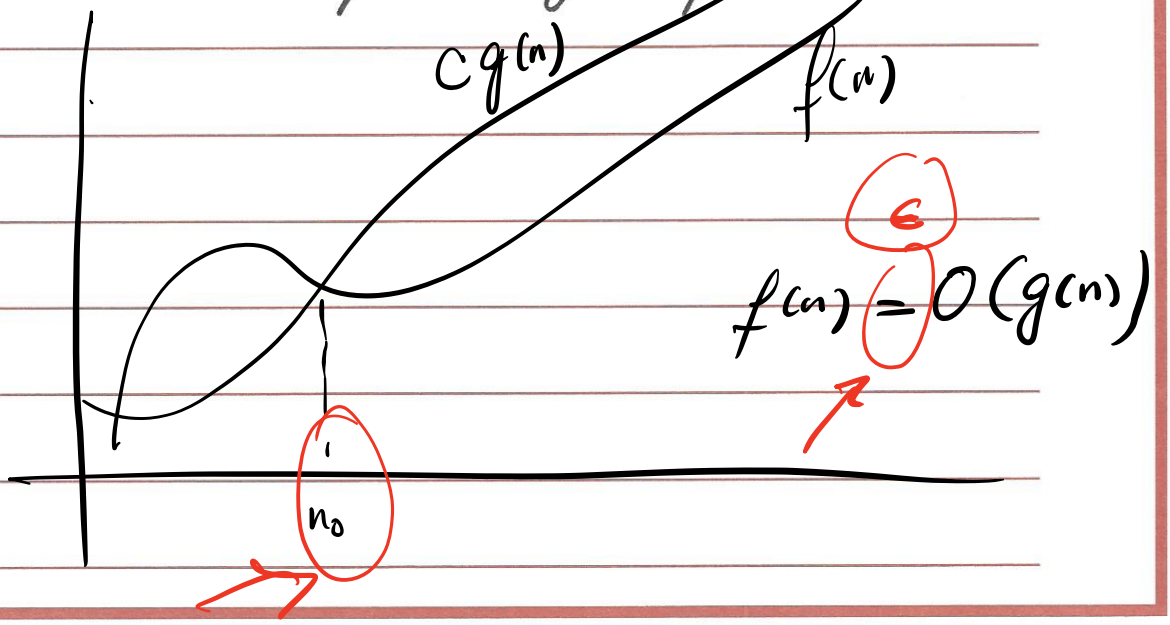
Review of the Asymptotic Notations

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 $n \geq n_0 \}$

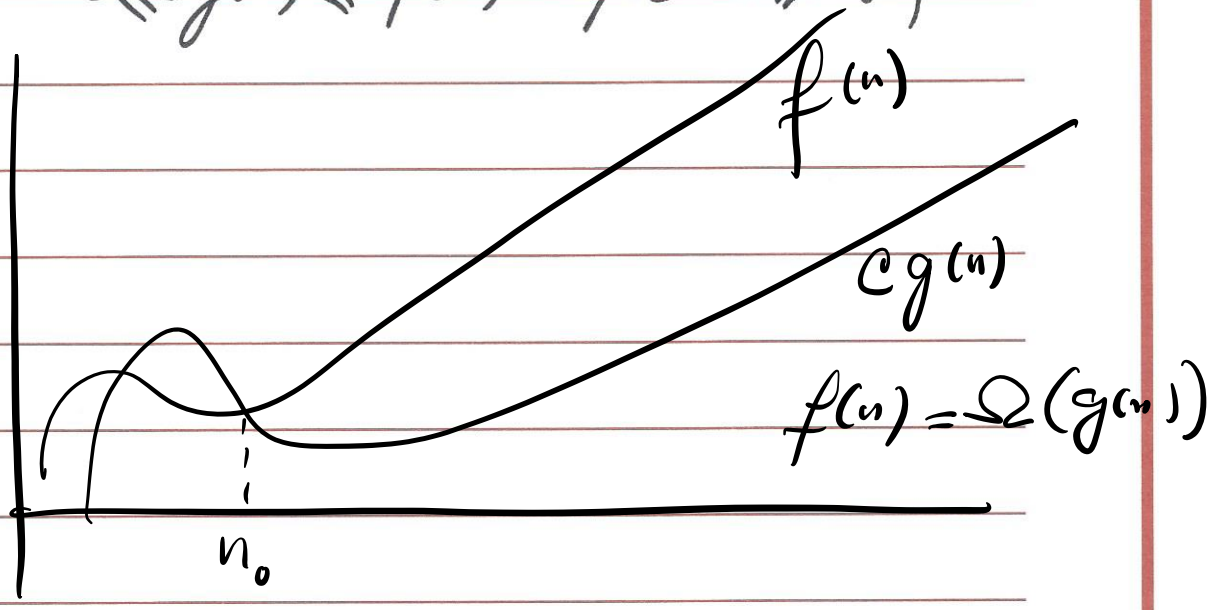


T Any quadratic function is $O(n^2)$

T Any linear " "

F Any cubic " "

