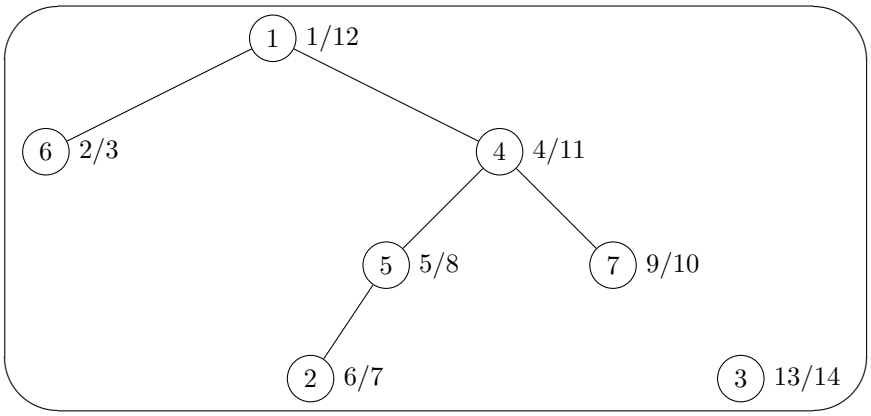
CSC263H1 Assignment 6

Jiatao Xiang, Xu Wang, Huakun Shen

March 28th, 2019

Question 1

a.



b. Forward edge: 2, Back edge: 0, Cross edge: 5.

c. From part b we know that there is 0 back edge in the DFS tree of the graph.
By the White-Path Theorem, we know, a graph G has a cycle $\Leftrightarrow$ DFS of G has a back edge.
Since there is not back edge in the DFS tree, there is no cycle in G, which means there cannot be two courses that are prerequisites of each other, and it is possible to take all the courses in a sequencial order that satisfies all the prerequisite requirements.

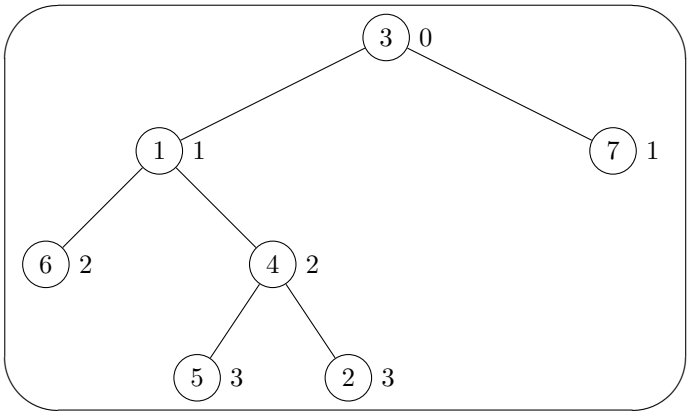
Detailed Proof:
To prove that it is possible to take all the courses in a sequential order that satisfies all the prerequisite requirements, we have to prove that there are not two courses that are prerequisites of each other.
Suppose u and v are two nodes in G that represent 2 courses, then u is a prerequisite of v if and only if $u.f > v.f$. So it is suffice to show that for every edge $(u, v) \in E$, $u.f > v.f$.

Proof. To prove that $\forall$ edge $(u, v) \in E$, $u.f > v.f$.
Case 1: (u, v) is a back edge
We know that, in the DFS forest above, there is no back edge, i.e. there is not cycle, so that there is not 2 courses being the prerequisites of each other. (by white-path theorem)
Case 2: (u, v) is a forward edge or tree edge
Then v is a descendant of u. By parenthesis theorem, $u.d < v.d < v.f < u.f \Rightarrow u.f > v.f$
Case 3: (u, v) is a cross edge
Then u and v are not descendants of one another, i.e. there is no prerequisite-relationship between u and v, which means the order doesn't matter for these two courses. So we can make $u.f > v.f$ (i.e. take u before v).

□

d. $3 \to 1 \to 4 \to 7 \to 5 \to 2 \to 6$, in this ordering, each vertex's finish time is greater than the finish time of its next node.

e.



Inside the node is the value of the node. Beside each node is the distance from node 3.

Question 2

Description:

First, we construct an adjacency list using all equality constraints. We also put all variables that not appeared in the equality constraints into the adjacency list. Since there are m constraints and n variables, constructing adjacency list totally takes $\mathcal{O}(m+n)$.

```
# m represents set of constraints, n is set of variables
def construct(m, n):
    # Construct an adjacency list using equality constraints of all variables;
    # Store all inequality constraints into a list called list_l;
    make_adjacency_list(m-list_l, n)
    value = 0
    for each v in n:
        if color[v] is white
            # When each variable is explored, we create a field in it to store
            # the value.
            perform BFS and set each node to value
            value ++
    for each x_i != x_j in list_l:
        if x_i.value == x_j.value:
            return NIL # contradiction!
```

BFS operation will change color[v] to black when it's fully explored and we loop over each line in the adjacency in order, therefore, after the for loop ends, every node in the list will be fully explored only once. This step takes $\mathcal{O}(m+n)$. At this stage, suppose there are k BFS trees found, each of them is a connected graph where each node in it equal to each other. Then, we check inequality constraints, as long as we find a $x_i \neq x_j$ and they belongs to the same tree, then it's a contradiction and we return NIL. This step takes $\mathcal{O}(m)$. Otherwise, since we have assigned proper values for each variable, therefore, the problem is solved.

Question 3

Lemma: if the vertex is in the cycle, it must have at least two edges.

Intuitively, let u be a vertex that is in the cycle, then there must be an edge "out" of this vertex and an edge "into" this vertex, which shows that there must be at least two edges. This is stated in MST handout posted on course website.

Now prove by contradiction:

suppose there is a MST that contain e_{max}.

According to MST construction theorem, this would only happen when e_{max} is the only path that connect one part of the graph to the other part, but every edge is in a cycle. Thus, contradiction! Since we have at least

two edges for each node to connect to the rest of the graph, then we would choose the other edge instead of e_{max}, because the weight of the other edge must be less than or equal to the weight of e_{max}.
Thus, we conclude that there is no MST contain e_{max}.