Asymptotic Analysis

* O → worst case → upper Bound
* Ω → Best case → lower Bound
* Θ → avg case → tight Bound

$O = \Omega$

$f(n) \in O(n)$ if there are positive constants (c) & n_0 $0 \leq f(n) \leq cO(n)$ $\forall$ $n \geq n_0$

* $f(n) = 15n + 93$

prove that $f(n) \in O(n)$ ← $O(n)$

$0 \leq f(n) \leq cn$

$15n + 93 \leq cn$

assume $n = 93$ →

$15n + n \leq cn$

$16n \leq cn$ ⟹ $c = 16$, $n_0 = 93$

$g(n) = 2n^2 + 1 \in O(n^2)$

$2n^2 + 1 \leq cn^2$

assume $n = 1$

$2n^2 + n^2 \leq cn^2$

$3n^2 \leq cn^2 \rightarrow c = 3,\ n = 1$

$g(n) \in O(n^2)$

$T(n) = 2n^2 + 1 \in \Omega(n^2)$

$\Rightarrow 2n^2 + 1 \geq cn^2$

Positive

$2 + 1 \geq c \rightarrow c = 3$ $\quad n \geq 1$

$\rightarrow 2n^2 + 1 \geq 3n^2$

$1 \geq n^2$

$\Theta(n^2)$

Order of Growth

$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2)$

$\Rightarrow < O(n^3) < O(2^n) < O(n^n)$

⇒ arr = [1, 2, 3] } ⟹ 1 → O(1)
x = arr[0]

Linear search

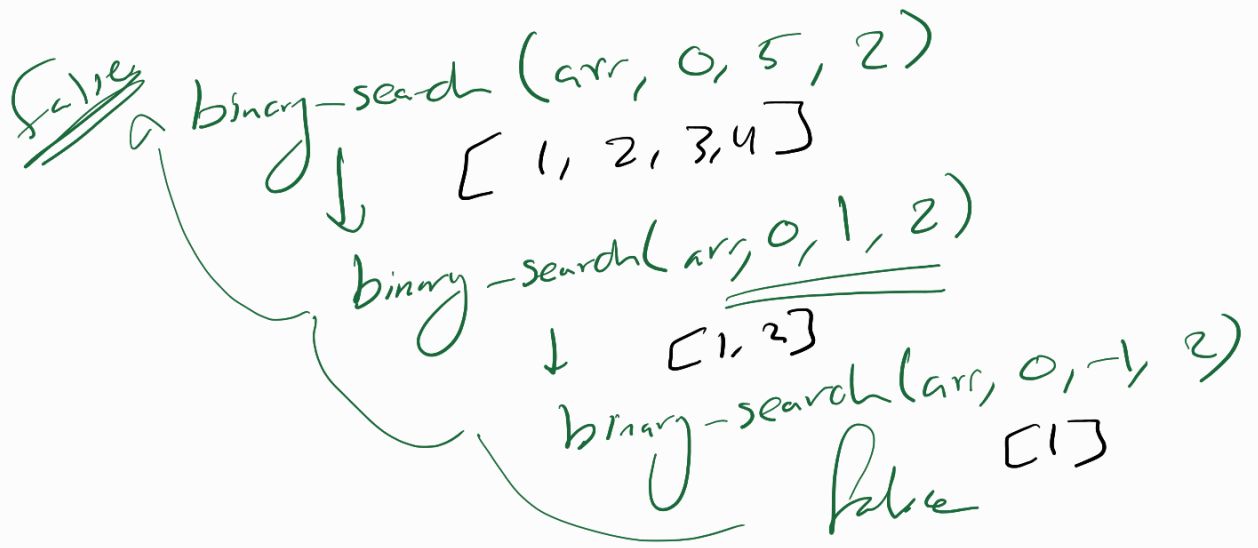
```
def find(x, arr):
    for i in arr:
        if i == x:
            return True
    return False
```