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

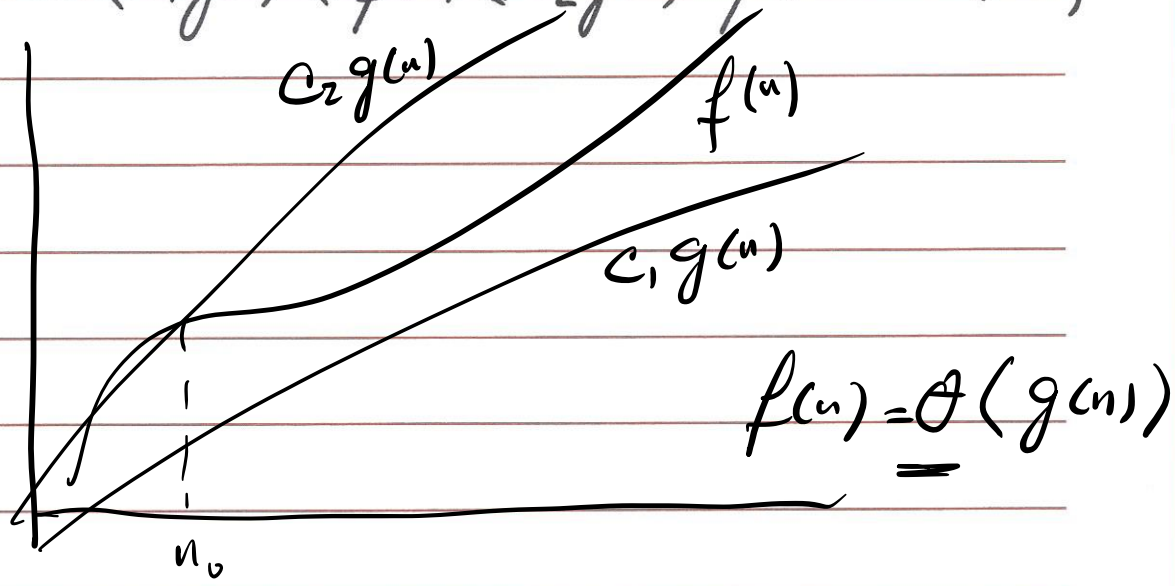


T Any quadratic func. is $\Omega(n^2)$

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 $n \geq n_0 \}$



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

T — Any quadratic func. is $\Theta(n^2)$

F — Any linear " " "

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 "	$O(\lg n), \Omega(\lg n), \Theta(\lg n)$	$O(1), \Omega(1), \Theta(1)$
Insertion Sort	$O(n^2), \Omega(n^2), \Theta(n^2)$	$O(n), \Omega(n), \Theta(n)$
Merge Sort	$O(n \lg n), \Omega(n \lg n), \Theta(n \lg n)$	$O(n \lg n), \Omega(n \lg n), \Theta(n \lg n)$

Worst Case Performances:

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

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

Exponential Component — fastest growing

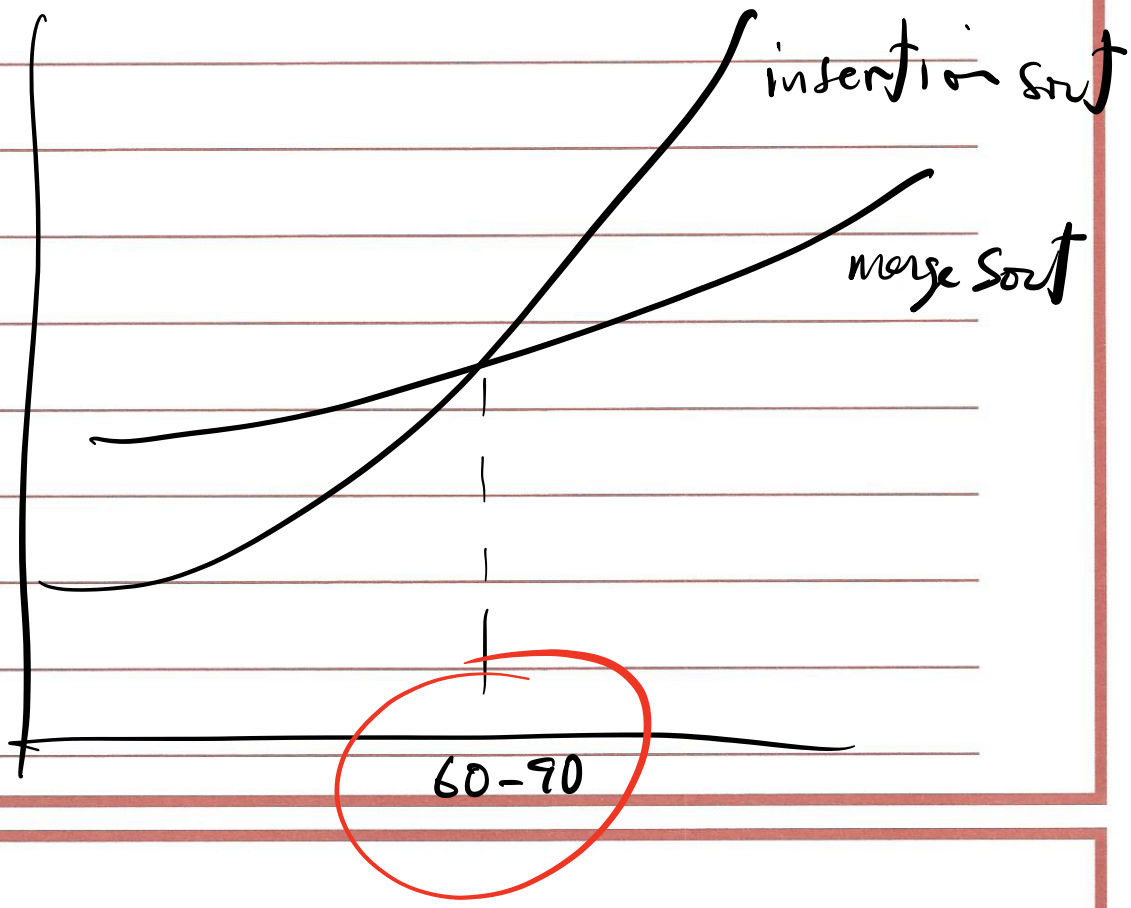
Polynomial "

Logarithmic " — slowest growing

$$[A]^T[B] = [C]$$

$\Theta(n)$ — $10n$ is

$\Theta(n^2)$ — $2n^2$



Can create a hybrid of merge sort + insertion sort.

