Critical Time Computation
Math 203

Our text slightly misstates the meaning of critical time for an order requirement digraph by excluding isolated tasks from the set of maximal paths. Furthermore, the way it says to compute the critical time of a project suffers from a problem called the combinatorial explosion. What this means is that problems can get too big, too fast, making them unmanageable. This handout provides a different approach that makes even large problems very manageable.

What do we mean by unmanageable? The largest number of maximal paths that an order-requirement digraph with n vertices can have is in the vicinity of $3^{\frac{n}{3}}$. If an order-requirement digraph has 30 vertices, then it can have up to 59049 maximal paths! Each path could contain 10 vertices, requiring 9 additions to find the weight of the path, meaning we could be faced with 531,441 additions to find the critical time. On the other hand, if done efficiently, the critical time for a project with 30 tasks never requires more than 29 additions. Which would you rather do? (In real life, something like a major construction project can have a hundred or more tasks. Even a fast computer could be overwhelmed if it had to use the method given in the book.) To provide a more efficient approach, we start with the following definition of forward critical time for each task:

Definition. If we have a project with clearly defined order-requirements and task times, for any task in the project, the task's forward critical time is the minimum amount of time that can elapse between the start of that task and the completion of the entire project.

If we have a weighted order-requirement digraph for the project, this definition is equivalent to saying the task's forward critical time is the largest weight we get for any path that <u>starts</u> with that task.

Definition. The critical time for the project is the largest forward critical time present among the tasks within the project.

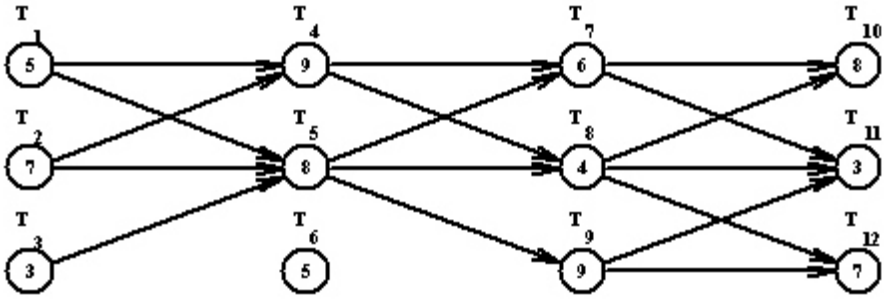
The nice thing here is that there is a simple, efficient procedure for finding tasks' forward critical times:

For any task in a project, to find its forward critical time:

1. For any task which is a sink, including any isolated tasks, its forward critical time is just the amount of time it takes to do that task.

2. For any task A which is at the start of one or more arcs, wait until you find the forward critical time for each of the tasks at the ends of those arcs. Then find the largest of those forward critical times and add to that the time it takes to do task A to get the forward critical time for A.

Example: In the following project with 12 tasks, find the forward critical time for each task, and the critical time for the project.



We see that T_6, T_{10}, T_{11} and T_{12} are sinks, so their forward critical times (we'll abbreviate this FCT) are 5, 8, 3 and 7 respectively. Once we've found those, we can find the FCTs of T_7, T_8 and T_9. For T_7, we need to look at the FCTs of T_{10} and T_{11}. The larger time is 8 at T_{10}, so the FCT for T_7 is 6+8, or 14. For T_8, with arcs going to T_{10}, T_{11} and T_{12}, we compare the FCTs of all three. The largest is again the 8 from T_{10}, so the FCT for T_8 is 4+8 = 12. From T_9, arcs go to T_{11} and T_{12}, and T_{12} has the larger FCT of 7, so the FCT of T_9 is 9+7 = 16. Now that we know those, we can find the FCTs of T_4 and T_5. For T_4, the arcs point to T_7 and T_8, and the maximum FCT of those is 14, so the FCT of T_4 is 14+9 = 23. For T_5, the maximum FCT at the vertices at the end of its arcs is the 16 at T_9, so the FCT at T_5 is 8+16 = 24. Finally, we're ready to find the FCTs of T_1, T_2 and T_3. All three of them have arcs ending at T_5 where the FCT is 24, T_1 and T_2 also have arcs ending at T_4 but T_4's FCT is lower, so they all use the FCT at T_5 plus their own task time. That means the FCT at T_1 is 5+24 = 29, at T_2 it's 7+24 = 31, and at T_3 the FCT is 3+24 = 27.

Since the largest FCT for any task is 31 at T_2, the entire project has a critical time of 31 and a critical path starting at T_2.

To find the critical path(s) once all the task FCTs are known:

1. Start at a task whose FCT is the critical time for the project.
2. If you're at vertex *A* which is not a sink, follow an arc that starts at *A* and ends at a vertex *B* such that (FCT for *B*) = (FCT for *A*) − (Time required to do *A*). Repeat this step until you reach a sink.
3. If you had multiple valid choices to make in steps 1 or 2, that means there are multiple critical paths. To find all critical paths, explore all choices that follow the above rules.

Following this approach, the only critical path for the project above is $T_2 \rightarrow T_5 \rightarrow T_9 \rightarrow T_{12}$.

Incidentally, the above project has 32 maximal paths, requiring 93 additions using the book's approach.