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 $n \geq n_0 \}$

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

$\Theta(g(n)) = \{ f(n) \mid$ 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 $n \geq n_0 \}$

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

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

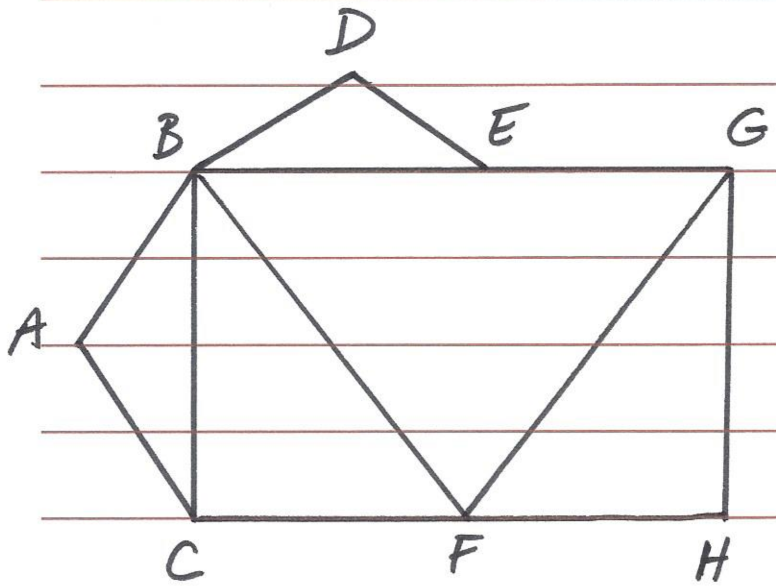
Review of BFS & DFS

Q: What are we searching for?

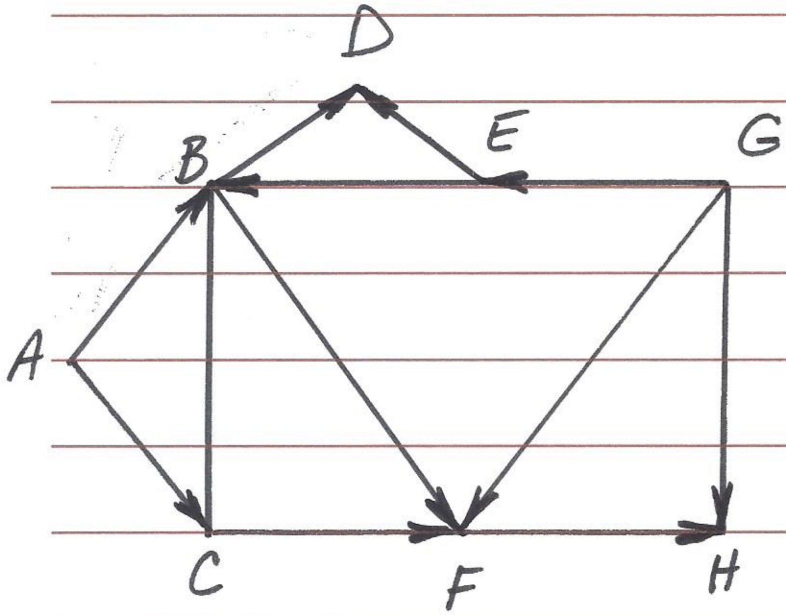
- Find out if there is a path from A to B.

- Find all nodes that can be reached from A.

BFS



DFS



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

Solution:

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.

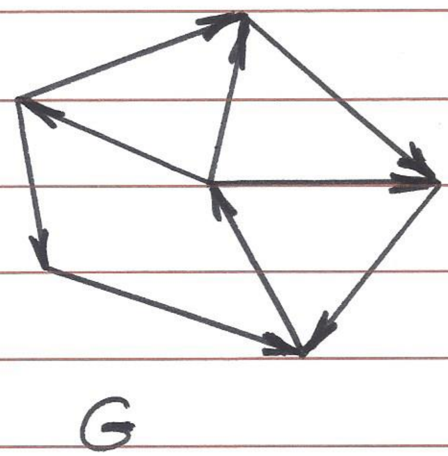
Then, go through all edges and examine the labels at the two ends of the edge. If all edges have a Red end and a Blue end, then the graph is bipartite.

Otherwise, the graph is not bipartite.

<u>Def.</u> 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?

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.

Otherwise, continue with step 2.

2. Create G^T (Transpose of G)
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.

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.