$O(nk + n \log \frac{n}{k})$

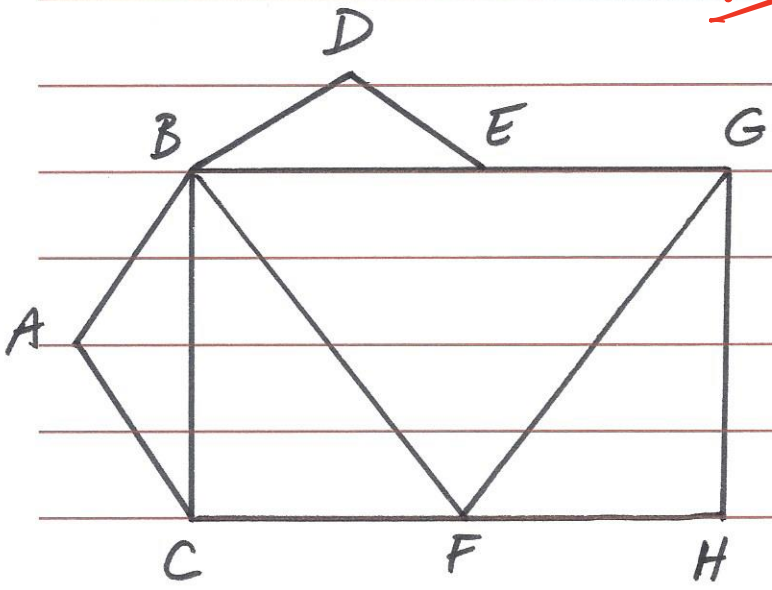
Review of BFS & DFS

Q: What are we searching for?

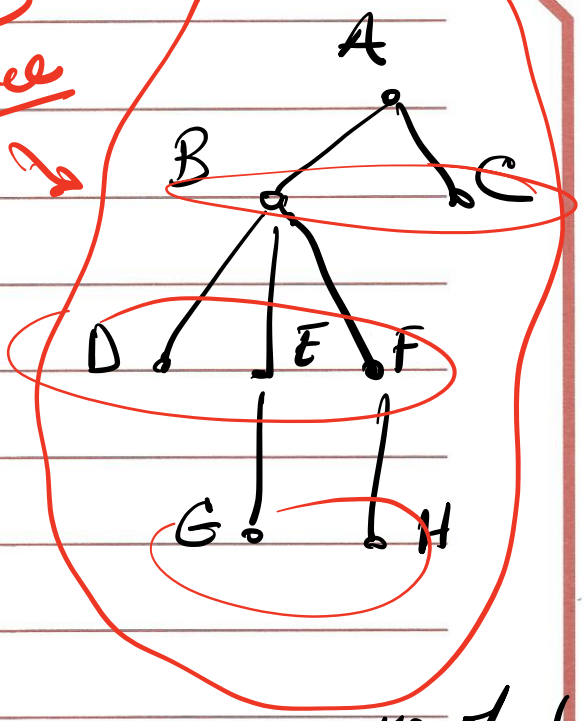
- Find out if there is a path from A to B.

- Find all nodes that can be reached from A.

