Prob: Show that if $f \in O(g)$ & $g \in O(h)$, then $f \in O(h)$.

$$f \le c_1 g$$
$$g \le c_2 h$$
$$f \le \underbrace{c_1 c_2}_{c} h \implies f \in O(h)$$

Prob: Show that $10n^2 + 7 \neq O(n)$

'O' $\quad \lim_{n \to \infty} \frac{f(n)}{g(n)} = \text{constant}$

or $\quad$ Contradiction method

$$10n^2 + 7 \le cn$$
$$\implies \underbrace{10n + \frac{7}{n}}_{\text{increases}} \le c$$

$$2n + 3 \neq o(1)$$

Theorem: If $f(n) = a_m n^m + a_{m-1} n^{m-1} + \cdots + a_1 n + a_0$ & $a_m > 0$, then $f(n) = \Theta(n^m)$

Proof: $f(n) = O(n^m)$
$f(n) = \Omega(n^m)$

$$c = \sum_{i=0}^{m} |a_i|$$

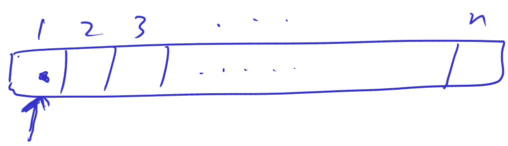
$$c\,n^m \ge f(n)$$
$$c'\,n^m \le f(n)$$

Prob: $\boxed{\Theta(n \log n) \text{ time algo}}$ — n integers — 'S'
— another integer X

to determine whether or not there exist two elements in S, whose sum is exactly X.

Soln

$$S \xrightarrow[\;]{\Theta(n\log n)\; \text{ms}} S'$$

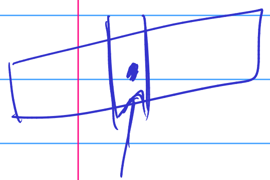
S' 

BS : I/P sorted

BS

$O(\log n)$

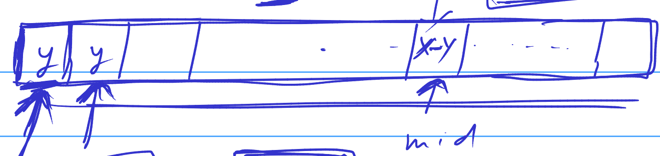
$\Omega(1)$



$\boxed{n \log n}$ ms : ——— Phase I

$\downarrow S'$

$\Theta(n \log n)$



$\boxed{x}$ $\boxed{x-y}$

for $i = 1$ to n

$y = A[i]$

$BS(S', x-y)$

——— Phase II

$O(n \cdot \log n)$

$\Omega(\cancel{n}\,1)$

Overall running time : $\Theta(n \log n) + O(n \log n)$

$= \Theta(n \log n)$

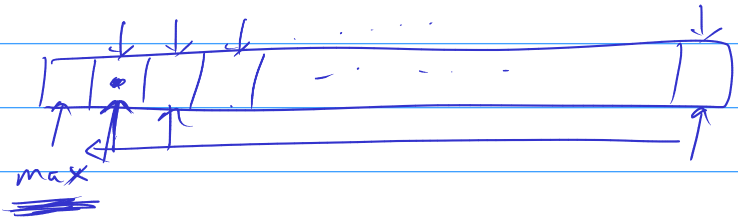
Prob: $2^{2n} \neq O(2^n)$

$0 \leq 2^{2n} \leq c \cdot 2^n$ for all $n \geq n_0$

$\Rightarrow 2^n \leq c$

Finding maximum of n elements

Straightforward method



$O(n)$

$\Omega(n)$

$\Theta(n)$

Procedure MAX(A, n) $= O(n) = \Omega(n)$

max $\leftarrow$ A[1] —— $t_n = C + C_1 n$

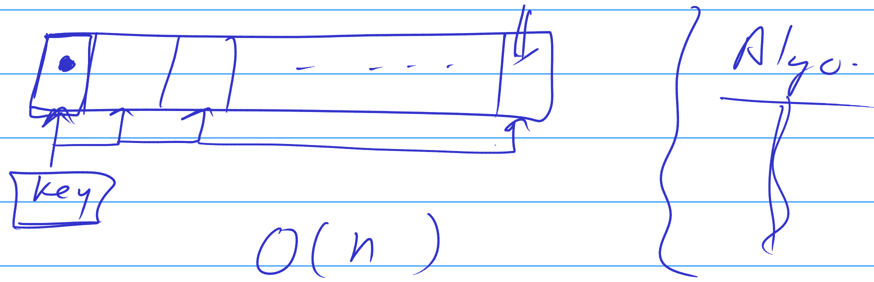
for $i = 2$ to n

if $A[i] > max$, then $max \leftarrow A[i]$

$c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \forall n \geq n_0$

$f(n) = \Theta(g(n))$

Sequential search / linear search



$O(n)$

$\Omega(1)$

~~$\Theta(\ \ )$~~

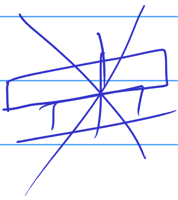
$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(k^n)$

