

Space Complexity

Space complexity is space required as a function (f(n)) of input size

$$\text{Space complexity} = \text{input space} + \text{auxiliary space}$$

OS 1tb - 50gb

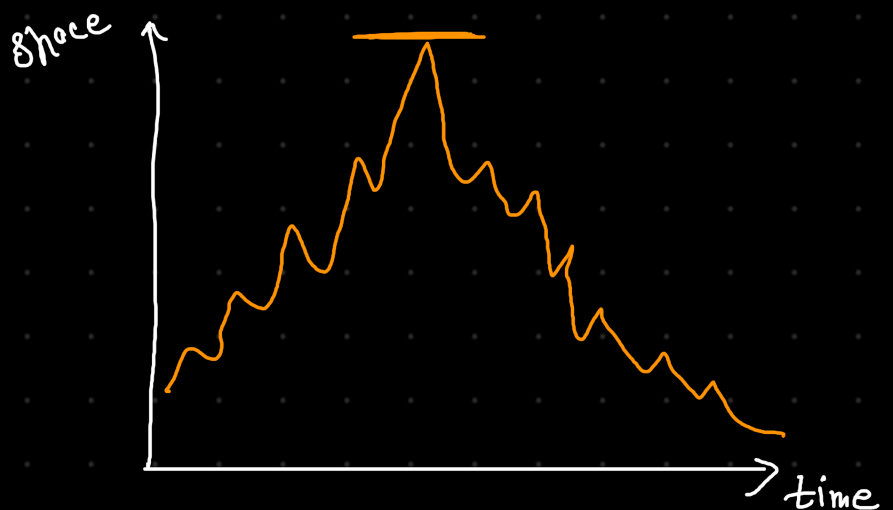
bubble sort
insertion

* we don't consider variables created as a extra space.

merge

sorted array = [] dependent on input size.

	8gb		
⇒ install	2gb	}	+
install	2gb		+
delete	4gb	}	-
install	<u>6gb</u>		+



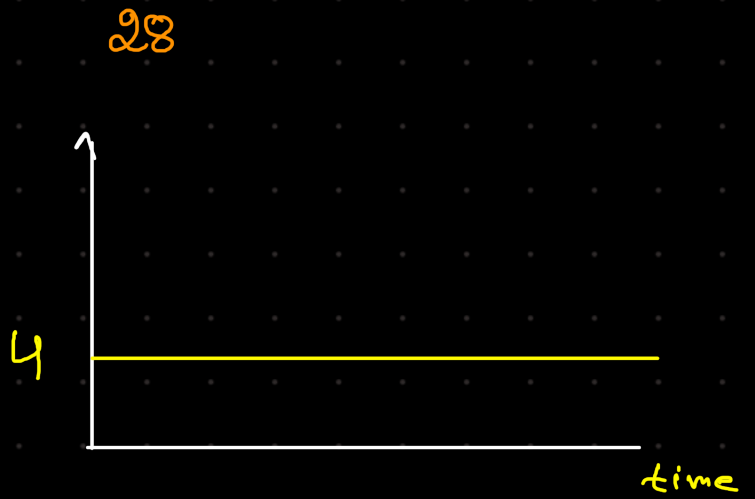
$$n = 10000$$

while ($i \leq n$):