BFS



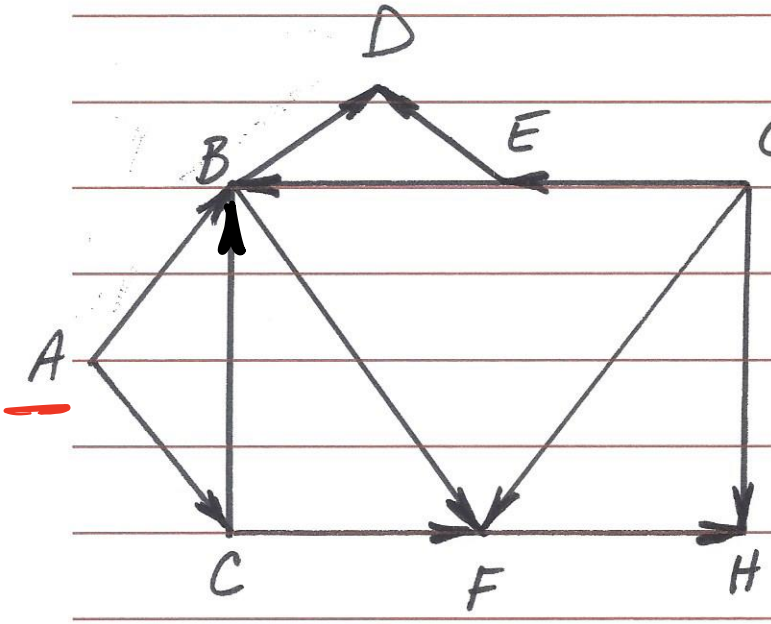
BFS Tree



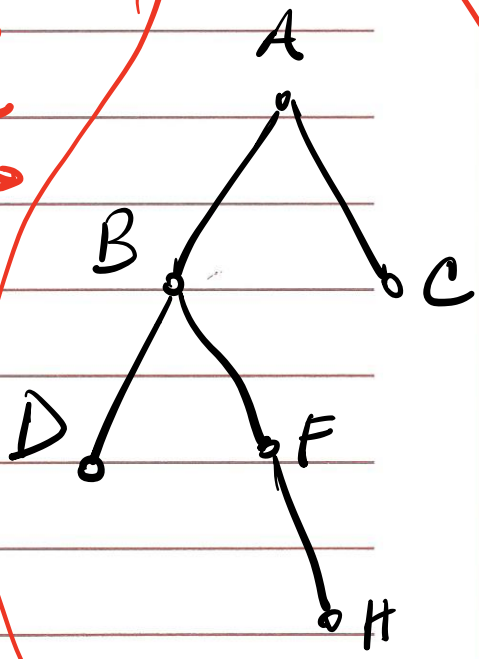
Worst Case Performance $O(m+n)$

m: no. of edges
n: no. of nodes

DFS

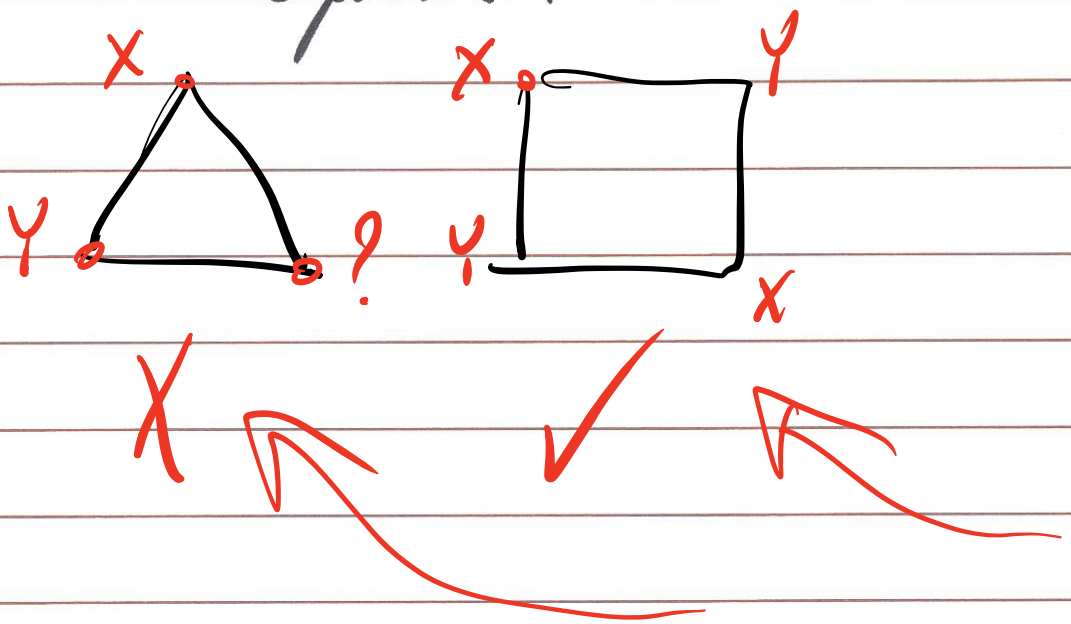


DFS Tree

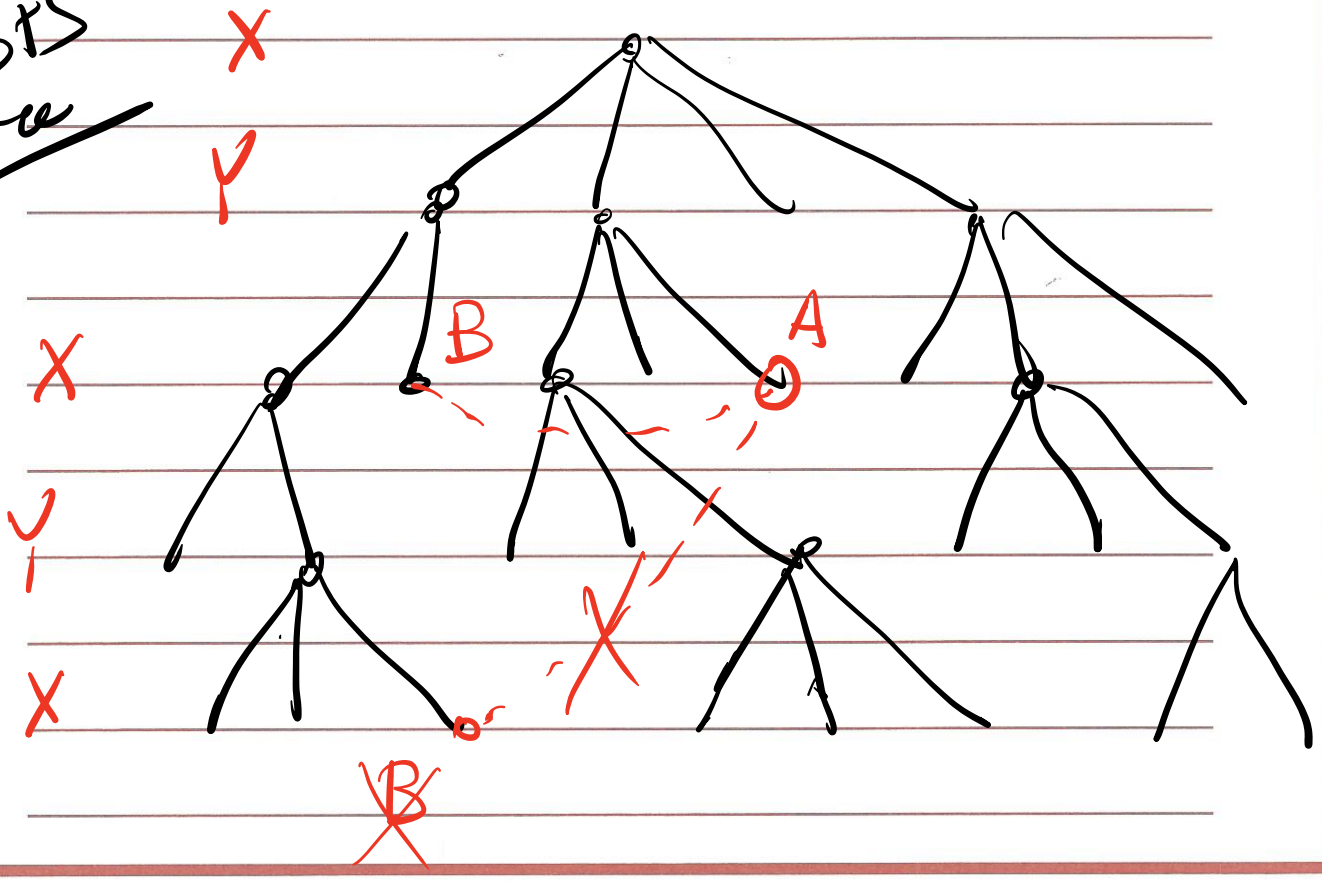


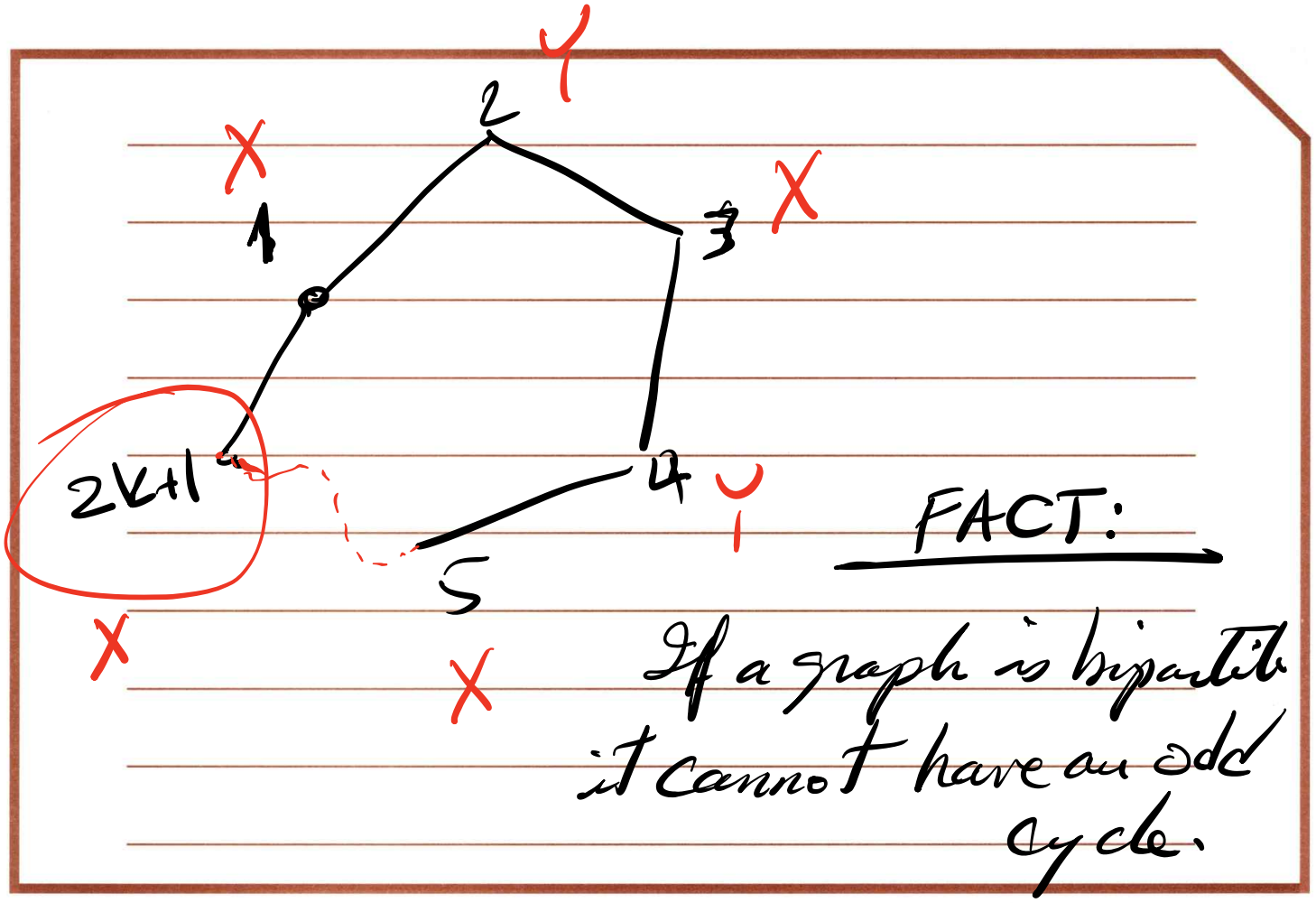
