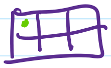
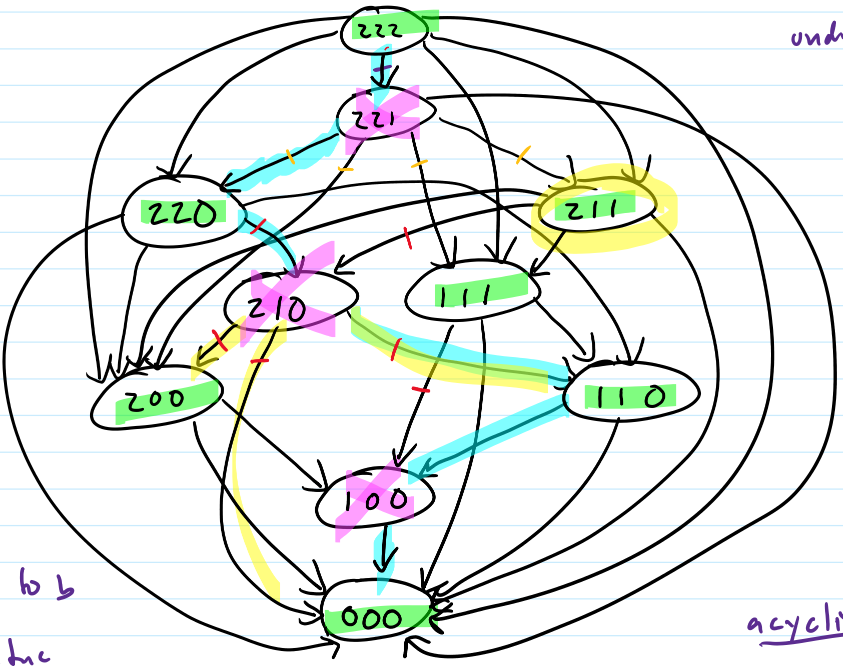
Chomp





undirected: irreflexive
symmetric

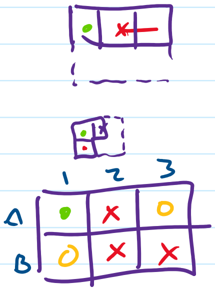
a

Edge a → b

pick square in a
to leave state b

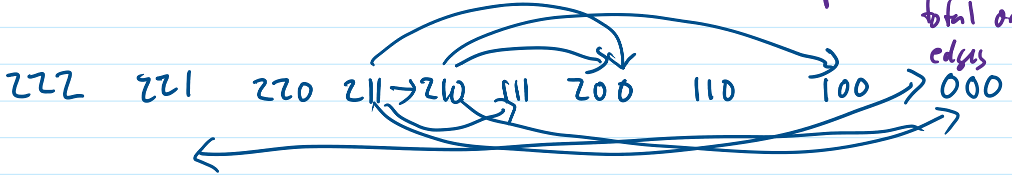
aRb iff
can move from a to b

chomp antisymmetric

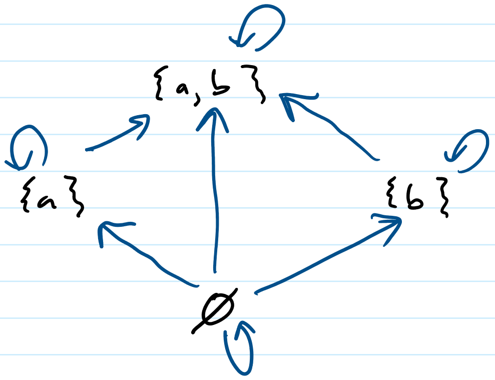


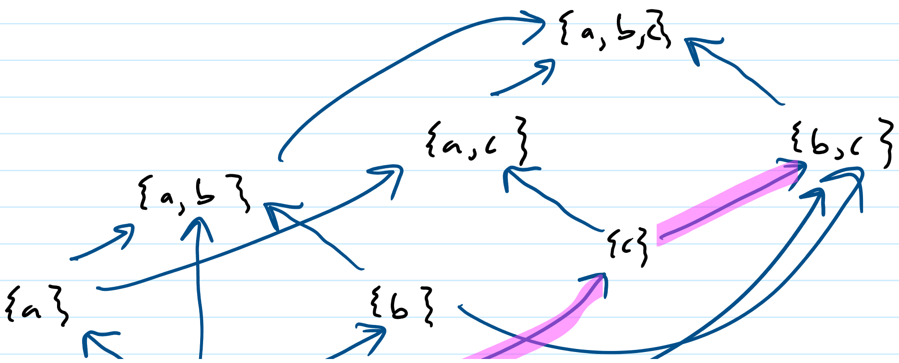
acyclic

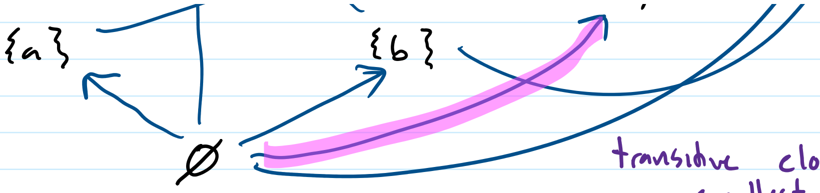
topological sort
total ordering of verts
edges all left → right



