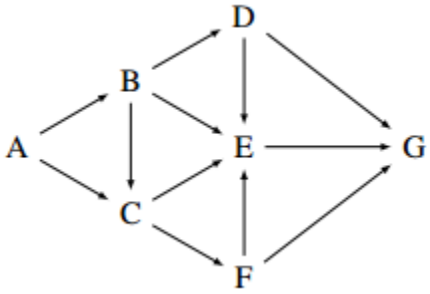
# CS 61B Discussion 11 Exam Prep Spring 2017

## 1 Warmup with DFS and BFS




1. For the graph above, give the vertices in the order they'd be visited by depth first search starting from vertex A, assuming that we always visit alphabetically earlier vertices first if there are multiple valid choices. The alphabet is ABCDEFG.
   ABCEGFD

2. For the graph above, give the vertices in the order they'd be visited by breadth first search starting from vertex A, assuming that we always visit alphabetically earlier vertices first if there are multiple valid choices.
   ABCDEFG

## 2 Summer '16 Final: Regex

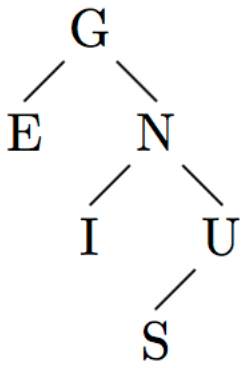
For the below regex problems, assume these are **non-java** strings, so double escaping is not necessary.

1. Consider the regex `"[hello]\w+rld"`. Circle all strings that the regex fully matches.
   `"helloworld"` `"hworld"` `"hellowrld"` `"otherworld"`

2. Consider the regex `"[a-d0-9]+\\[^\d]{2,5}"`. Circle all strings that the regex fully matches. `"61b\party"` `"9a9\\+hue"`

## 3 Fall '14, MT2: Reverse Traversal

Consider a binary search tree whose labels are each one capital letter, and assume that a BFS traversal yields the sequence "GENIUS". What is the preorder traversal of this tree? Draw the tree itself.
The preorder traversal also yields GENIUS. The tree is




## 4 Fall '14, MT2: BST —> Heaps

Suppose we have a complete binary tree, X, that is not a heap, and we wish to heapify it. The obvious thing to do is just insert all nodes of X into a new binary heap Y . This works, but it doubles the space requirement. Suppose we'd prefer to heapify in place. Which of the following procedures will convert X into a heap (containing all of the original values)? To each, either answer "yes" or give a counter-example.

1. Sink (heapify down) all nodes in level order (first the root, then its left child, then the right child of the root, etc).
   No. [Here and below, we assume a max heap, which we present as an array.] Consider 1, 2, 3, 12, 13, 4, 5. This will bring 3 to the top, instead of 13.

2. Swim (heapify up) all nodes in level order.
   Yes. This is just what you **do** to add elements to the heap.

3. Swim all nodes in reverse level order.
   No. Consider 1, 6, 7, 2, 3, 4, 5. This will leave 1 above 4 and 5.

4. Sink all nodes in reverse level order.
   Yes. At each step, the current node will be the root of a subtree all of whose nodes, with the possible exception of the current node, will obey the heap property (inductive assumption). In **this case**, heapifying down will necessarily work, just as when you swap a (small) node **for** the root in the remove first operation.

# 5 Graph Theory Basics

1. Provide a brief description of how to solve each of the following graph problems efficiently. Provide a worst case runtime bound in $\Theta$ notation in terms of V and E. Let WUG mean weighted, undirected graph.

   i Find a path from node s to node t in a strongly connected, directed graph.

   Run DFS starting from node s. Terminate at t, saving back pointers. Comment: BFS is also acceptable. Running Dijkstra's or A* is unnecessary as we don't want a shortest path. Runtime: $\Theta(|V| + |E|)$. $\Theta(|E|)$ is also okay because it's connected.

   ii Determine if a cycle exists in a directed graph.

   Run DFS. If a node encountered is in the call stack (can be maintained as a set), there is a cycle. Runtime: $\Theta(|V| + |E|)$

2. **Extra for Experts:** A directed acyclic graph $G$ is *semiconnected* if for any vertices $A$ and $B$ there is either a path from $A$ to $B$ or a path from $B$ to $A$. Show that $G$ is semiconnected if and only if there is a directed path that visits all of the vertices of $G$.