$$\underline{a} = 5$$

$$a_t = 1$$

28 0000

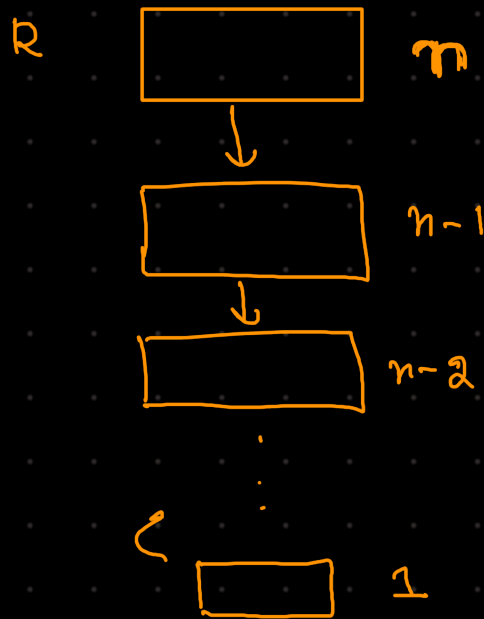


Space Complexity of algorithms

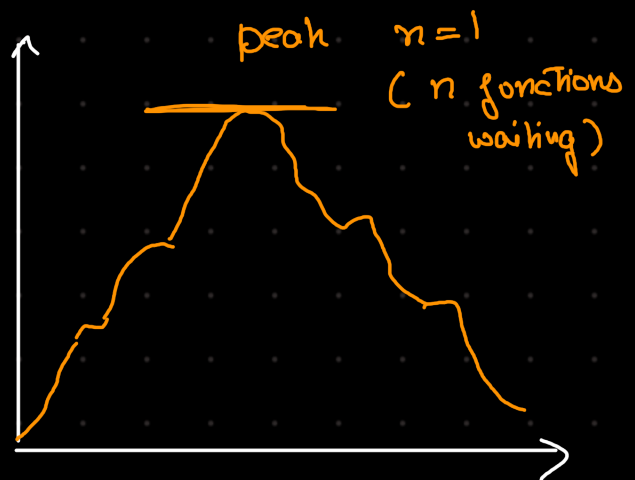
- Insertion sort

Space complexity - recursive algorithm

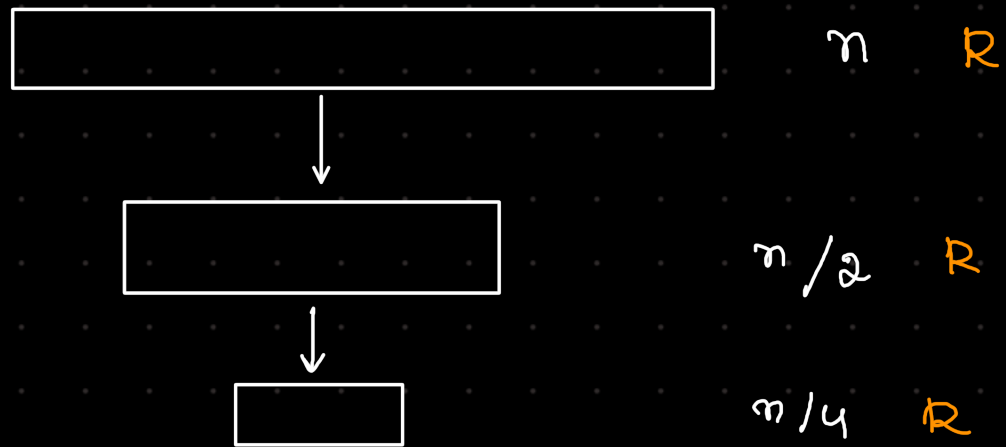
```
def fact (n):  
    if (n <= 1)  
        return 1  
  
    return fact(n-1) * n
```



recursion is not free, the functions waiting for answers, takes up some space.



Binary Search (Recursion)

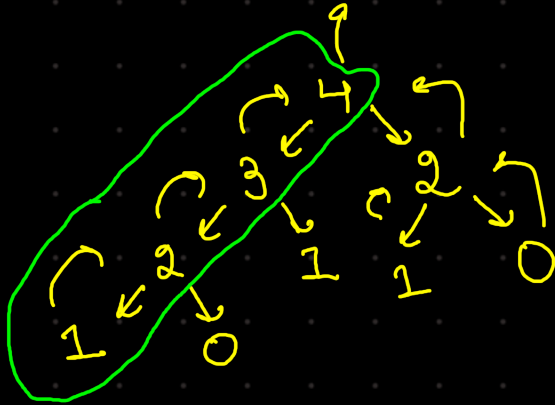
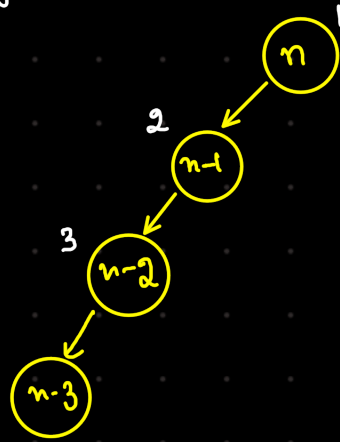


$$R \sim (\text{no. of functions}) \Rightarrow R \underline{\log_2 n}$$

we will have $\log_2 n$ functions waiting.

