1.017) Prove: $x^{\log_b y} = y^{\log_b x}$ Derrick DeBose

we will first $\log_b(\,)$ each equation

$$\log_b(x^{\log_b y}) = \log_b(y^{\log_b x})$$

using point 2 from Lemma 1

$$\log_b x \cdot \log_b y = \log_b y \cdot \log_b x \quad \checkmark$$

Because left side is equivalent to right side

$$x^{\log_b y} = y^{\log_b x}$$

Baase & Van Gelder

Derrick DeBose

1.02) Lemma 1.5 - A function $f \in O(g)$ if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = c < \infty$, including the case in which the limit is 0.

Lemma 1.7 - Function $f \in \Omega(g)$ if $\lim_{n \to \infty} \frac{f(n)}{g(n)} > 0$, including the case in which the limit is ∞.

Lemma 1.8 - Function $f \in \Theta(g)$ if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = c$ for some constant c such that $0 < c < \infty$.

$p(n) = a_k n^k + a_{k-1} n^{k-1} + \dots + a_1 n + a_0$

Show $p(n) = \Theta(n^k)$

let $f(n) = p(n)$ & $g(n) = n^k$

$$\lim_{n \to \infty} \frac{a_k n^k + a_{k-1} n^{k-1} + \dots + a_1 n + a_0}{n^k}$$

For large n, the lower order terms are irrelevent ($n^{k-1} \dots n$). So we can reduce the equation.

$$\lim_{n \to \infty} \frac{a_k n^k}{n^k} \Rightarrow \lim_{n \to \infty} a_k = \boxed{a_k}$$

Since $a_k > 0$ and we know that a_k is a constant, we can prove $p(n) = \Theta(n^k)$ using Lemma 1.8.

$f \in \Theta(g)$ if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = c$, where $0 < c < \infty$

