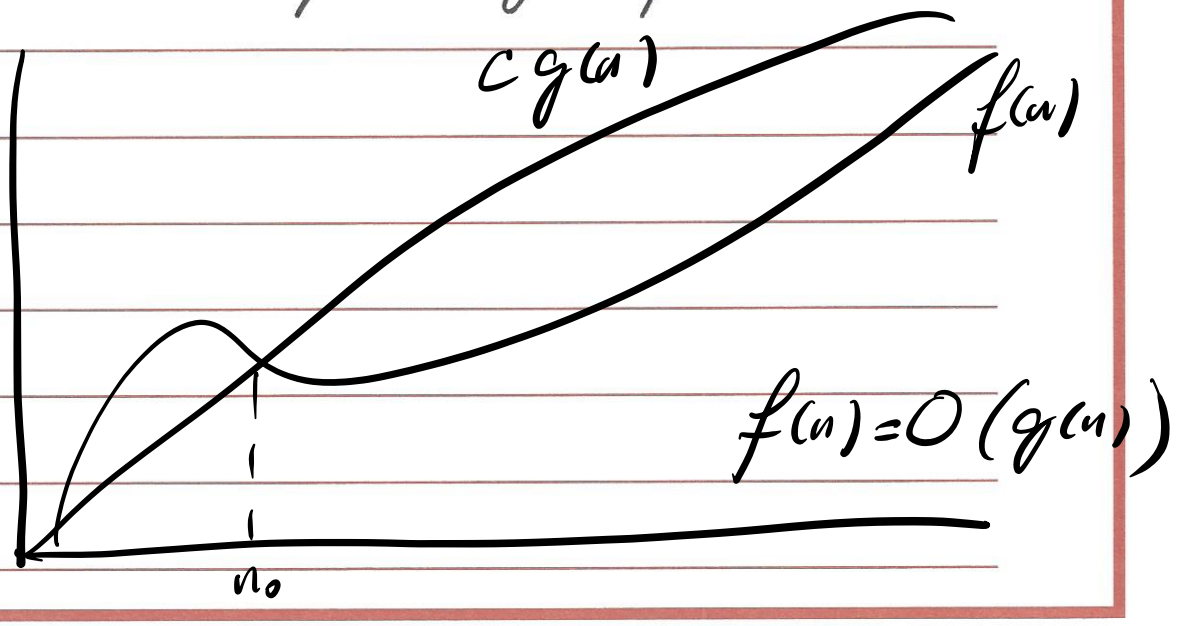
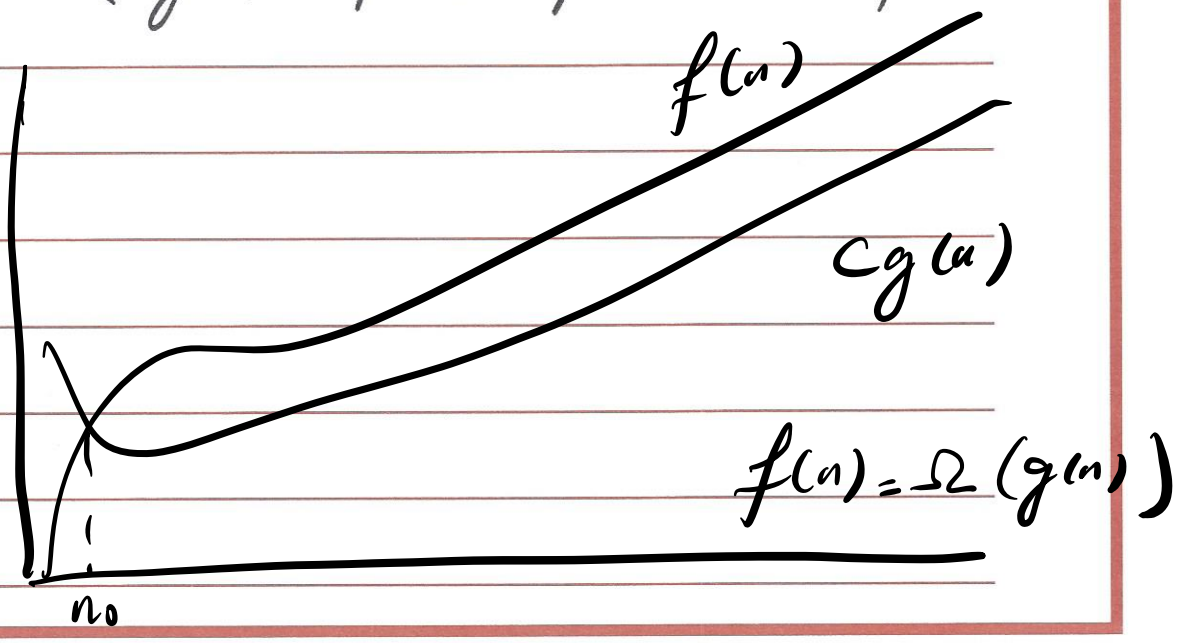
Review of the Asymptotic Notations