Worst Case Performance $O(m+n)$

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



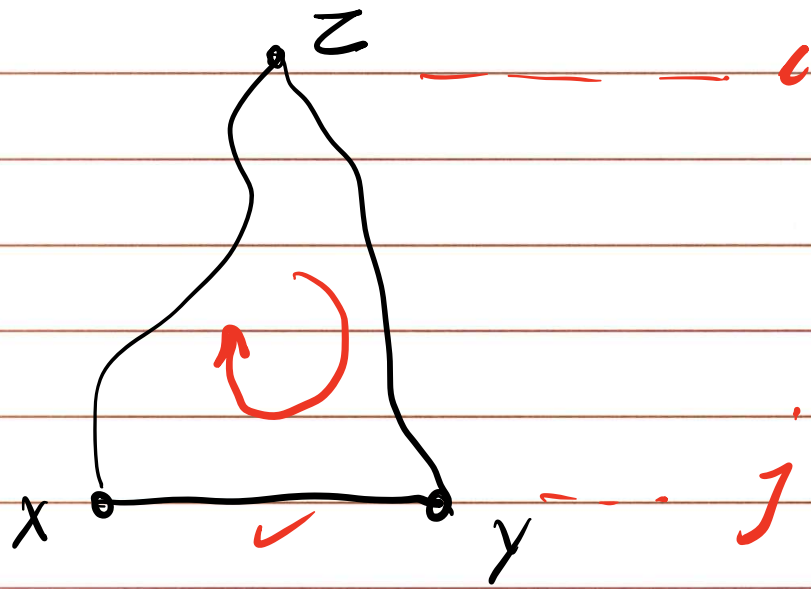
BFS Tree





FACT:

If a graph is bipartite it cannot have an odd cycle.



length of cycle = $2*(j-i)+1$

odd!

Solution:

$O(m+n)$ Run BFS starting from any node, say S. Label each