Since $f \in \Theta(g)$, $\boxed{p(n) = \Theta(n^k)}$

Derrick DeBose

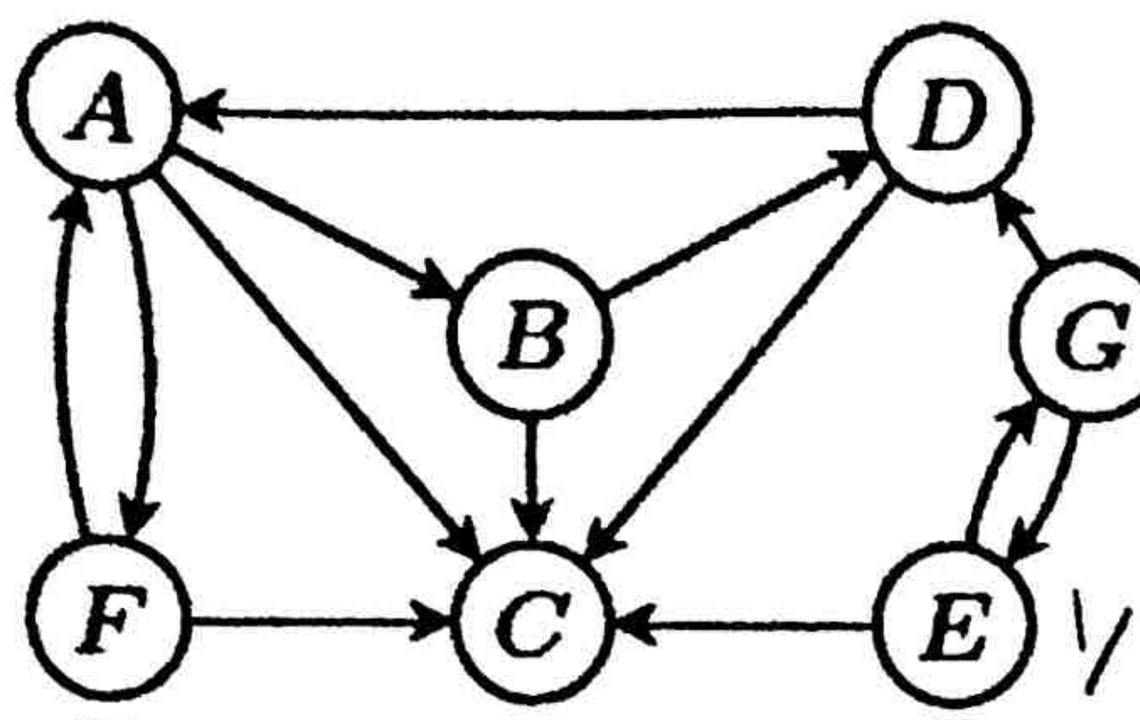
Problem 1.03

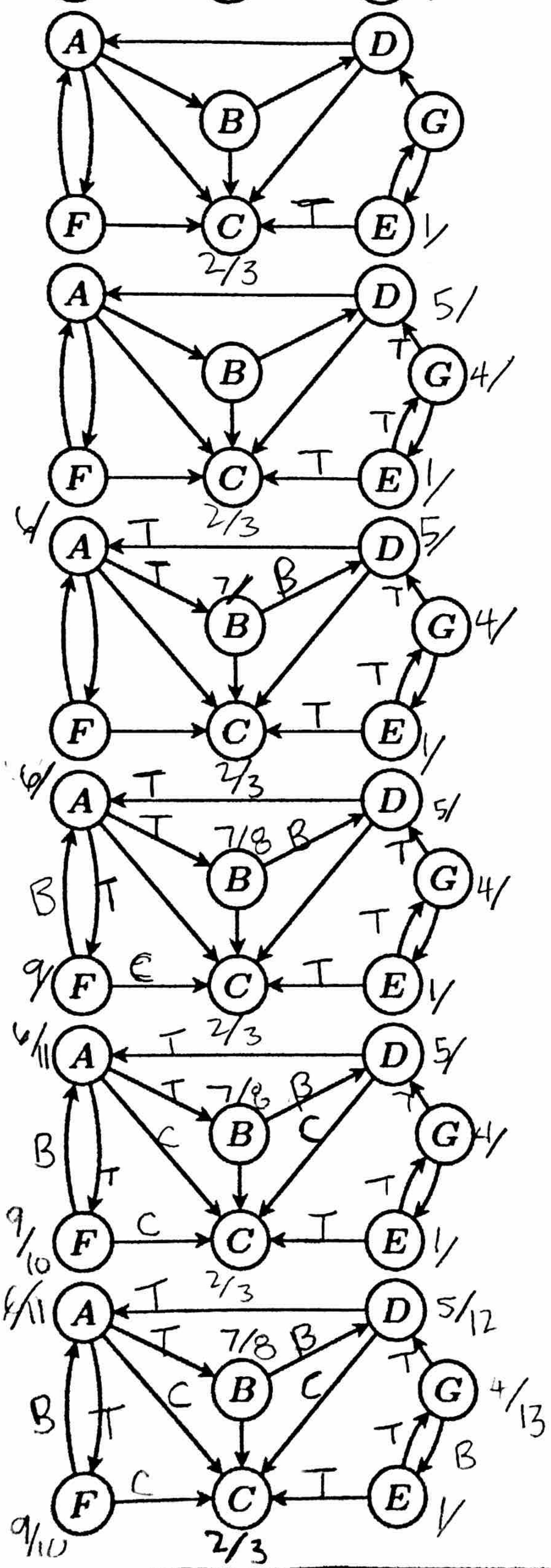
```
define set 1 and set 2 and total coin count
define fake coin check
put lower bound n/2  coins in set 1
put lower bound n/2 different coins in set 2
while (set 1 size and set 2 size != 1){
	if (total coin count is odd){
		measure 2 sets in balance scale
		if (weight is equal) {
			return fake coin check
		}
		else if(set 1 weight < set 2 weight) {
			put lower bound n/2  set 1 coins in set 1
			put lower bound n/2 set 1 different coins in set 2
			if (set 1 coin count is odd){
				set fake coin check to odd coin out the sets
			}
		}
		else{
			put lower bound n/2  set 2 coins in set 1
			put lower bound n/2 set 2 different coins in set 2
			if (set 1 coin count is odd){
				set fake coin check to odd coin out the sets
			}
		}
	}
	else{
		measure 2 sets in balance scale
		if(set 1 weight < set 2 weight){
			put lower bound n/2  set 1 coins in set 1
			put lower bound n/2 set 1 different coins in set 2
			if (set 1 coin count is odd){
				set fake coin check to odd coin out the sets
			}
		}
		else{
			put lower bound n/2  set 2 coins in set 1
			put lower bound n/2 set 2 different coins in set 2
			if (set 1 coin count is odd){
				set fake coin check to odd coin out the sets
			}
		}
	}
}
measure set 1 last coin and set 2 last coin
if (set 1 weight < set 2 weight){
	return set 1 lasts coin
}
else{
	return set 2 last coin
}
```

Best case: 2 weighings. Worst case: 6 weighings.
The algorithm has a $\Theta(\log n)$ running time because it reduces the problem in half each step.