Formally, $O(g(n)) = \{ f(n) \mid$ there exist positive constants c and n_0 such that $0 \leq f(n) \leq c\,g(n)$ for all $n \geq n_0 \}$



- T Any quadratic function is $O(n^2)$
- T Any linear " " $O(n^2)$
- F Any cubic " " $O(n^2)$

$\Omega(g(n)) = \{ f(n) \mid$ there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for $n \geq n_0 \}$

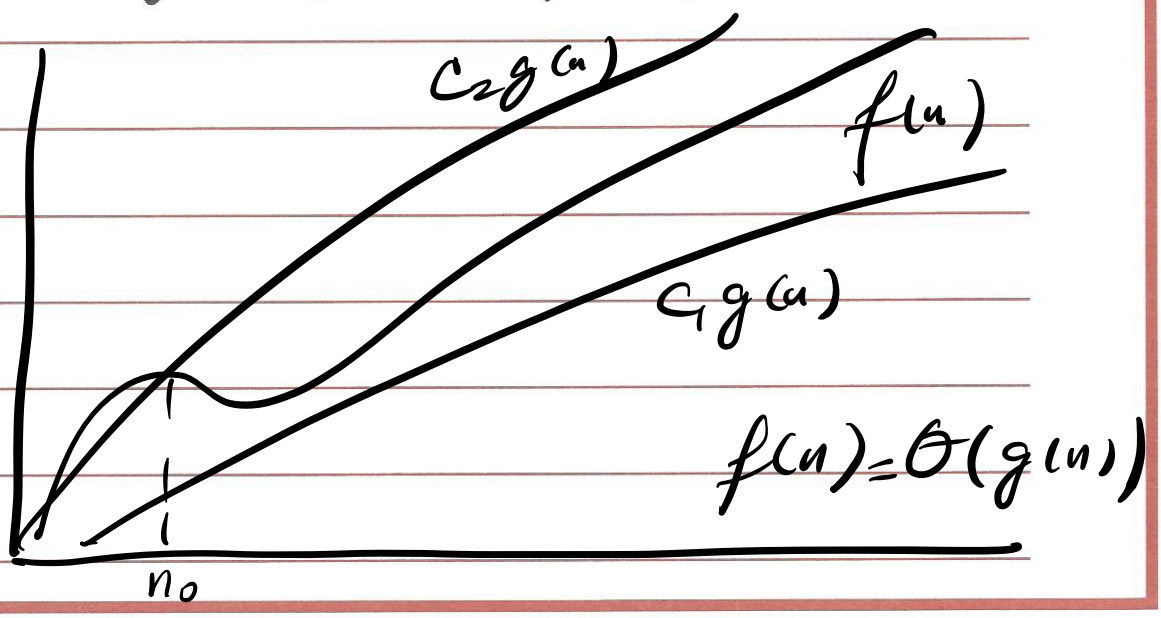


T Any quadratic function is $\Omega(n^2)$

F Any linear " " $\Omega(n^2)$

T Any cubic " " $\Omega(n^2)$

$\Theta(g(n)) = \{ f(n) \mid$ there exist positive constants c_1, c_2, and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0 \}$