$A \subseteq B$







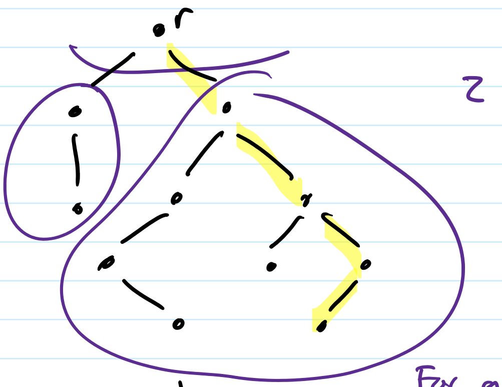
transitive closure

smallest containing transitive relation

Binary Tree

Binary tree: a rooted tree in which every node has ≤ 2 children

$\forall n \in \mathbb{N}, n \geq a \rightarrow$



For all #s of nodes n,
for all trees with n nodes ....

THM: All binary trees have $\leq 2^{h+1}-1$ nodes

Proof: We prove for all possible heights h, all binary trees of height h have $\leq 2^{h+1}-1$ nodes

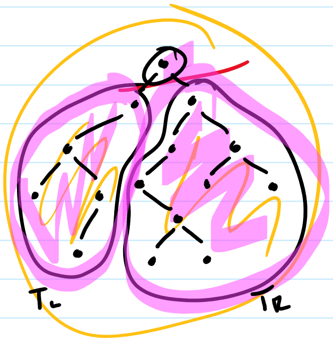
Base cases:

$(h=0)$ one node $2^{h+1}-1 = 1$; $1 \leq 1$

Ind step: suppose $h>0$ and for all i, $0 \leq i < h$, all trees of height i have $\leq 2^{i+1}-1$ nodes

Let T be a tree of height h

$h_L, h_r \leq h-1$



$$n_L \leq 2^{h_L+1}-1 \leq 2^h-1$$
$$n_R \leq 2^{h_R+1}-1 \leq 2^h-1$$

$$n = n_L + n_r + 1 \leq 2^h - 1 + 2^h - 1 + 1$$
$$= 2^{h+1} - 1$$

Sorting

i = 0
i = 2

	assign	arrcy	len	add	sub	compare	and
`def insertion_sort(l):`							
`n = len(l)`	1		1				
`i = 1`	1						
`while i < n:` ← get to this line n times						n	
`j = i`	$n-1$				$n-1$		
`while j - 1 >= 0 and l[j] < l[j - 1]:`		n^2+n-2			n^2+n-2	n^2+n-2	n^2+n-2
`temp = l[j - 1]`	$2n^2-2n$	$2n^2-2n$			$\frac{3}{2}n^2-\frac{3}{2}n$		
`l[j - 1] = l[j]`							
`l[j] = temp`							
`j = j - 1`							
`i = i + 1`	$n-1$			$n-1$			
`return l`							
TOTAL OPS	$2n^2$	$3n^2-n-2$	1	$n-1$	$\frac{5}{2}n^2+\frac{1}{2}n-3$	n^2+2n-2	n^2+n-2
time /op	t_1	t_2	t_3	t_4	t_5	t_6	t_7

there may be mistakes in here, but the point is that our tool for comparing running times shouldn't require us to do this much work

TOTAL TIME $(2t_1+3t_2+\frac{5}{2}t_5++t_6+t_7)n^2-(t_2-t_4-\frac{1}{2}t_5-2t_6-t_7)n-(2t_2-t_3+t_4+3t_5+2t_6+2t_7)$

	assign	array	len	add	sub	comp
`def selection_sort(l):`						
`n = len(l)`	1		1			
`i = 0`	1				n	n
`while i < n - 1:`						
`smallest_loc = i`	$n-1$					
`j = i + 1`	$n-1$			$n-1$		
`while j < n:`						$\frac{1}{2}n^2+\frac{1}{2}n-1$
`if l[j] < l[smallest_loc]:`		$2n^2-2n$				$\frac{1}{2}n^2-\frac{1}{2}n$
`smallest_loc = j`	$\frac{1}{2}n^2-\frac{1}{2}n$					
`j = j + 1`	$\frac{1}{2}n^2-\frac{1}{2}n$			$\frac{1}{2}n^2-\frac{1}{2}n$		
`temp = l[smallest_loc]`	$4n-4$	$4n-4$		$4n-4$		
`l[smallest_loc] = l[i]`						
`l[i] = temp`						
`i = i + 1`						
`return l`						

