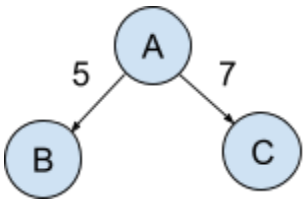
The Challenge

Write a workflow runner that can accept the specification of a workflow in the form of a DAG represented in JSON where letters are assigned to the vertices and numbers are assigned to the edges. One node will be designated as the start vertex.

As the runner goes through the graph it should print the letter of each vertex it visits. Next, for each edge e_i going out of a vertex wait t_i seconds before traveling to the connected vertex where t_i is the number that is tied to edge e_i.

Note: The runner should process edges in parallel so that it starts the “timer” for each edge going out of a vertex at the same time after printing the vertex letter. For example, consider the graph where A is the start vertex:



The runner should start by immediately printing A, then after 5 seconds print B, and then 2 seconds later print C. This graph, represented as JSON, would look something like:

```
{
  "A": {"start": true, "edges": {"B": 5, "C": 7}},
  "B": {"edges": {}},
  "C": {"edges": {}}
}
```

What We Expect

We want to be very clear that there is no “correct” or "trick" answer to this challenge, and we are not asking you to attempt to replicate any existing system. This is a vehicle to understand how you approach an engineering problem, what things you worry about, and where you put your focus. It's a showcase for your technical skills.

At the end of the challenge, you will deliver a set of artifacts that you wish to be judged on.

- Runnable software
    - Code, in a language of your choice, that provides the requested functionality
    - Some sort of driver, main(), etc. that can execute test cases against your code

- One or more test cases that illustrate the requested functionality
- Instructions and/or tooling that describes how to execute (and build, if appropriate) your program
- Any documentation you think is important to our understanding of your approach, design, thoughts, and choices.

How much polish you want to add is up to you, but we anticipate the challenge taking a matter of hours, rather than days or weeks. We only require that you meet the basic functionality expressed in the challenge, but if you want to show off some particular area of skill for extra credit, then that's totally fine, too. We understand this is only a coding challenge and do not expect comprehensive, production grade systems.

Let’s agree on a completion date that works for you in this thread, by which time you should deliver your artifacts. Once that occurs, we will schedule your full interview to discuss it in depth and to ask about other aspects of your technical background.

Things to Avoid

- Please understand we cannot install arbitrary frameworks needed to run your code. We have security and legal processes around what software we install and execute on our machines, so please simplify your solution enough that it does not require special hypervisors, package management systems, or runtime frameworks. Remember, this is a coding challenge, not an application development challenge.
- Similarly, please understand that we can't connect to, or execute, arbitrary systems outside of BigHat, such as solutions running in AWS or Microsoft Azure.

Tips

- Feel free to contact us - we expect engineers will need to ask questions about requirements, priorities, and resources.
- We prefer a narrower submission that shows craft and nuance to a broader submission that only superficially touches on as many areas as possible.
- Taste, quality, and creativity matter. This should represent your best work.