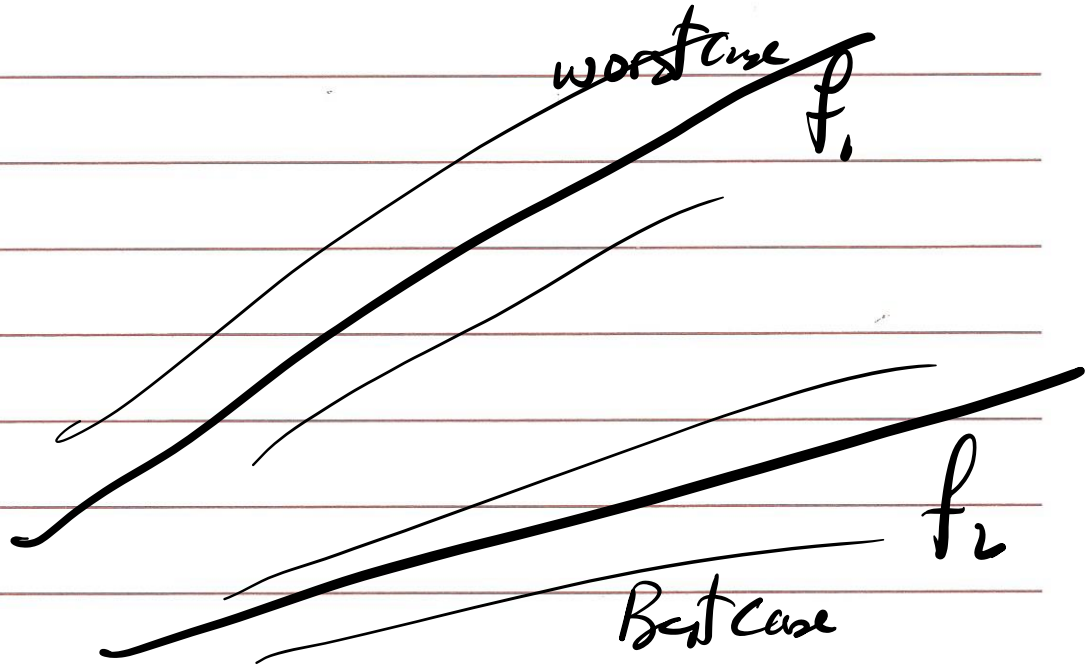


T Any quadratic function is $\Theta(n^2)$

F Any linear " $\Theta(n^2)$

F Any cubic " $\Theta(n^2)$

	Worst Case	Best Case
linear search	$O(n), \Omega(n), \Theta(n)$	$O(1), \Omega(1), \Theta(1)$
Binary "	$O(\lg n), \Theta(\lg n)$	$O(1), \Omega(1), \Theta(1)$
Insertion sort	$O(n^2), \Theta(n^2), \Omega(n^2)$	$O(n)\ \Theta(n), \Omega(n), \Omega(1)$
Merge Sort	$O(n \lg n), \Theta(n \lg n)$	$O(n \lg n), \Theta(n \lg n)$



Worst Case Performance:

Algorithm A: $\Theta(4^n n^3 \lg n)$

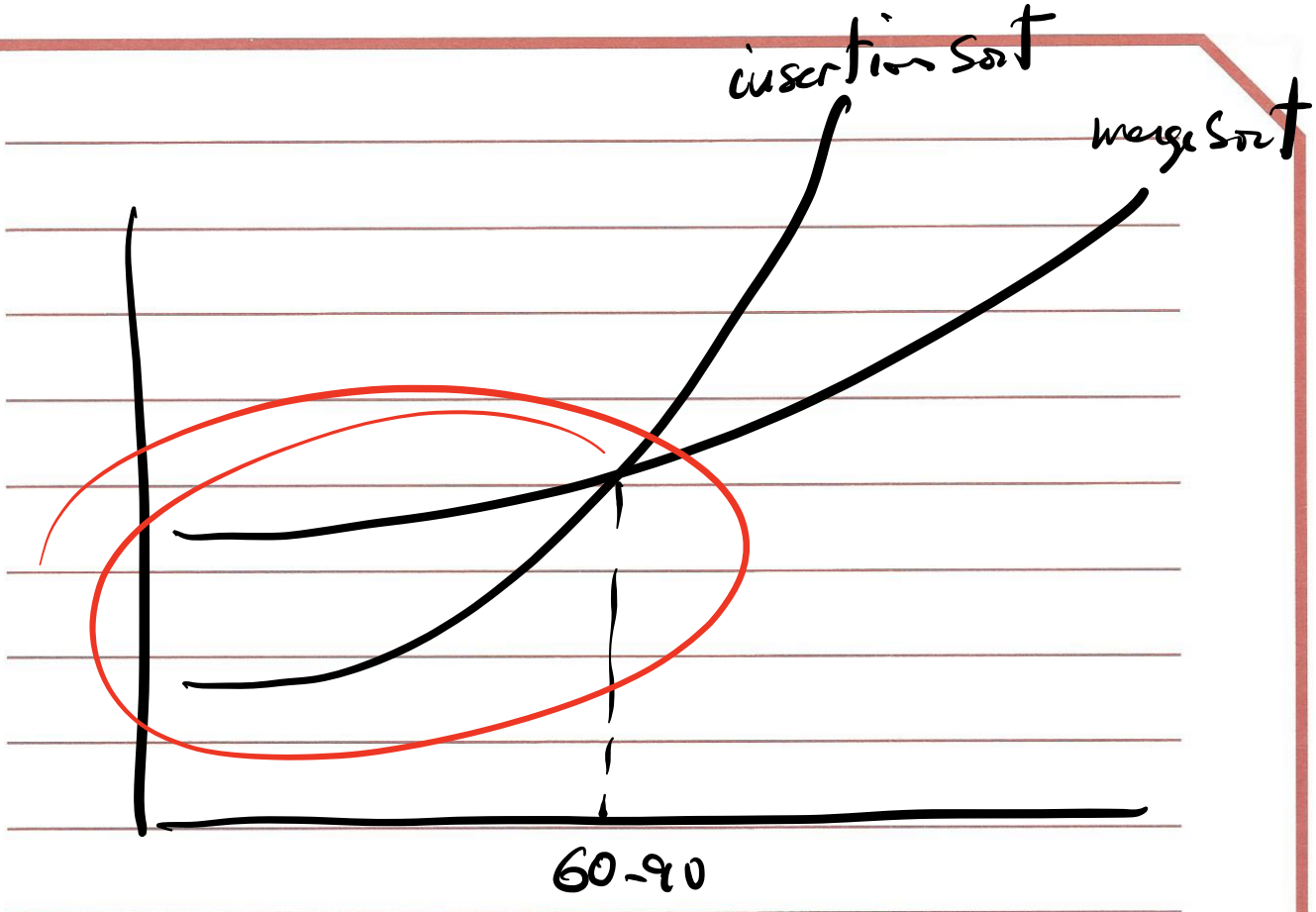
Algorithm B: $\Theta(3^n n^8 (\lg n)^2)$