(5) [5, 7, 6]

O(n)

[5, 8, 16, 22, 23, 26, ----, 1000000, ---, m_n] ⟹ size of array = n

```
def binary_search(arr, l, r, x):
    if l > r:
        return false
    m = (l + r) / 2
    if arr[m] == x:
        return True
    elif arr[m] > x:
        return binary_search(arr, l, m-1, x)
    else:
        return binary_search(arr, m+1, r, x)
```



8	$\frac{8}{1}$	$\frac{8}{2^0}$	$\frac{n}{2^0}$
4	$\frac{8}{2}$	$\frac{8}{2^1}$	$\frac{n}{2^1}$
2	$\frac{8}{4}$	$\frac{8}{2^2}$	$\frac{n}{2^2}$
1	$\frac{8}{8}$	$\frac{8}{2^3}$	$\frac{n}{2^3}$

$2^3 = n$

$\log_2 8 = 3$

$O(\log_2 n)$

upper bound

```
def f(arr):                    # arr of size n
    t = 0
    for x in arr:              # n *
        l = len(arr)
        while l > 1:           # log n
            l = l // 2
            t += 1
    return t
```

$O(n * \log n)$

```
def g(arr):
    for i in range(len(arr)):
        for j in range(i)
            if arr[i] > arr[j]:
                arr[i], arr[j] = arr[j], arr[i]
```

$O(n^2)$

i	j	
1	1	1
2	1, 2	2
3	1, 2, 3	3
⋮		
n	1, 2, ..., n	n

$1+2+3+4+\cdots+n$

$\frac{n(n+1)}{2}$

$\frac{n^2+n}{2}$

$\frac{1}{2}n^2+\frac{1}{2}n$

$\Rightarrow O(n^2)$

---

arr = [[1, 2, 3], [1, 2, 3]] (i = 0, j = 0)

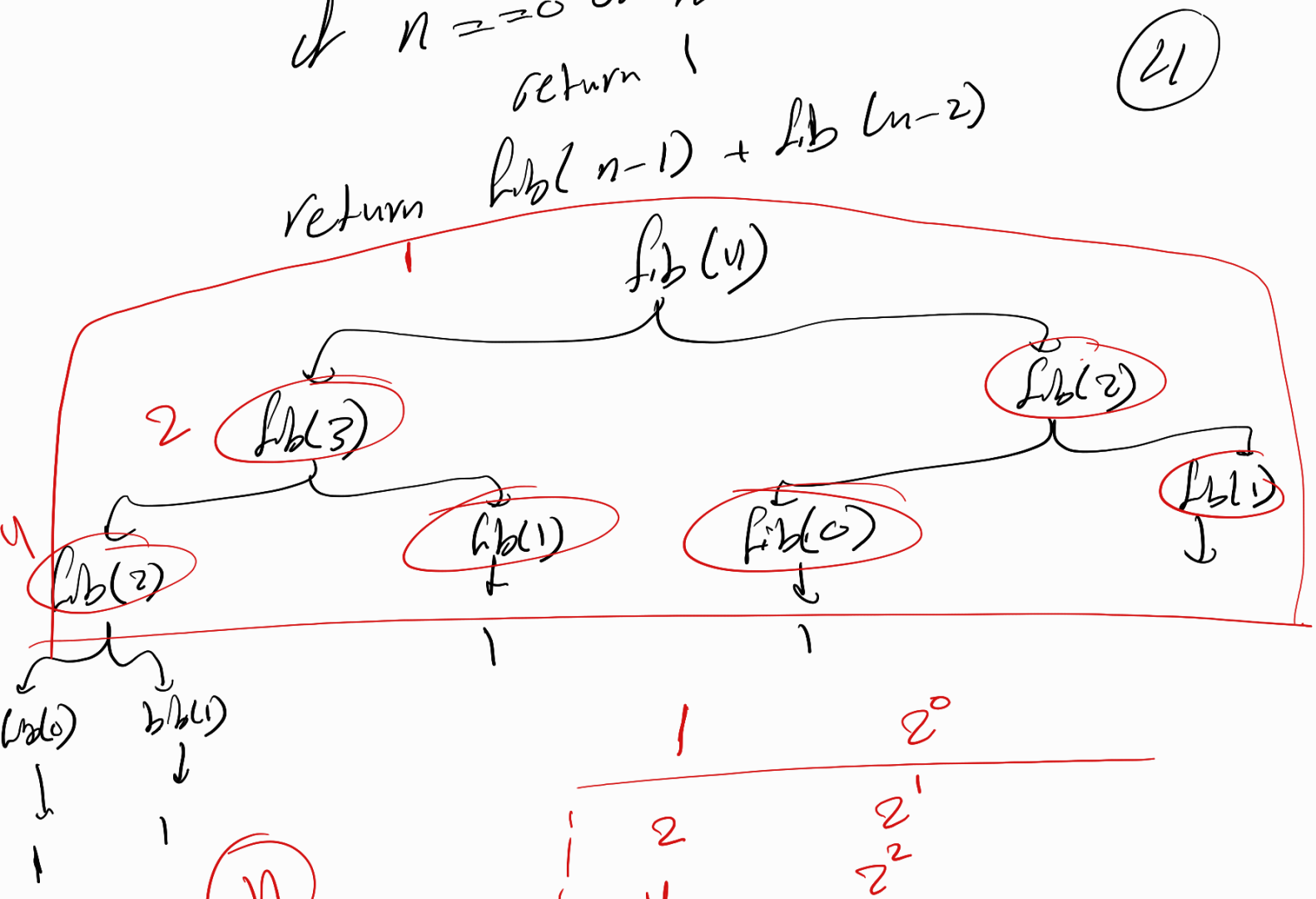
arr[0][0]

