Jake Schinto

Homework #7

1. a.

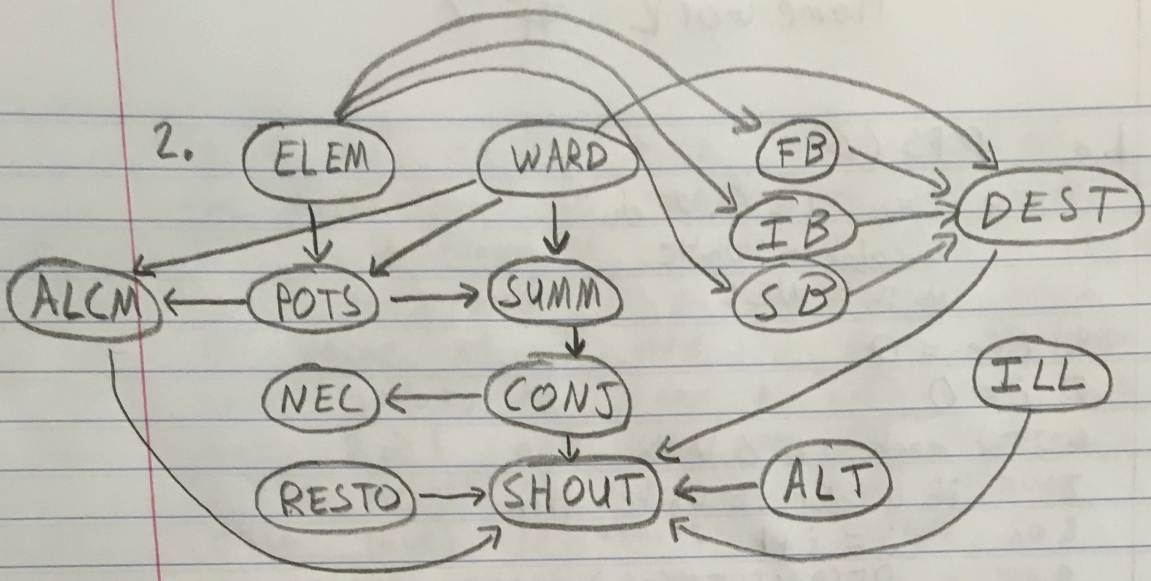
```
DFS(G)
1.  for each u ∈ G.V do
2.      u.color = white
3.      u.π = NIL
4.  time = 0
5.  i = 0
6.  for each u ∈ G.V do
7.      if u.color == white then
8.          i = i+1
9.          DFS_VISIT(G,u)
```

```
DFS_VISIT(G,u)
1.  time = time + 1
2.  u.d = time
3.  u.color = gray
4.  u.component = i
5.  for each v ∈ G.Adj[u] do
6.      if v.color == white then
7.          v.π = u
8.          DFS_VISIT(G,v)
9.  u.color = black
10. time = time + 1
11. u.f = time
```

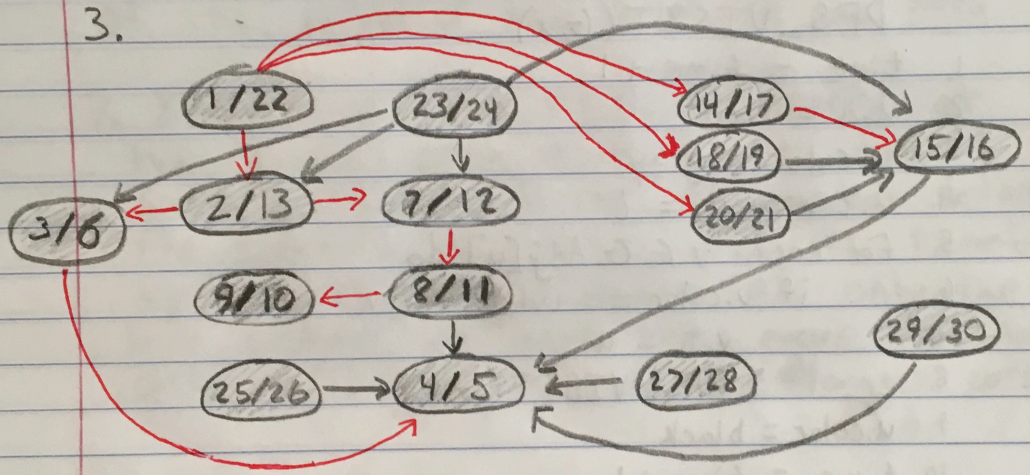
b. The algorithm is correct because the outer loop goes and fully explores the verticies that it visits. Therefore, when the loop finds an unexplored vertex, it is not connected to any previously explored and must be a new component.

c. The new component adds $\Theta(V)$ to DFS and $\Theta(1)$ to DFS_VISIT, since the runtime before was $\Theta(V+E)$, adding $\Theta(V)+\Theta(1)$ to it doesn't change the runtime. It is still $\Theta(V+E)$

2.



3.



4. Tree: (ELEM, POTS), (POTS, SUMM)
Back: NONE
Forward: NONE
CROSS: (ALT, SHOUT), (ILL, SHOUT)
all gray edges in my graph are cross

5. ILL, ALT, RESTO, WARD, ELEM, SB, IB, FB, DEST, POTS, SUMM, CONJ, NEC, ALCM, SHOUT

You use topological sorting by creating a linked list of the classes based on when they finish in DFS. This leaves them in order from last to be finished to the earliest.