Exponential — fastest

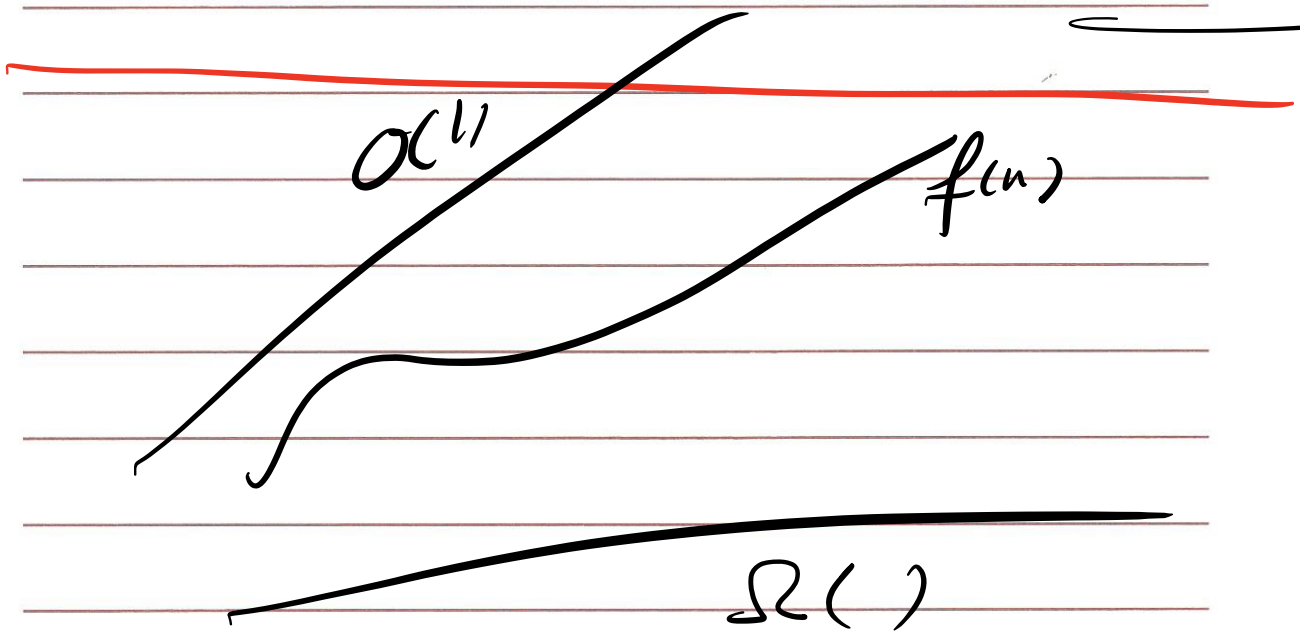
polynomial

Logarithmic — slowest

$[A][B] = [C]$



hybrid sort. will be $O(n \log n)$



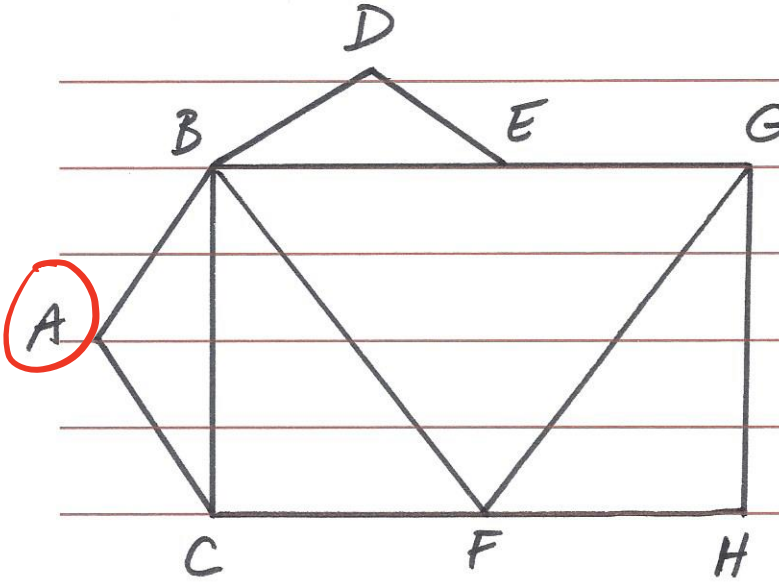
Review of BFS & DFS

Q: What are we searching for?

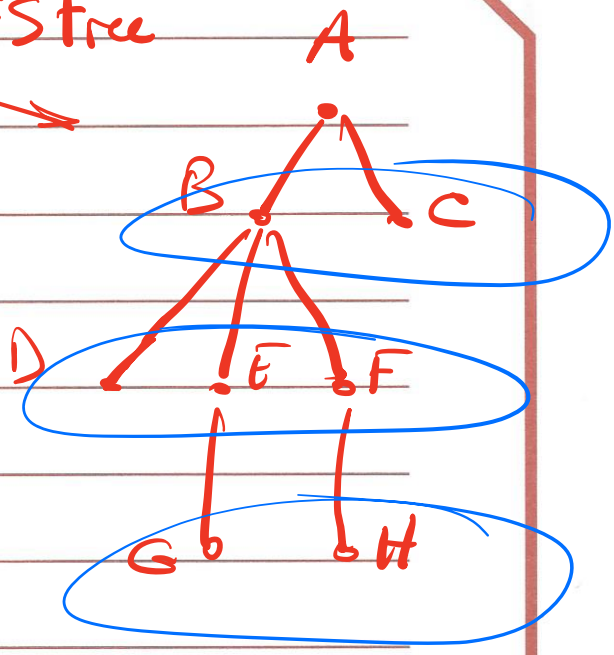
- Find out if there is a path from A to B.

- Find all nodes that can be reached from A.

