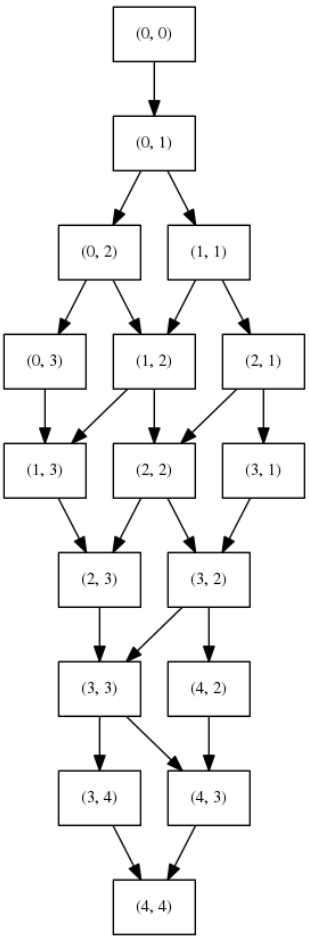
DS Coursework

Jacob Essex
s1040340

Q1

I assume that the starting time $(0, 0)$ is some point before either of the first events for p_1 or p_2 occur. This is slightly more general than the notation used in some of the slides in which it is assumed that both of the first events have occurred. Should this actually be what is wanted then as the below graph is more general the point at which the first two events have occurred is $(1, 1)$



Q2

To avoid starvation an algorithm needs to be fair (i.e. requests for a lock are honored in the order that they are made). This prevents starvation. A system that is liable to have starvation is one that always grants the request of the most recent request for a mutex over that of older requests.

In a three process system where processes requires and release the mutex in a loop, this mutex algorithm will lead to two of the processes being granted access to the critical section and the third becoming starved. This is because the third process will always have the oldest request for the critical section one of the other two processes will always have the more recent request.

Q3

The response data is of fixed size and is represented using the following tuple

$$(\text{sum of all p.f seen}, \text{max of all p.f seen}, \text{number of processors seen})$$

The algorithm is as follows:

The root node r sends a request to all of its children in the tree T for the data. All children then send the request onwards to their children. This happens recursively and eventually message has will propagate to all nodes. This takes $O(n)$ time as the message is a of a fixed length and in the worst case the tree is a list where each node has 0 or 1 children.

When a leaf node receives the request for the data it then sends $(p.f, p.f, 1)$ to its parent node.

For a node p, such that $p \neq r$, on receipt of data $(sum_i, max_i, count_i)$ from all its n children, the node p sends sends the following data to its parents

$$(p.f + \sum_{i=0}^{n} sum_i, \text{MAX}(p.f, max_1, max_2, ...max_n), 1 + \sum_{i}^{n} count_i)$$

For the node r on receipt of data from all its n children, r does the same calculation as above:

$$(r.f + \sum_{i=0}^{n} sum_i, \text{MAX}(r.f, max_1, max_2, ...max_n), 1 + \sum_{i}^{n} count_i)$$

So it then holds hold $(\text{sum}, \text{max}, \text{count})$. It then preforms the following operation on its tuple to product a new tuple

$$\text{result} = (\text{avg}, \text{max}) = (\text{sum}/\text{count}, \text{max})$$

The response message length is also fixed, so the cost of sending a message is fixed. Again in the worst case, the minimum spanning tree represents a list of processors so in this case $O(n)$ messages need to be sent sequentially taking $O(n)$ time.

This new tuple is the required result, although it needs to be propgated to all nodes. This can be done by r sending the result tuple to all children, and then all reciving no leaf nodes sending the result to their children. This will propgate the result through all nodes in T.

Again for the same reasons as before this takes $O(n)$ time.

In total the time taken for this algorithm is $O(n) + O(n) + O(n) = O(n)$

Q4

The diameter of the network is defined by the longest shortest path. In the case the diameter of the weighted graph this is: $d \to h \to l \to m$

If each edge had a weight of 1 then the following paths of cost 5 would realise this: $e \to a \to c \to g \to l \to m$ and $e \to a \to c \to f \to i \to k$

Of course in both cases as the graph is undirected the reverse paths would also give the same results