Derrick DeBose

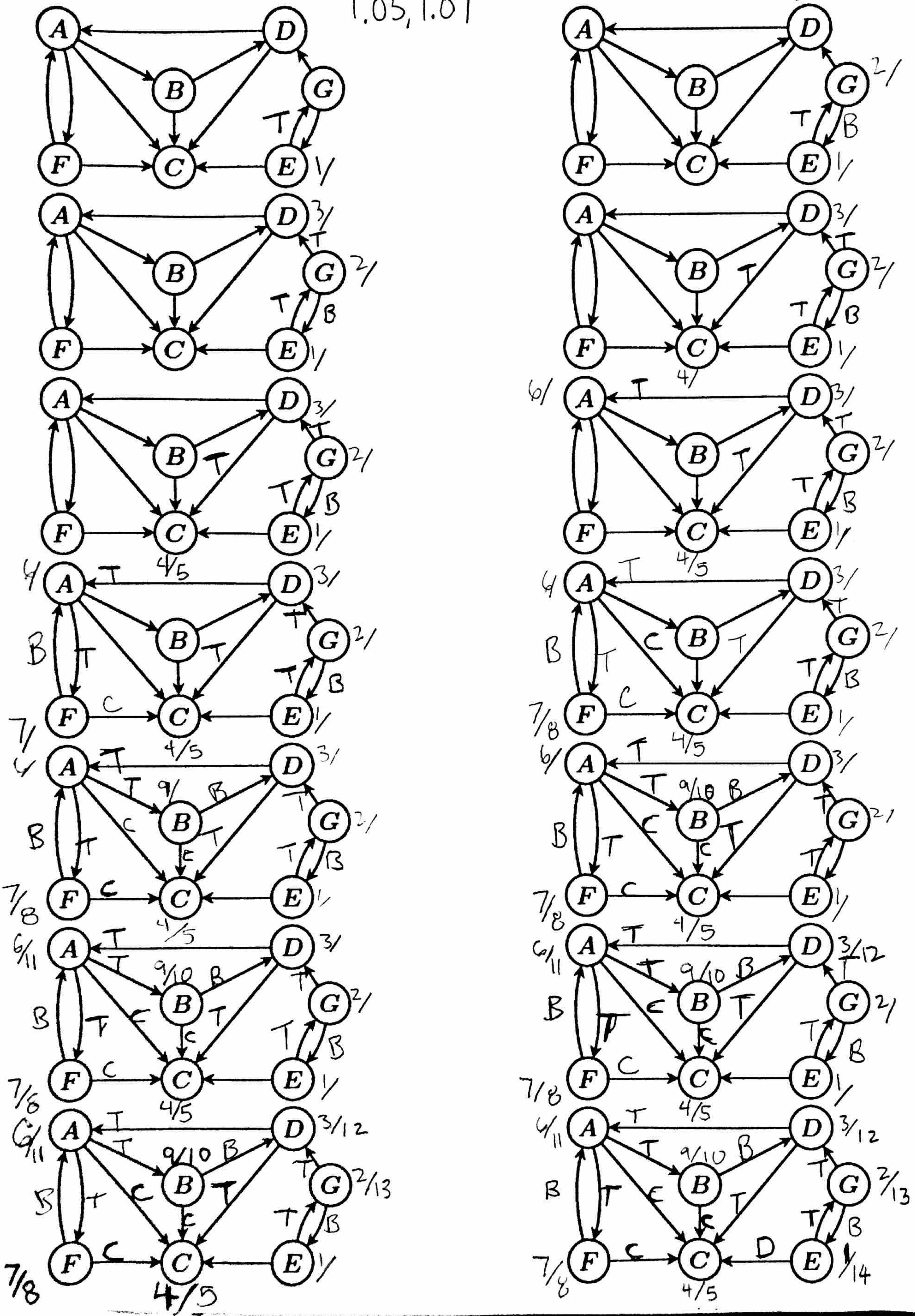
1.04, 1.06





1.05, 1.07

Derrick DeBose



Derrick DuBose

1.08

1) If edge vw is a backedge then vertex w is $\boxed{\text{grey}}$ when edge vw is checked

Baase & Van Gelder
Theorem 7.1
3. d. vw is a back edge if and only if

active(v) ⊂ active(w)