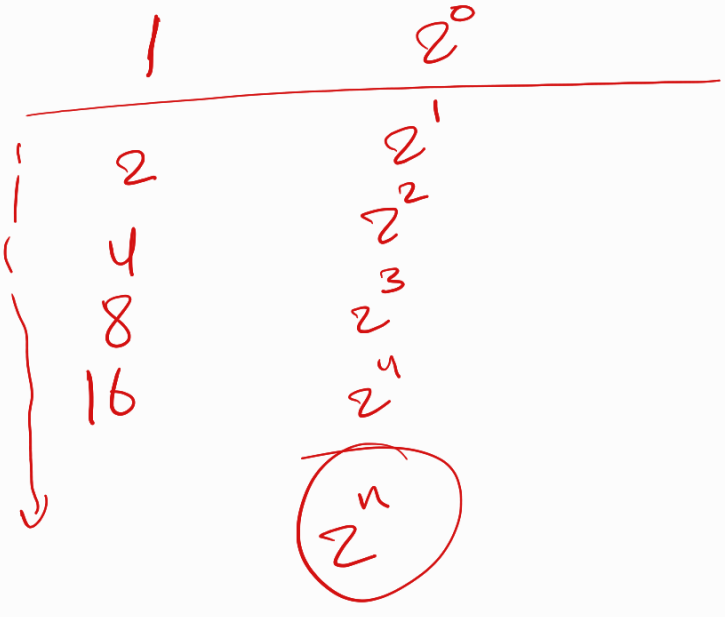
```
def bubble(matrix):
    n ← for i in range(len(matrix)):
    n ← for j in range(len(matrix)):
    n ← for a in range(len(matrix[0])):
    n ← for b in range(len(matrix[0])):
```

$O(n^4)$

if matrix[i][a] < matrix[j][b]:
matrix[i][a], matrix[j][b] = matrix[j][b], matrix[i][a]

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    return fib(n-1) + fib(n-2)
```





$O(2^n)$