Overall running time: $O(k^n)$

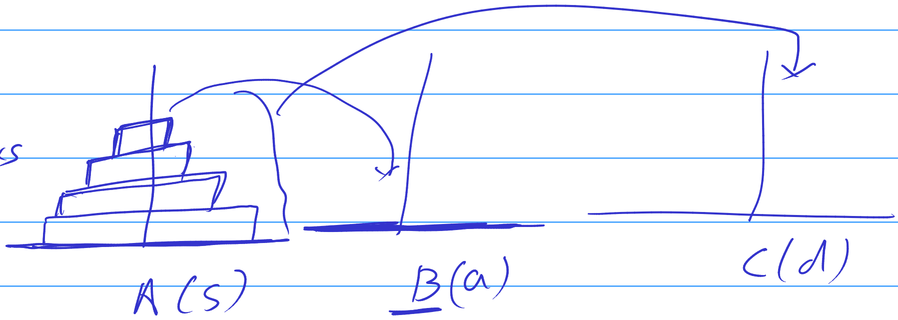
$t_n = \boxed{2^n} + \boxed{3n^3 + 2n^2 + 5}$ $\quad n \to \infty$

$= O(2^n)$

Tower of Hanoi problem:



n disks

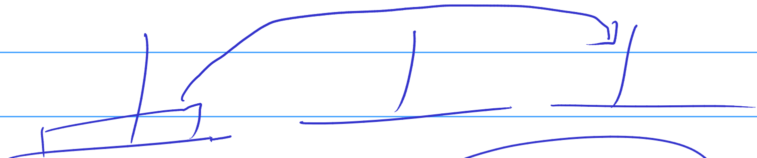


Constraints: 1) Move only one disk at a time
2) A larger disk can't be placed over a smaller disk

t_n : no. of movements

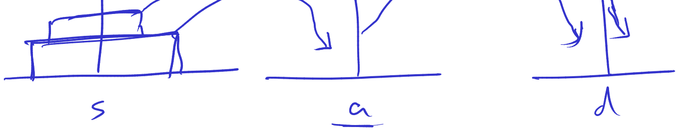
$n = 1$

$t_1 = 1$



$n = 2$

$t_2 = 1 + 1 + 1$
$= 3$



~~$a_1, a_2, a_n \ldots a_{n+1}, a_n$~~

$t_1 = 1,\ t_2 = 3,\ \ldots,\ t_n = ?$

$\boxed{t_n} = 2t_{n-1} + 1 = 2^n - 1$

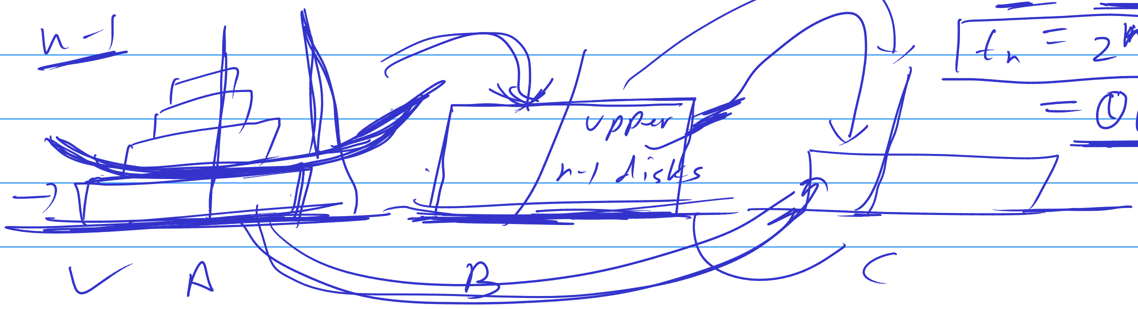
$= f(n)$

$t_n = 2t_{n-1} + 1$

$\boxed{t_n = 2^n - 1}$

$= O(2^n)$

$\boxed{X}$



Recursive way of solving

$t_n = t_{n-1} + 1 + t_{n-1}$

$\boxed{t_n = 2t_{n-1} + 1} \to$ recurrence relation

$t_0 = 0,\ t_1 = 1,\ t_2 = 3 \to$ initial conditions

$f_n = f_{n-1} + f_{n-2}$

~~implic~~ explicit expression $t_n = f(n)$

Next experiment
→ Recursive algo
→ Non-recursive of tower of Hanoi algo/problem

$t_n = 2t_{n-1} + 1$ —— 1)

$t_{n-1} = 2t_{n-2} + 1$ —— 2)

1) − 2)

$t_n - t_{n-1} = 2t_{n-1} - 2t_{n-2}$

$t_n - 3t_{n-1} + 2t_{n-2} = 0$ (Homogeneous RR) → in depth

$x^2 - 3x + 2 = 0$

$x = 1, 2$

r_1, r_2

r_1

~~$t_n = (c_1 + c_2 n)\cdot n$~~

$t_n = 2^n - 1$

$t_n = c_1 r_1^n + c_2 r_2^n$

$\boxed{t_n = c_1 + c_2 2^n} \to t_n = -1 + 2^n$

$\boxed{t_n = 2^n - 1}$

$t_1 = 1$
$t_2 = 3$
→
$1 = c_1 + c_2 \cdot 2$
$3 = c_1 + c_2 \cdot 4$
→ $c_1 = -1$, $c_2 = 1$