Definition 7.15

active(v) = discoverTime[v], ..., finishTime[v]

assuming active(v) ⊂ active(w), that means that

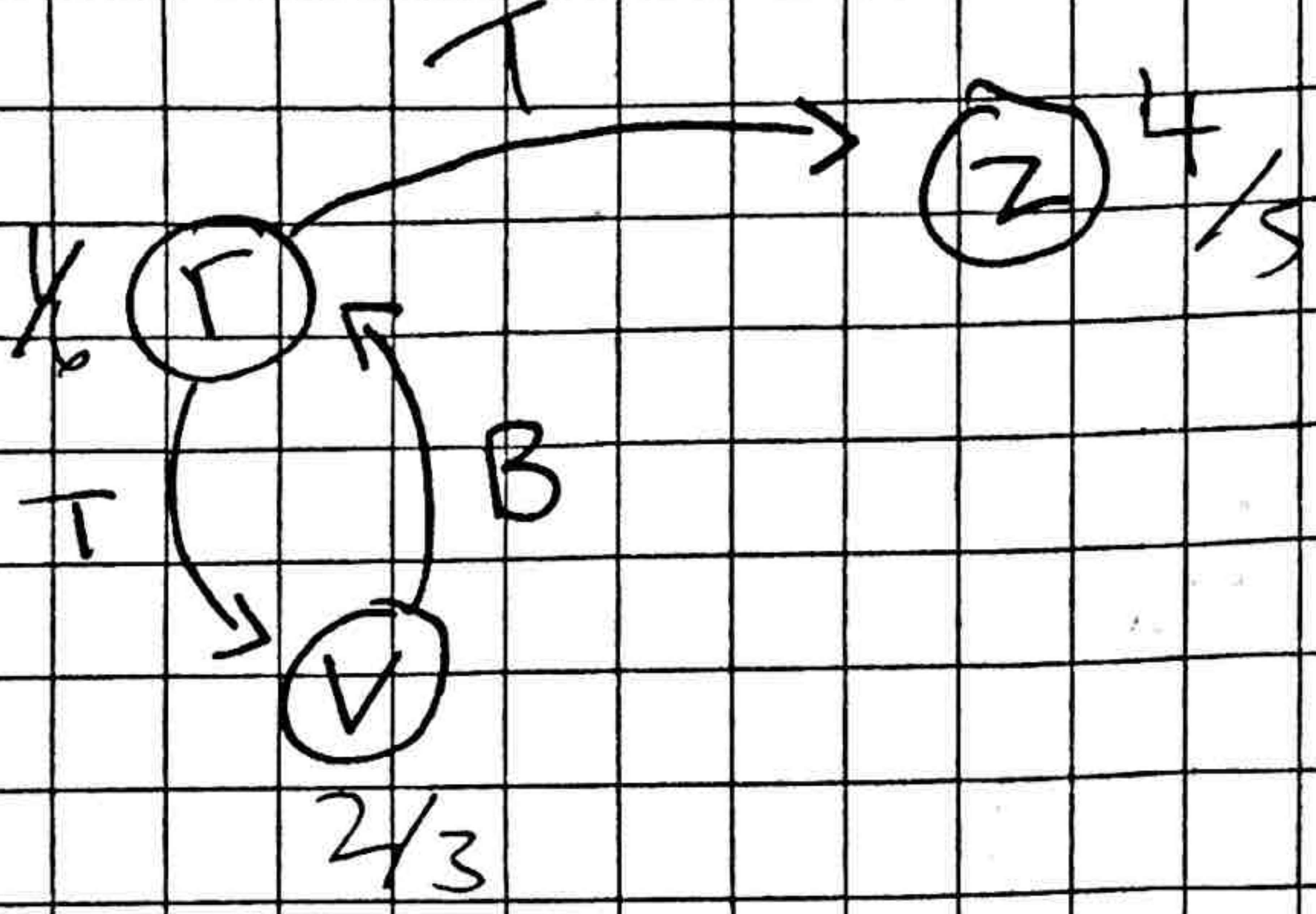
discoverTime[w], ..., discoverTime[v], ..., finishTime[v], ..., finishTime[w]

w must be grey when finding a back edge vw because Theorem 7.1 states a back edge must have active(v) ⊂ active(w).

When Definition 7.15 is applied to Theorem 7.1 we can see vertex v is discovered and finished, after vertex w is discovered and before vertex w is finished. This means vertex w can not be black or white, so it must be $\boxed{\text{grey}}$.