TOTAL TIME $(t_1+2t_2+\frac{1}{2}t_4+t_6)n^2+(5t_1-2t_2+\frac{9}{2}t_4+t_5+t_6)n-(4t_1+4t_2-t_3+5t_4+t_6)$

big-Oh Notation

DEF: For $f, g : \mathbb{Z}^+ \to \mathbb{R}^+$

size of input to alg

time used
# steps
storage used

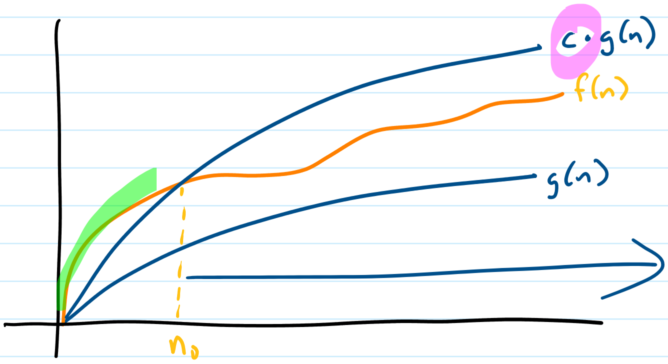
f is big-oh of g

$f \le g$

f is order at most g ($f(n) \in O(g(n))$) means

$f \underset{\sim}{\le} g$

$\exists n_0 \in \mathbb{Z}^+, c \in \mathbb{R}^+$ s.t. $\forall n \ge n_0,\ f(n) \underset{\sim}{\le} c \cdot g(n)$



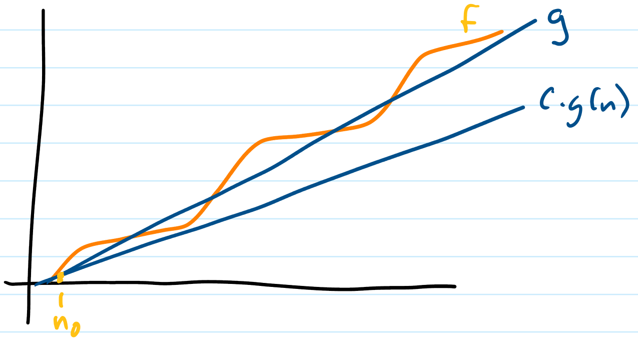
$\frac{1}{2} \cdot n^2 \in O\left(\frac{1}{20000000} \frac{1.62^n}{\sqrt{5}}\right)$

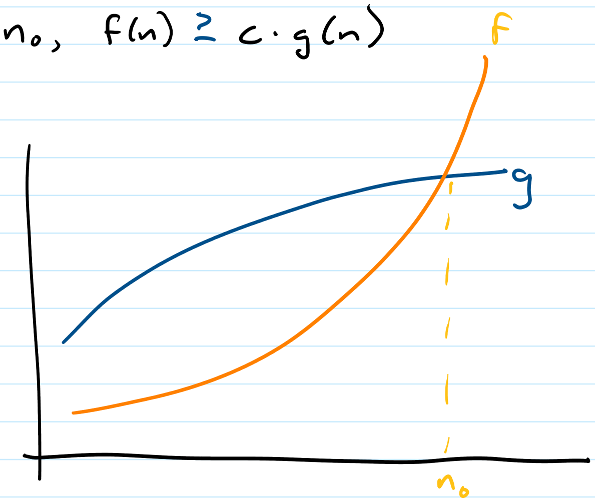
$n^2 \in O(1.62^n)$

f is order at least g ($f(n) \in \Omega(g(n))$) means

$\exists n_0 \in \mathbb{Z}^+, c \in \mathbb{R}^+$ s.t. $\forall n \ge n_0,\ f(n) \ge c \cdot g(n)$

$f \ge g$



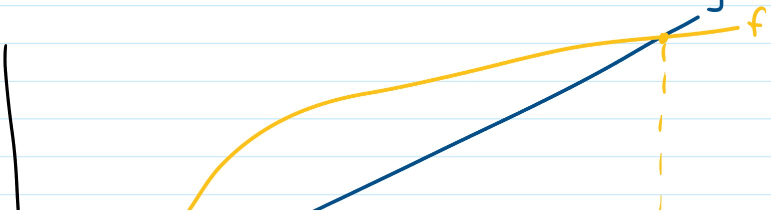


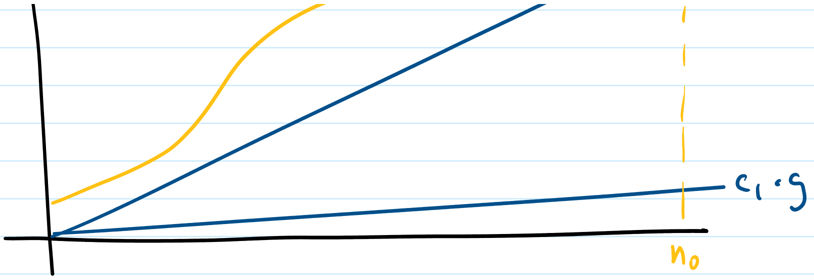
$f = g$

f is order g ($f(n) \in \Theta(g(n))$) means

$\exists n_0 \in \mathbb{Z}^+, c_1, c_2 \in \mathbb{R}^+$ s.t. $\forall n \ge n_0,\ c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$

$g = 1 \cdot g = c_2 \cdot g$




$c_1 \cdot g$
n_0