BFS



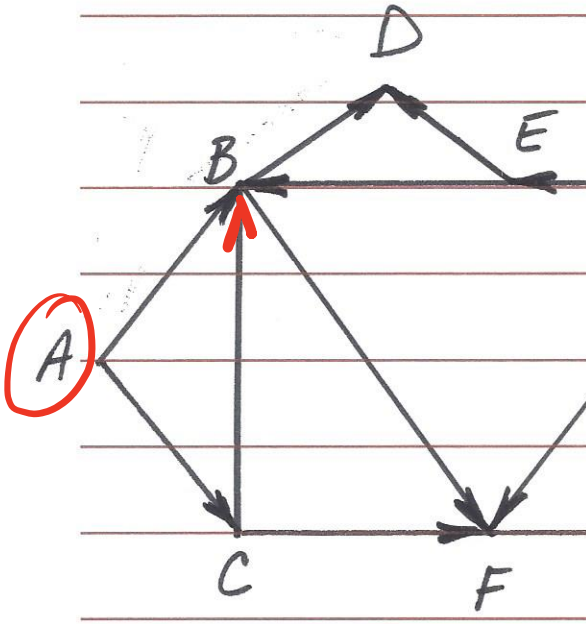
BFS tree



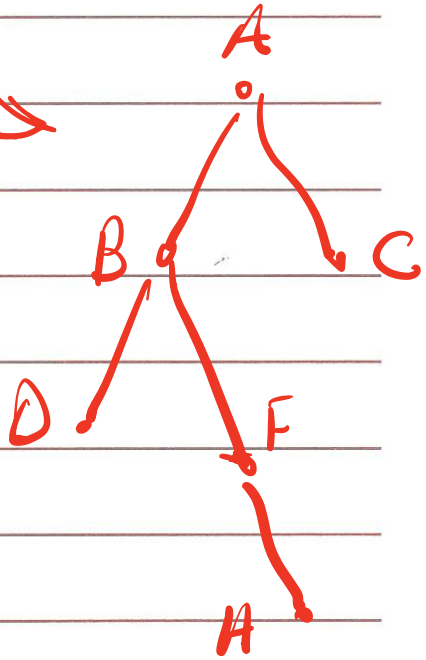
runs in $O(m+n)$

m = no. of edges
n = no. of nodes

DFS

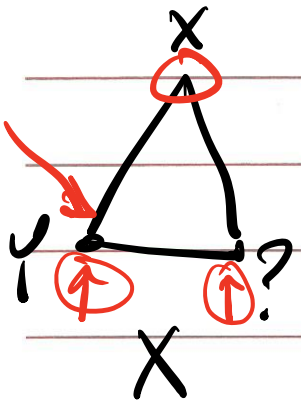


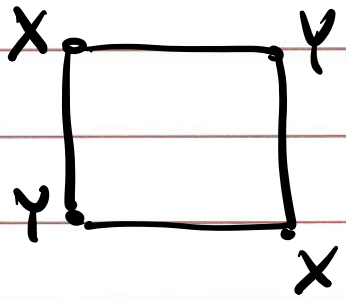
DFS tree

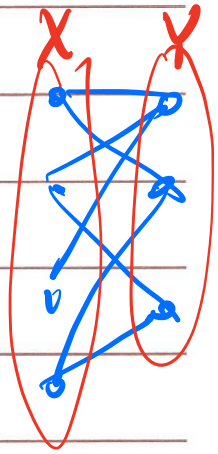