node Red or 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.

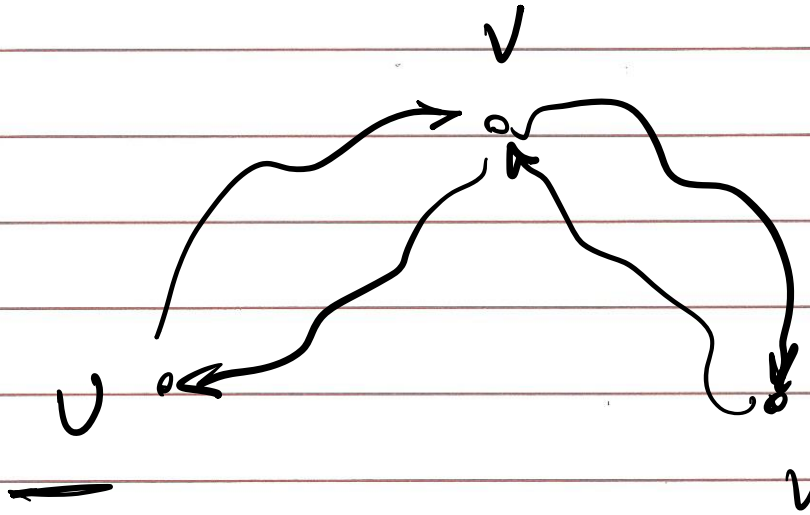
Otherwise, the graph is not bipartite.

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.

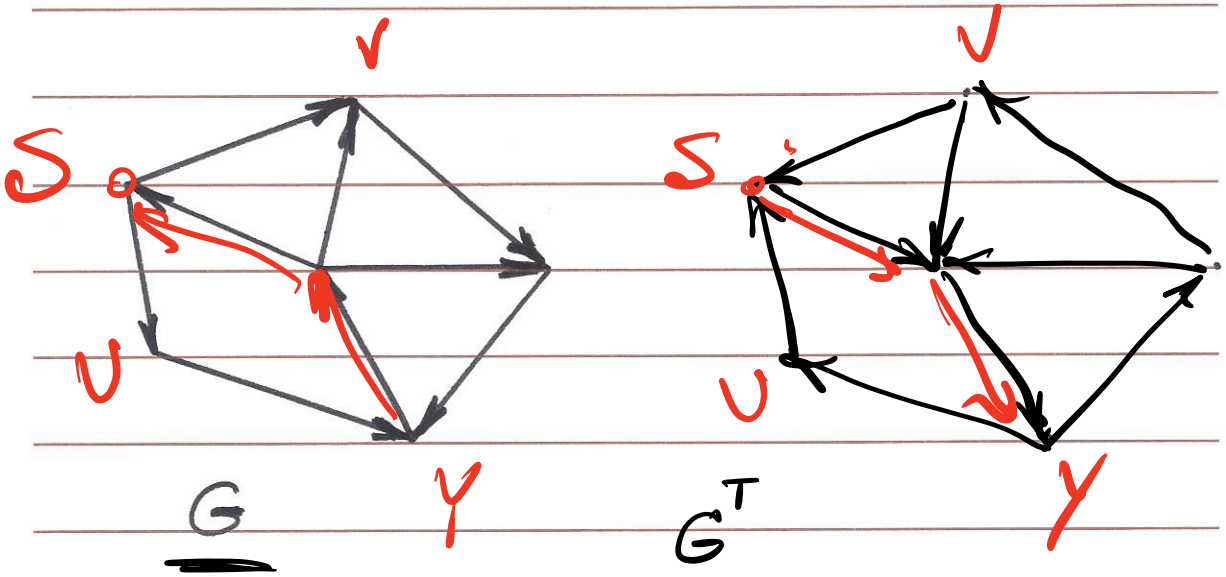
Q: How do you know if a given directed graph is strongly connected?

