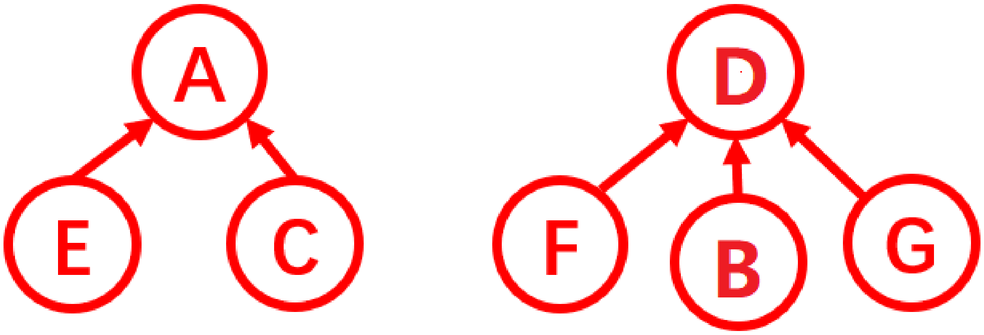
**CS 101 Fall 2020 - Quiz 7-A**
**11/17/2020 - 15 Minutes** **Name:** **ID number:**

**Problem 1(2+3pts)**: Consider a disjoint set with both path compression and union-by-size optimization. When two trees have the same height, the set specified first in the union will be the root of the merged set. The following operations are done:

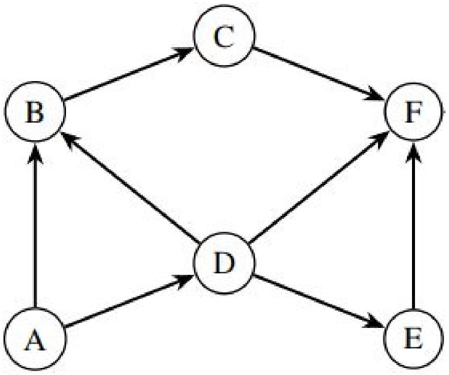
$$union(A, C), union(D, B), union(F, B), find(F), union(E, A), find(G), union(F, G)$$

Please write down the result for each *find* operation appeared in the above operations, and draw the disjoint set tree after all operations above is finished.



D; G

**Problem 2(3pts)**:
Run DFS the following graph starting with vertex A. Please write down the result. Whenever there is a choice of which node to visit next, follow the alphabetical order.



A B C F D E

**CS 101 Fall 2020 - Quiz 7-A**

**11/17/2020 - 15 Minutes** **Name:** **ID number:**

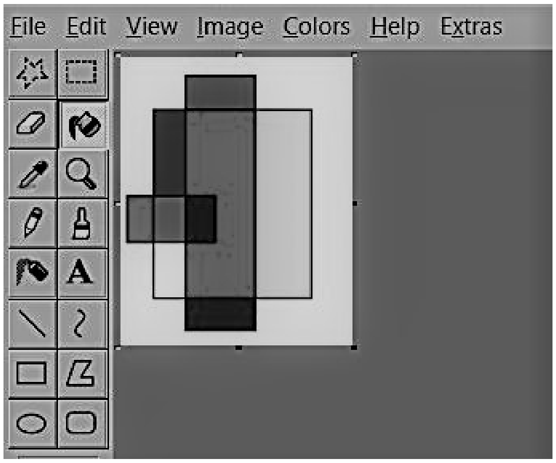
---

**Problem 3(7pts)**: MS Paint

There is an image of $m \times n$ pixels which is mono-color (only contains black or white). To make the image colorful, you decided to use the 🪣 tool in **Microsoft Paint** to paint its **white** part. Note that **the black part should *NOT* be painted with this tool.**

This tool has the following property: if we use this tool to color one **white** pixel with position (x, y), its neighbour pixel (one or more of: (x-1, y), (x+1, y), (x, y-1), (x, y+1)) will also be colored **if they are originally white**; also, this property will keep executing on **every** pixel that is colored just now (i.e, the origin pixel, together with its neighbor pixels that are colored), until no more pixel can be colored.

Please design an algorithm to count how many different types of color can be used at most. Briefly explain your algorithm with natural language. For the provided example, the number is 10.



The input image can be seen as a undirected graph, each pixel as a vertex, and there are edges between each white pixel and each of its neighbor. Traverse this graph, each time we see a unvisited white pixel, do DFS or BFS starting with this vertex, until all nodes are visited. Finally count how many times DFS is done, and the number is exactly the answer.