Derrick DeBose

1.09)



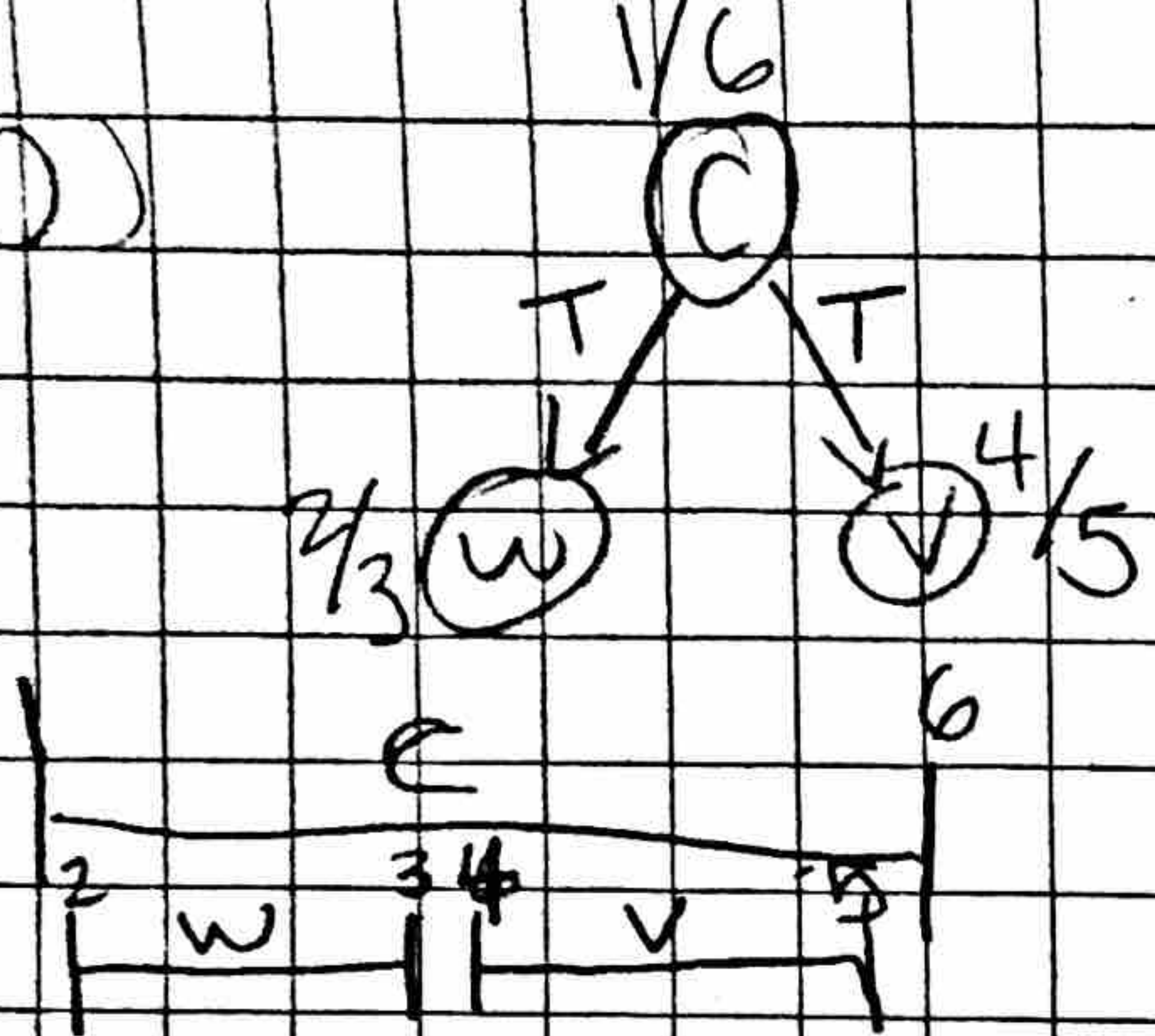
Given this graph, ~~vertex~~ r is the root that travels to vertex v. Given that at time 2, there is one edge that is discovered.

DFS backtracks from vertex v before all the vertices vertices that can be reached from v.

There is a path from v→z but when vertexr is the root the back edge from v→r will not taken. So vertex v backtracks before reaching vertex z.

Derrick DuBose

1.10)



Vertex c is the least common ancestor

There is a path from c to w and a path from c to v.

The 2 vertices w & v are connected to root c and there always exactley one path from the root to any vertex.

The example given causes vertex w to back track to the root c before exploring to vertex v. This means there is 2 distinct paths that have no edges in common.