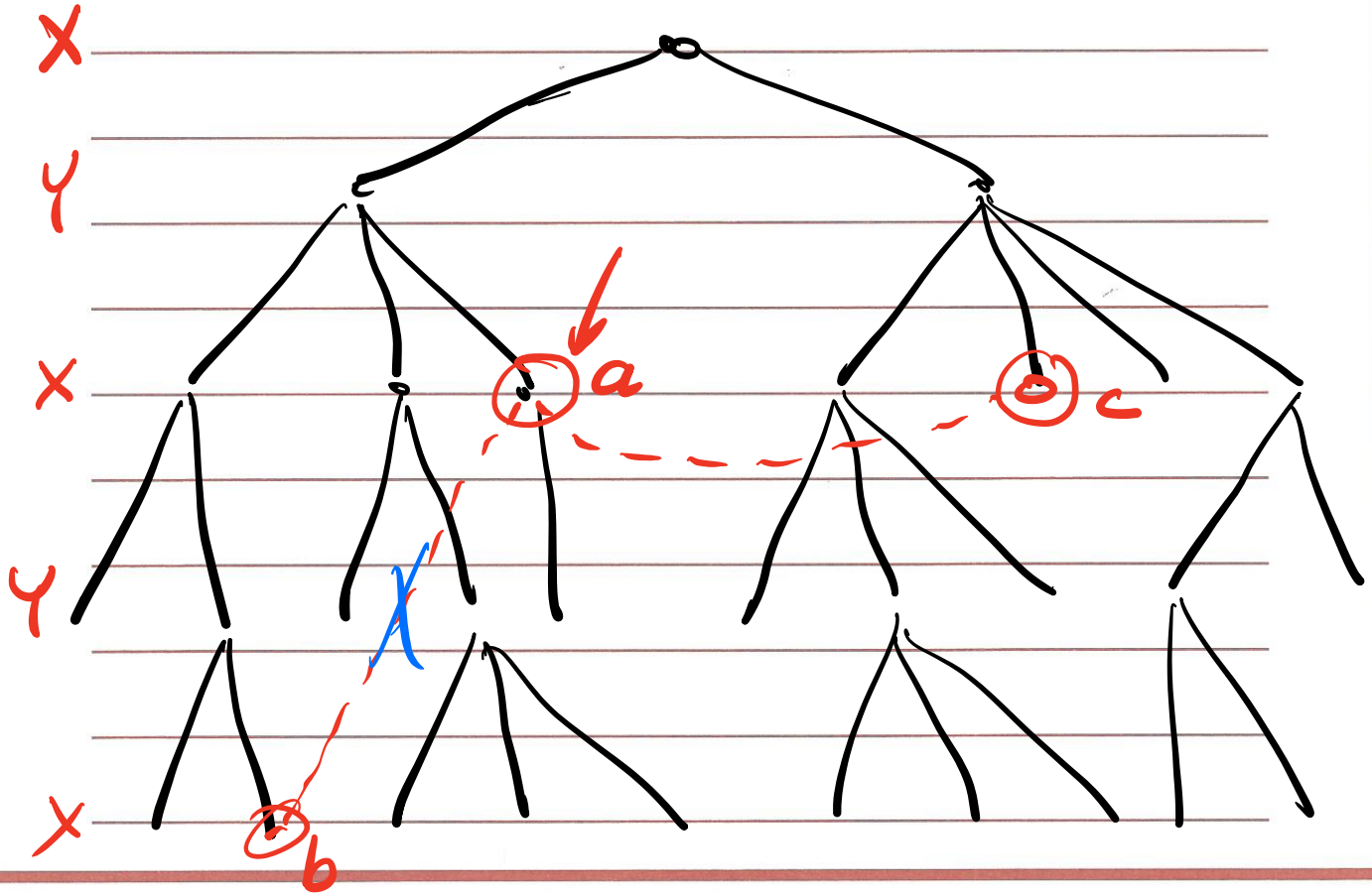
runs in $O(m+n)$

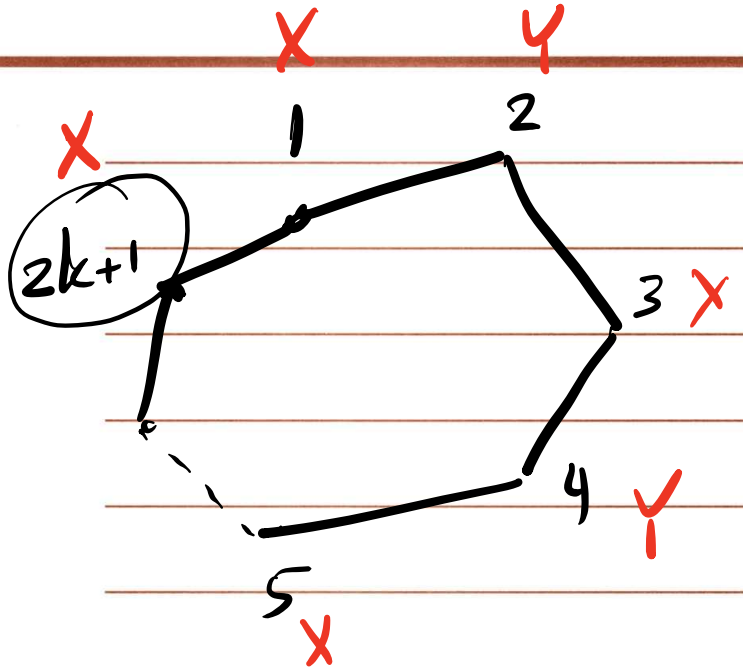
Q: How do you determine if a graph is bipartite?



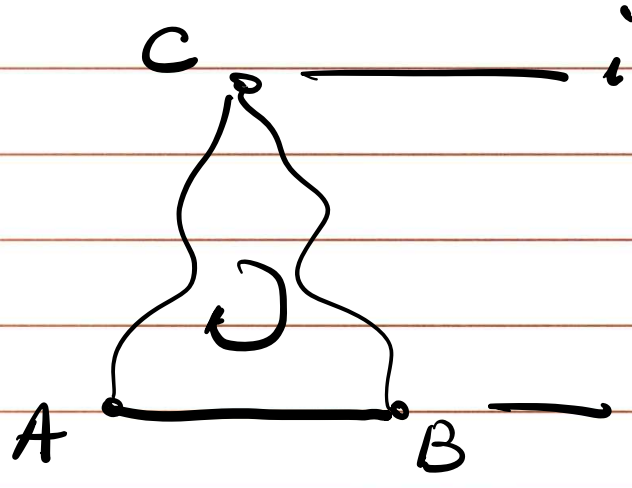








FACT: If a graph G is bipartite then it cannot contain an odd cycle.



length of cycle = $\underbrace{2*(j-i)+1}$

odd!