Fibonacci Number : Space Complexity

```
def fib(n):  
    if (n ≤ 1)  
        return 1  
    else fib(n-1) + fib(n-2)
```

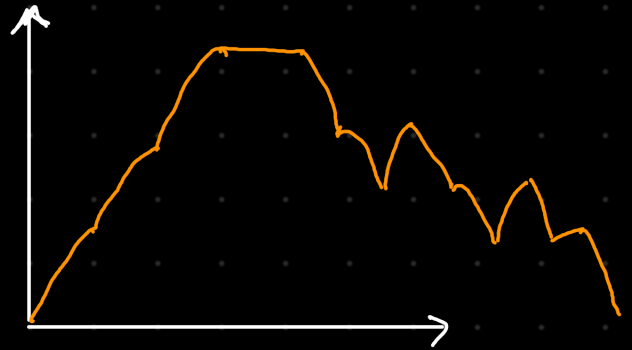


max^m number
of calls in memory

$$nR \leq (R+1)n$$

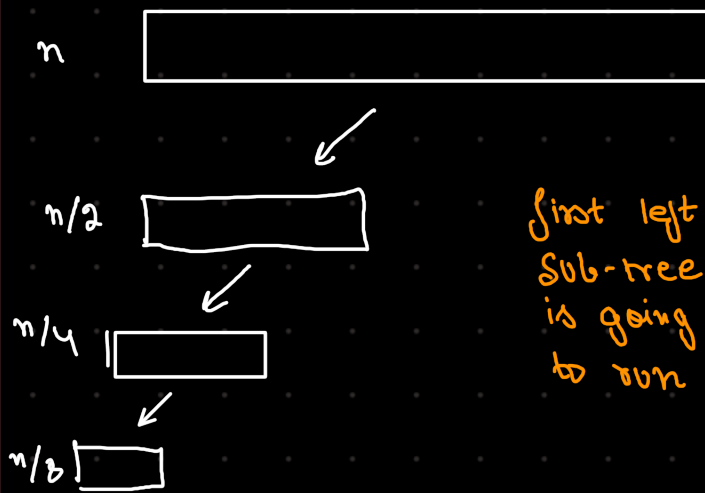
\downarrow
 $g(n)$

$$O(g(n))$$



Space complexity is $O(n)$

Merge sort : Space complexity

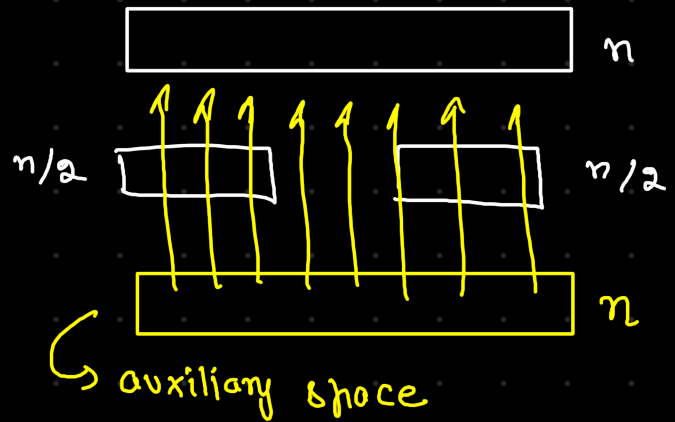


$\log_2 n$ function
at max waiting
in the memory.

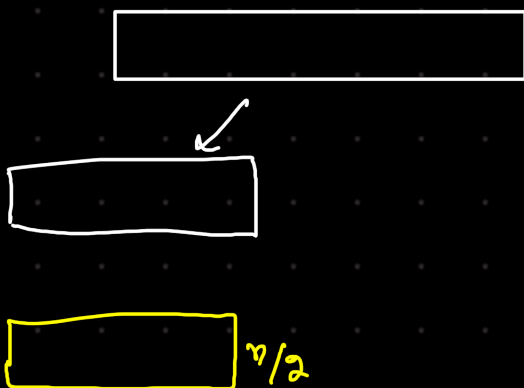
def merge Sort (arr):

$\log_2 n$ { merge sort (left)
merge sort (right)

n { merge (left, right)



Space complexity $O(\log_2 n + n)$



for a very large n

n \gg $\log_2 n$
major term

My major term is n

Space complexity $\rightarrow Rn$

$O(n)$ is the space complexity of merge sort
array of n size

