CRITICAL PATH ANALYSIS

yay → this is actually how we use this stuff IN REAL LIFE !!

AWESOME

A good way to think about this section of digraphs is to consider the quickest way or shortest path to getting where you want to get to.

Sounds sensible ....

Think about cooking something

- You must follow the recipe steps in a certain order for some bits.
- For other things, they might be able to be done at any time

What is a good example M^rs V.?

OK, if I want to cook some marinated meat and roast vegetables for dinner, it makes sense that ...

1. you have to prepare the marinade first
2. then leave the meat to marinate
3. then cook the meat.

(has to be this order)

The vegies can be prepared anytime and then cooked → as long as done by the time the meat is done.

Sorry to the vegetarians out there !!

So steps ① and ② are called predecessors to step ③ as they must happen before ③ happens.

The good news is you are going to learn how to represent the flow of activities using a network diagram.
First step is to construct a table (also called a dependency list) to make sure everything is included along with appropriate times.

Activity Letter	Activity	Predecessor (Immediate)	Time
A	prepare marinade	—	20 mins
B	marinade meat	A	min 5 hours
C	cook meat	B	5 hours
D	prepare vegies	—	30 mins
E	cook vegies	D	1 hour
F	eat everything	B, E	30 mins

I'm thinking slow roasted marinated lamb hm

Now we can consider how to minimise the amount of time required and thus determine the critical path

If you just added up all the times without overlapping any activities you can see it would take:

↳ 20 mins + 5 h + 5 h + 30 mins + 1 h + 30 mins

= 12 hrs 20 mins

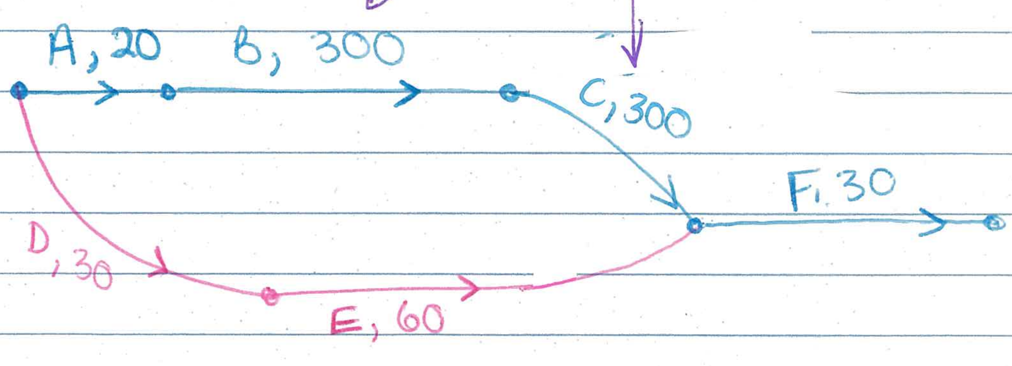
FORWARD SCANNING → this helps determine the earliest start time (EST) for each activity thus earliest finish time overall

record on the network ←

* Can't start an activity before a predecessor has finished * EST for 1st activity is 0

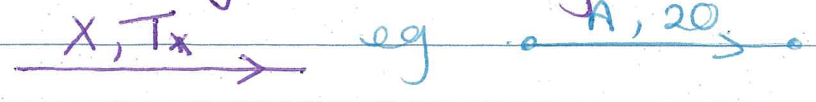
OK, lets try and construct a network for cooking the meat!

changed to minutes so all in same units



① Find activities with no predecessors (A and D here and join them to a starting vertex

