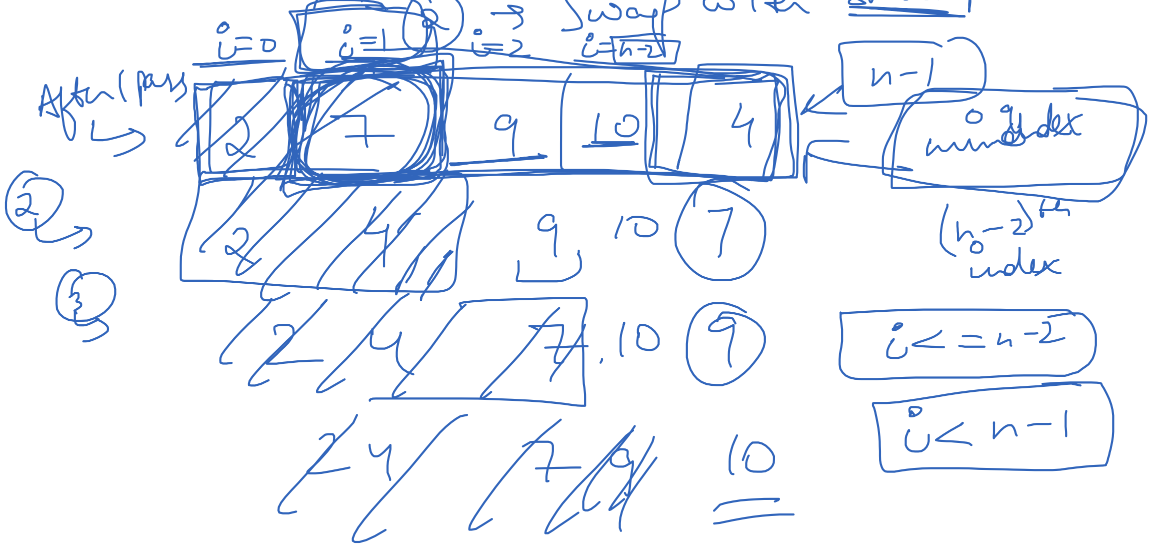
given [1 | 2 | 3 | 4 | 5]
create [| | | |] ← Extra space

① Selection Sort



① → Find Min index

② → Swap with start pos



$i <= n-2$

$i < n-1$

② Bubble Sort

[10, 7, 9, 2, 4]