Brute force: Run BFS/DFS from each node. Takes $O(n^2+nm)$

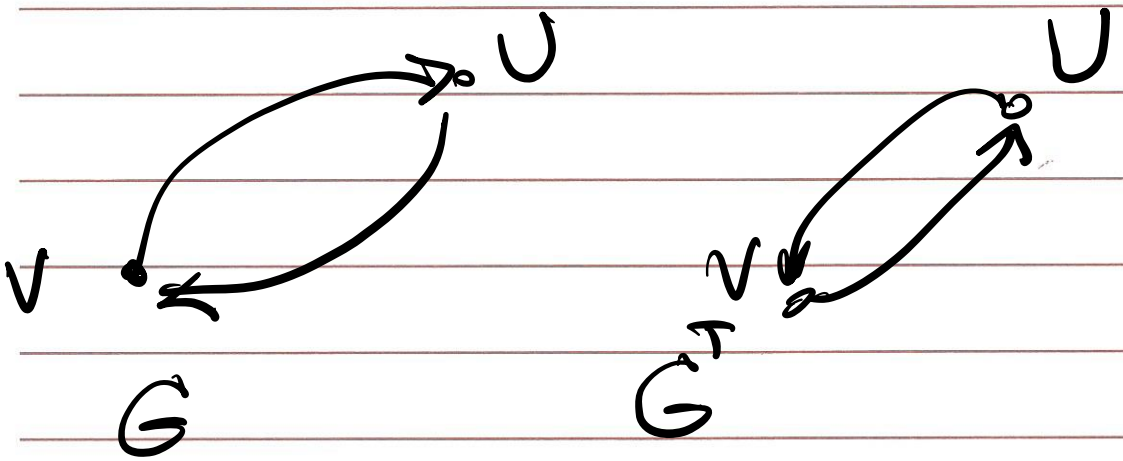


u & w are mutually reachable

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 = $O(m+n)$

Discussion 2

1. Arrange the following functions in increasing order of growth rate with g(n) following f(n) in your list if and only if *f(n) = O(g(n))*

$$\log n^n,\ n^2,\ n^{\log n},\ n \log \log n,\ 2^{\log n},\ \log^2 n,\ n^{\sqrt{2}}$$

2. Suppose that *f(n)* and *g(n)* are two positive non-decreasing functions such that *f(n) = O(g(n)).* Is it true that $2^{f(n)} = O(2^{g(n)})$?

3. Find an upper bound (Big O) on the worst case run time of the following code segment.

```
void bigOh1(int[] L, int n)
   while (n > 0)
      find_max(L, n); //finds the max in L[0…n-1]
      n = n/4;
```

Carefully examine to see if this is a tight upper bound (Big θ)

4. Find a lower bound (Big Ω) on the best case run time of the following code segment.

```
string bigOh2(int n)
   if(n == 0) return "a";
   string str = bigOh2(n-1);
   return str + str;
```

Carefully examine to see if this is a tight lower bound (Big θ)

5. What Mathematicians often keep track of a statistic called their Erdős Number, after the great 20th century mathematician. Paul Erdős himself has a number of zero. Anyone who wrote a mathematical paper with him has a number of one, anyone who wrote a paper with someone who wrote a paper with him has a number of two, and so forth and so on. Supposing that we have a database of all mathematical papers ever written along with their authors:

a. Explain how to represent this data as a graph.
b. Explain how we would compute the Erdős number for a particular researcher.
c. Explain how we would determine all researchers with Erdős number at most two.

6. In class, we discussed finding the shortest path between two vertices in a graph. Suppose instead we are interested in finding the *longest* simple path in a directed acyclic graph. In particular, I am interested in finding a path (if there is one) that visits all vertices.
Given a DAG, give a linear-time algorithm to determine if there is a simple path that visits all vertices.

1. Arrange the following functions in increasing order of growth rate with g(n) following f(n) in your list if and only if *f(n) = O(g(n))*

$n \lg n$ ← $\log n^n$, n^2, $n^{\log n}$, $n \log(\log n)$, $2^{\log_2 n}$, $\log^2 n$, $n^{\sqrt{2}}$ ← pol.

$2^{\log_2 n} = n^1$

$\log^2 n,\ n,\ n \log\log n,\ n \lg n,\ n^{\sqrt{2}},\ n^2,\ n^{\log n}$

2. Suppose that *f(n)* and *g(n)* are two positive non-decreasing functions such that *f(n)* = *O(g(n))*. Is it true that $2^{f(n)} = O(2^{g(n)})$?