Solution:

$O(m+n)$ Run BFS starting from any node, say $\underline{S}$. Label each node $\underline{\text{Red}}$ or $\underline{\text{Blue}}$ depending on whether they appear at an odd or even level on the BFS tree.

$O(m)$ Then, go through all edges and examine the labels at the two ends of the edge. If all edges have a Red end and a Blue end, then the graph is bipartite.

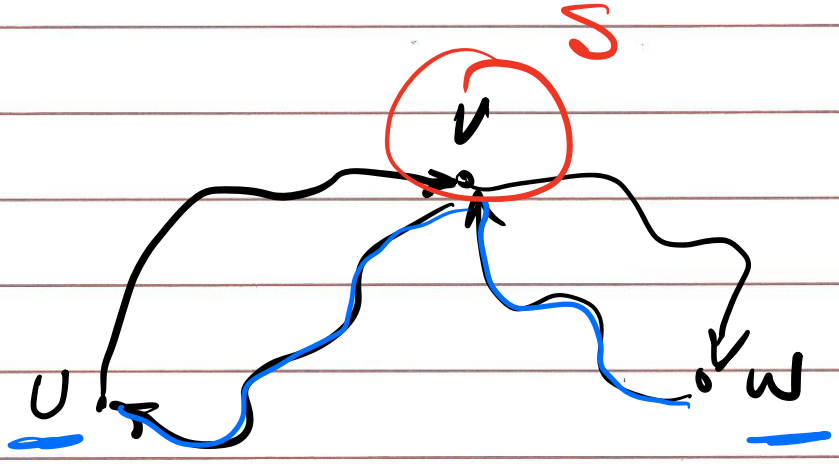
Otherwise, the graph is not bipartite.

overall complexity $= O(m+n)$

Def. A directed graph is strongly connected if there is a path from any point to any other point in the graph.

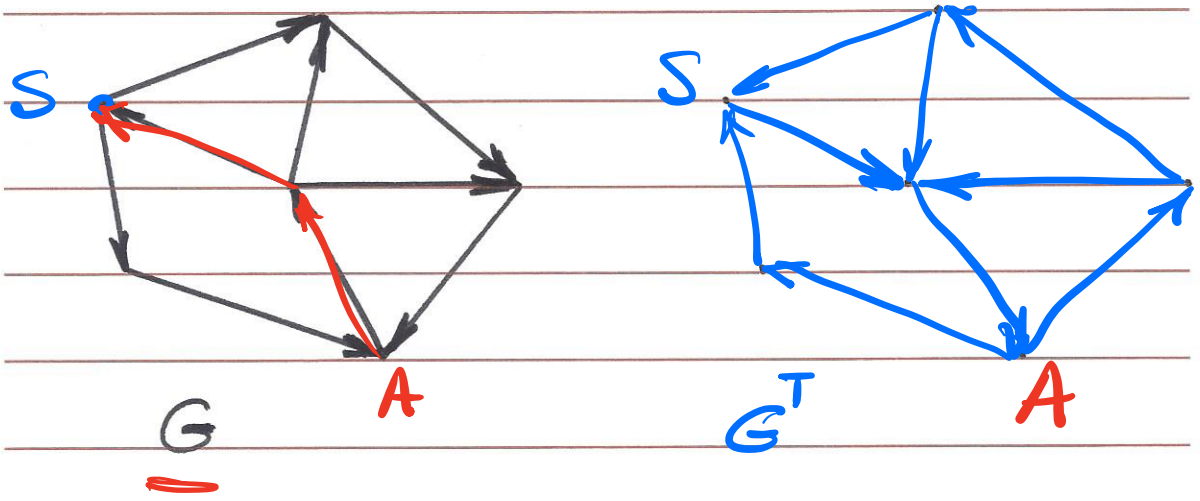
Q: How do you know if a given directed graph is strongly connected?

Brute Force: Run BFS/DFS starting from every node ... Takes $O(n^2+nm)$



u and w are also mutually reachable

Transpose of a directed graph



Mutually Reachable Nodes

Solution:

1. Use BFS or DFS to find all nodes reachable from s (an arbitrary node) in G. If some nodes are not reachable from s, stop. The graph is not strongly connected.

O(m+n)

otherwise, continue with step 2.

O(m+n) 2. Create G^T (Transpose of G)

O(m+n) 3. Use BFS or DFS to find all nodes reachable from s in G^T. If some nodes are not reachable from s, then the graph is not strongly connected.

Otherwise, the graph is strongly connected.

overall complexity = O(m+n)