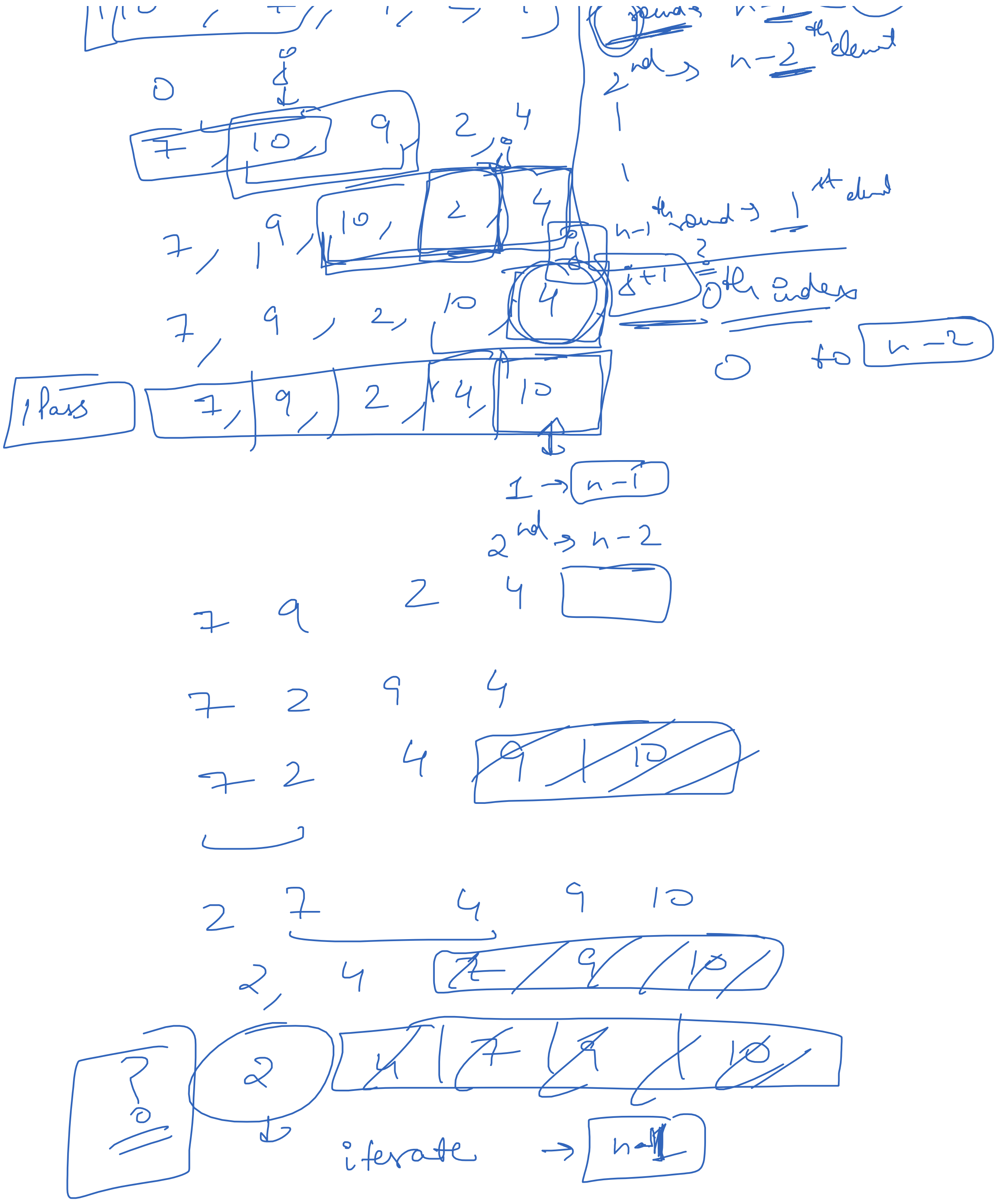
i

4 passes

last index

1st round → n-1 ← j

2nd → n-2 th element



Recursion

↳ A function calling itself directly / indirectly.

Direct	Indirect
void fun(..)	void A()
	{

```
void fun(.)        → Base Case()
{
    fun();
}
```

```
void A()
{
    B();      Base Case ( )
}
void B()
{
    A();
}
```

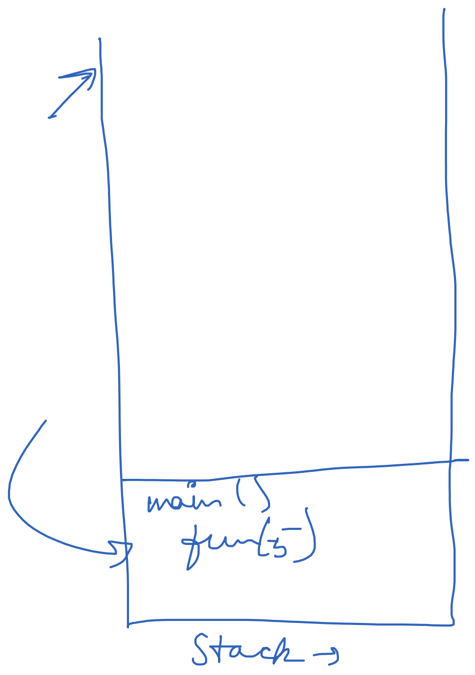
$\boxed{\text{Base Case / Terminating Condition}}$

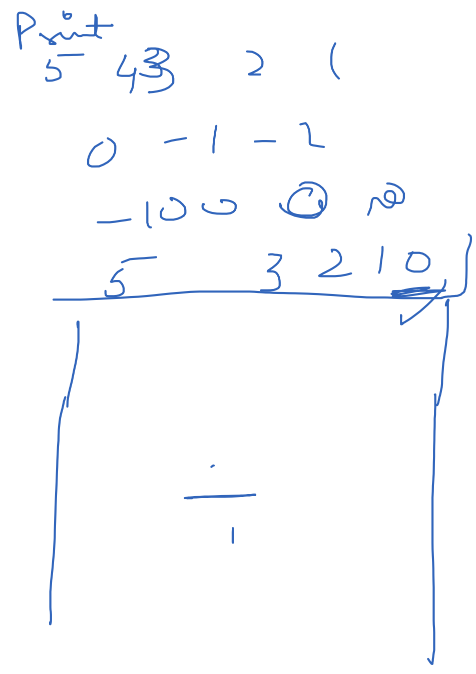
Example

```
void fun(int n)
{

    S.O.P(n);
    fun(n-1);

}
```





1 → Your Assignment
2 → Revise everything
3 → $\boxed{\text{Types of Recursion}}$ *
4 → Print (0 to 5) in Ascending Order using recursion. And the first function call must be fun(5);