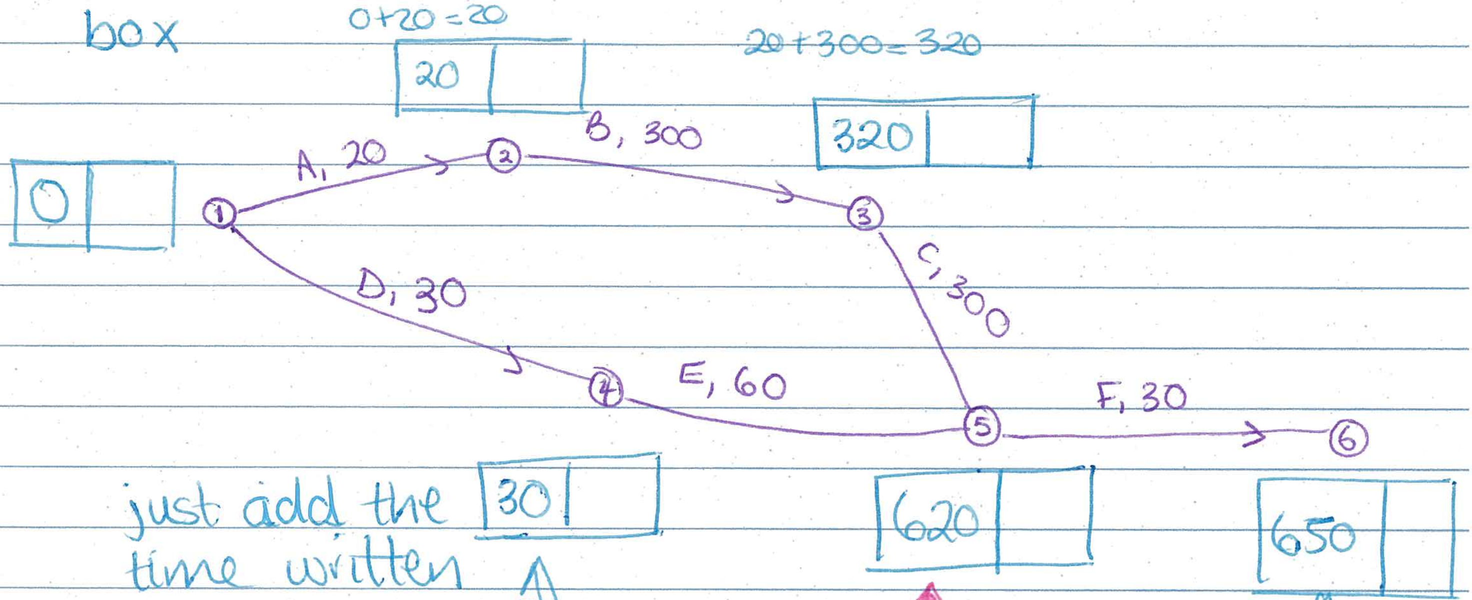
② Label each edge with activity letter and time



(3) Add an edge for each activity making sure it matches the predecessor information

(4) Record the earliest start time (EST_x) for each activity in box next to vertex. (The text book uses little houses (is v. cute) instead)

EST in left hand box



just add the time written on the edge to the value in the previous box (0+30)

why not 30+60 = 90 ??

Because you can't actually start F (eating) until everything is cooked.
So this number is the highest of the possible paths.

earliest completion

* We can see earliest finish time is 650 mins or 10 hr 50 mins

Ahhh OK!

Interpreting this information tells us that:

① the marinade prepartion process and meat cooking can not be delayed if you want process to finish at 650 mins.

② preparation and cooking of the vegies can be delayed without affecting end time → but not indefinitely .....

So A-B-C-F is the CRITICAL PATH

FLOAT TIME → this is the difference in time between critical and non critical paths.

↕

because you can move the timing of non critical activities to help reduce costs

LATEST START TIME → latest time non critical activities can start without delaying project completion

To calculate float time compare the times with the maximum time for that section then subtract the smaller from the larger number.

Back to our eg.

A-B-C = 620 mins

D-E = 90 mins ∴ float time for DE activity = 620 - 90 = 530 mins

To calculate **latest start time** look at earliest completion on critical path and subtract activity duration

Back to our eg.

latest time E can start is
620 − 60 = 560 mins

latest time D can start is
560 − 30 = 530 mins

* This process is SCHEDULING. There are many jobs in industry for people with strong scheduling skills. *

Notes:

① 2 nodes | vertices can be connected by 1 edge only

② Each activity is resprented by 1 edge only

For critical path analysis.

Try Ex 15B questions OK ?!