More Scheduling

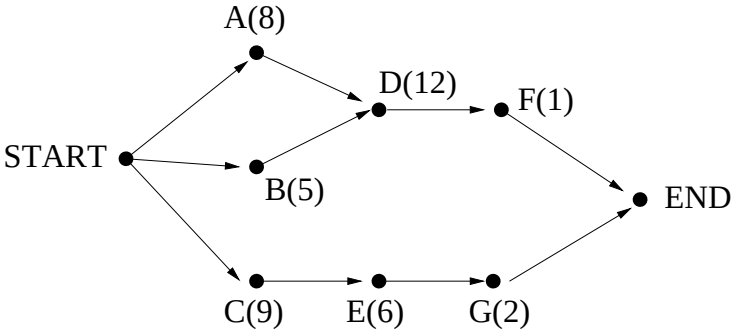
Importance of tasks when scheduling-
Note that some tasks are independent
Others have many tasks after them
These tasks are "critical"- it is important to do them early since so many other tasks depend on their completion
Try a different way to pick a priority list to reduce time spent waiting for completion of critical tasks.

Before we can start the critical time algorithm, we need to talk about critical times and critical paths.

critical path for X- path from X to END with the *longest* total processing time

critical time for X- total processing time for the critical path from X to END

When finding critical times, work from END to START.



Start at END, look at F and G.

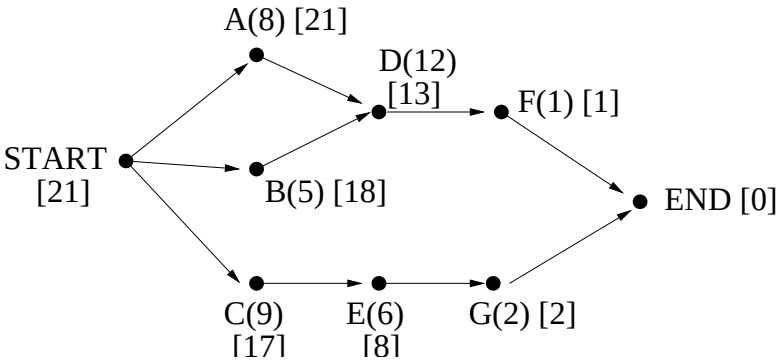
Critical time for F is 1, G is 2.

Write critical numbers in brackets.

Before F, we have D, so critical time is processing time for D + processing time for F, or 12+1=13.

Before G, we have, E, with critical time 8.

Continuing thus, we arrive at the following critical times:



What is the critical path for vertex B? For vertex C? What about the critical path for the entire project? Remember to look for the longest processing time.

Critical Path: START → A → D → F → END.

Critical Time: 21

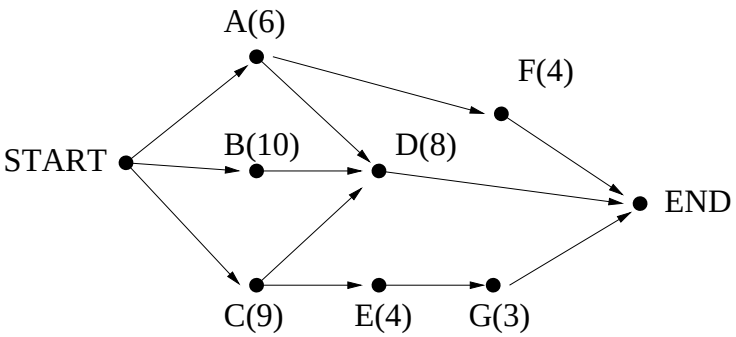
<u>Backflow Algorithm</u> Finds critical times and paths

1. **Start at END vertex, critical time is zero**
2. **Move back to vertices incident to END, critical time is zero plus processing time**
3. **Move backward to each vertex incident to later vertices, critical time is the *longest* critical time plus processing time for that vertex.**
4. **Repeat until START vertex is reached.**

Critical time for the entire project is the critical time for the start vertex

The critical path is the path following the largest critical times.

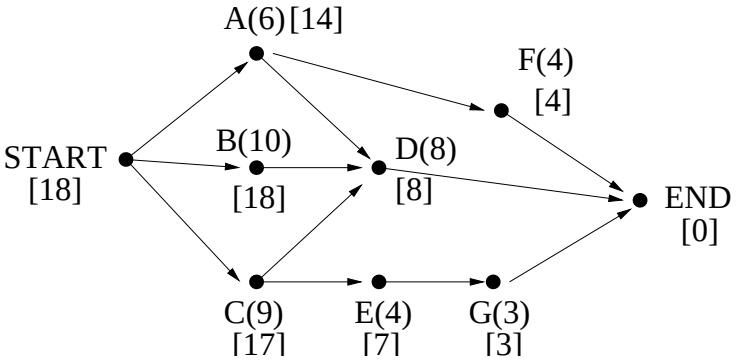
Find the critical times for all the following vertices:



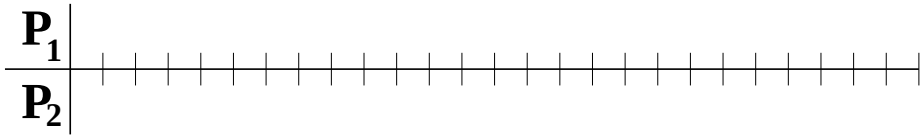
Make a priority list based on the critical times

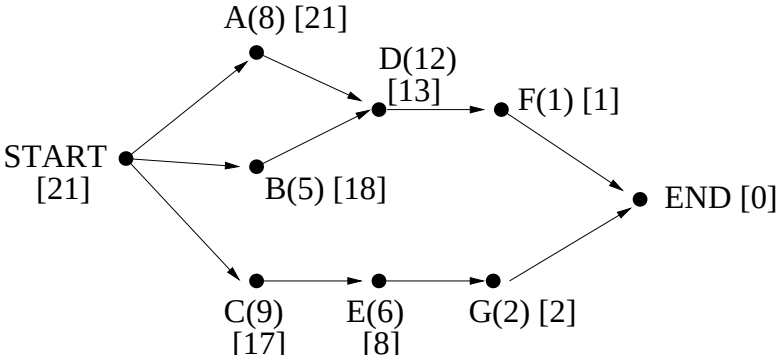
List largest critical times first

For example:

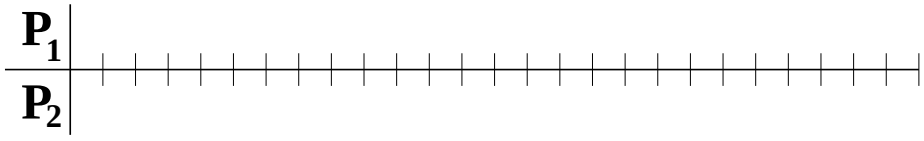


B(10), C(9), A(6), D(8), E(4), F(4), G(3)





Priority List:



The Critical Time Algorithm:

Pro: Good method, most commonly used

Con: Won't always give optimal solution, but better than chosing at random.

Note: There is no effecient scheduling algorithm that will give an optimal solution. The only way to find an optimal solution is to do a brute force type of algorithm, but this is inefficient.

Wedding example: