Distributed Systems Assignment

s1140740

2.1

Proof: If V is a vector clock, prove that $a \rightarrow b \iff V(a) \leq V(b)$.

$a \rightarrow b \implies V(a) \leq V(b)$

There are three possibilities:

1. Event b was a local event of a process i, by definition $V(b)[i] += V(b)[i] + 1$
2. Event b was a send message event of a process i.
3. Event b was a receive message event of a process i.

$V(a) \leq V(b) \implies a \rightarrow b$

2.2

Liveness means that every request for critical section (CS) is eventually granted.

When a process i wants to request access to CS, it will send a REQUEST message with their id and current timestamp to all processes and add the message to its own queue. Assuming all channels are FIFO and there are no failures in channels all processes will send a timestamped REPLY message and add the message received into its own queue.

Process `i` decides to use the resource, it sends a `REQUEST` message, puts its own request into its queue and gets a `REPLY` message from every other process (no channel failures assumption). Now let's assume that process `i`'s request never gets satisfied. This means that its request never reaches the top of its queue (if it did, the request would get satisfied as per the algorithm). This is clearly impossible under the assumption that all accesses are *not indefinite*. This is because every time a process finishes accessing the resource it sends a `RELEASE` message to all other processes at which point those processes pop that process's request from their queue. This is guaranteed to happen due to the no process and channel failures assumption.

Induction on the position of the request in the queue.

Base case:

Request is at the top of the queue => it gets satisfied.

Induction hypothesis: If our request eventually gets satisfied at position k, it also gets satisfied if we are at position $k+1$.

Our request is at position $k+1$. Let the request at first position belong to process i and let us call the request R_i. There are three options:

1. Process i is currently accessing the resource. Since we assume processes do not fail, this means that when it is done it will send a `RELEASE` message and we will remove R_i from our queue.
2. Process `i` has finished accessing the resource but we have not yet received the `RELEASE` message. Channels do not fail so we will eventually receive the message and remove R_i from our queue.
3. Process `i` has not started accessing the resource. Since process `i` is at the first position in the queue that means all the previous processes accessing the resource are finished, otherwise we would have their requests in our queue before R_i (we can only remove them once we receive a `RELEASE` message).

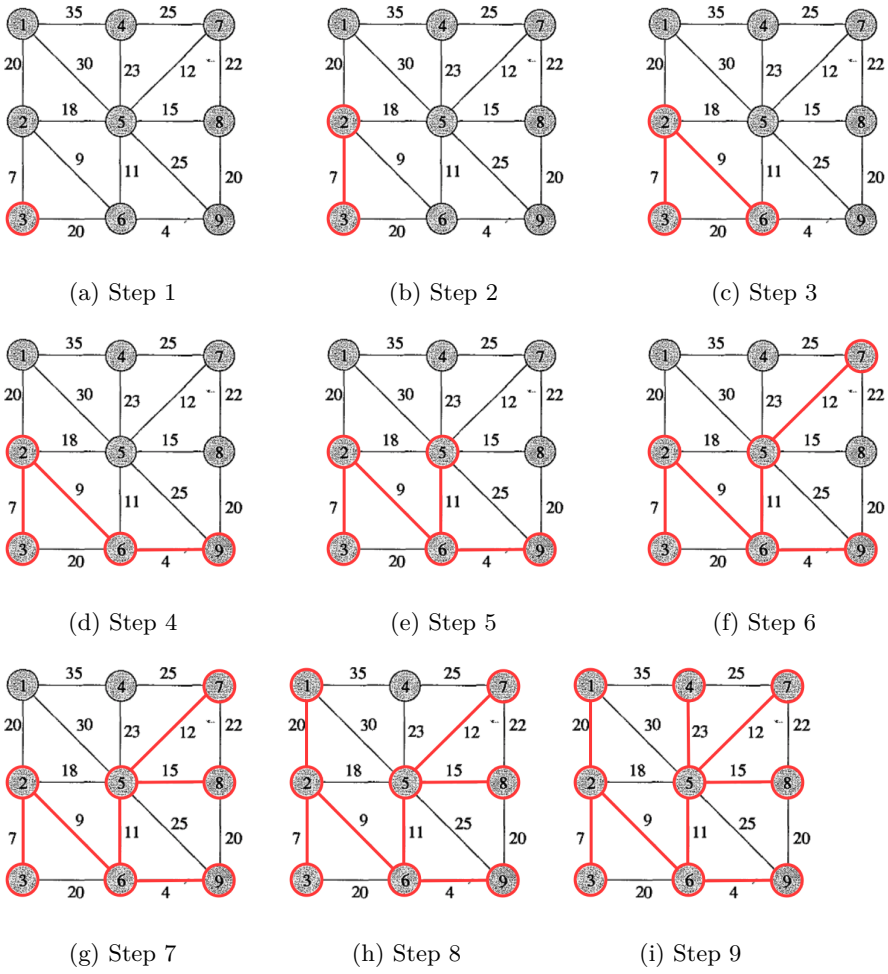
because it has not received the `RELEASE` message of the previous process accessing the resource but we have. Again, channels do not fail so it will eventually receive the message and start its CS.

2.3

The weighted diameter of this graph is 7. The path realising this diameter is $A \to C \to E \to G \to H$.

If the graph was unweighted, the diameter would be 4 and the corresponding path would be $A \to C \to F \to G \to I$.

2.4



(a) Step 1 (b) Step 2 (c) Step 3

(d) Step 4 (e) Step 5 (f) Step 6

(g) Step 7 (h) Step 8 (i) Step 9

Figure 1: Prim's algorithm