$f(n) = 2n \qquad g(n) = n$

$f(n) = O(g(n))$

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

3. Find an upper bound (Big O) on the worst case run time of the following code segment.

```
void bigOh1(int[] L, int n)
  while (n > 0)
    find_max(L, n); //finds the max in L[0…n-1]
    n = n/4;
```

O(n)

$\log_4 n$

Carefully examine to see if this is a tight upper bound (Big θ)

Worst Case perf = $O(n \log n)$

$cn + \frac{1}{4}cn + \frac{1}{16}cn + \cdots$

Sum $< 2n$

$n + n/2 + n/4 + n/8 + n/16 + \cdots$

$2n$

worst case:

tight bound = $\theta(n)$

4. Find a lower bound (Big Ω) on the best case run time of the following code segment.

```
string bigOh2(int n)
   if(n == 0) return "a";
   string str = bigOh2(n-1);
   return str + str;
```

Carefully examine to see if this is a tight lower bound (Big θ)

Best case takes $\Omega(n)$

not a tight bound

n	output
0	a
1	aa
2	aaaa
⋮	⋮
n	aa...aa (2^n)

tight bound is $\Theta(2^n)$

$O(2^n)$

function representing Best case performance

$f(n)$

$\Omega(2^n)$

$\Omega(n)$

$\Omega(1)$

$$2^n + \frac{1}{2}2^n + \frac{1}{4}2^n + \cdots + 1$$

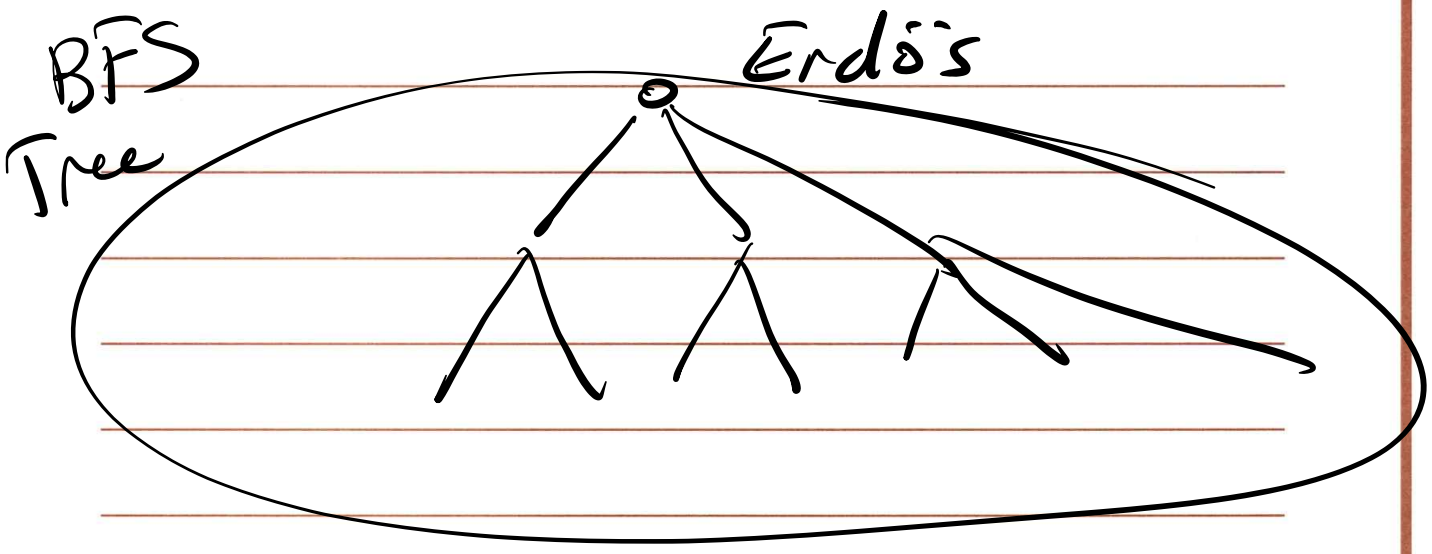
$$\sum = 2 \times 2^n = \Theta(2^n)$$

5. What Mathematicians often keep track of a statistic called their Erdős Number, after the great 20th century mathematician. Paul Erdős himself has a number of zero. Anyone who wrote a mathematical paper with him has a number of one, anyone who wrote a paper with someone who wrote a paper with him has a number of two, and so forth and so on. Supposing that we have a database of all mathematical papers ever written along with their authors:

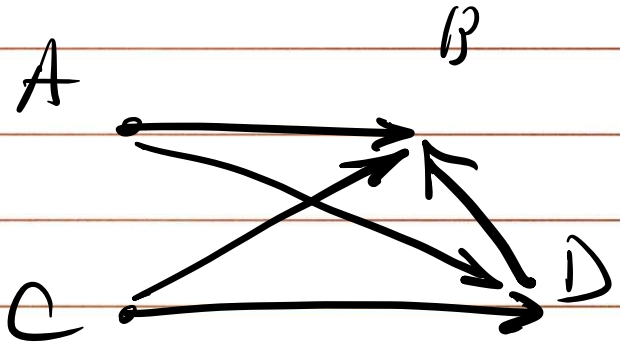
a. Explain how to represent this data as a graph.
b. Explain how we would compute the Erdős number for a particular researcher.
c. Explain how we would determine all researchers with Erdős number at most two.

a - nodes represent mathematicians
edge = co-authorship

b - Run BFS to find the level at which the mathematician appears



6. In class, we discussed finding the shortest path between two vertices in a graph. Suppose instead we are interested in finding the *longest* simple path in a directed acyclic graph. In particular, I am interested in finding a path (if there is one) that visits all vertices.
Given a DAG, give a linear-time algorithm to determine if there is a simple path that visits all vertices.



reverse topological ordering