CS3000: Algorithms & Data — Spring 2019 — Paul Hand

Homework 6
Due Wednesday 3/20/2019 at 2:50pm via Gradescope

Name:
Collaborators:

- Make sure to put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- This assignment is due Wednesday 3/20/2019 at 2:50pm via Gradescope. No late assignments will be accepted. Make sure to submit something before the deadline.
- Solutions must be typeset in LaTeX. If you need to draw any diagrams, you may draw them by hand as long as they are embedded in the PDF. I recommend using the source file for this assignment to get started.
- I encourage you to work with your classmates on the homework problems. *If you do collaborate, you must write all solutions by yourself, in your own words.* Do not submit anything you cannot explain. Please list all your collaborators in your solution for each problem by filling in the `yourcollaborators` command.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class is strictly forbidden.

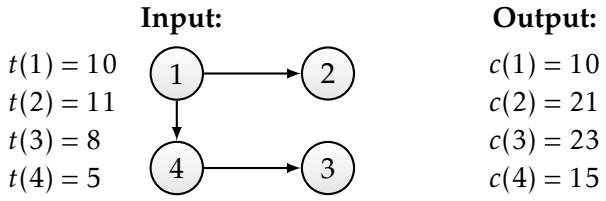
Problem 1. *Stagecraft*

You're in charge of assembling the stage for this year's *AlgoRythms Music and Informatics Jamboree.* Assembling the stage in time will require careful planning. You are given:

- A set V of n small tasks that are required to complete the stage.
- A set E of pairs of tasks in V. A pair (u, v) is in E if task u must be completed before task v is started. There are no cycles in E.
- For each task $u \in V$, the amount of time $t(u)$ required for the task.

You have a very large team, so you can work on any number of tasks in parallel, but you cannot start a task u until all of the prerequisite tasks v have been completed.

Design an algorithm that takes as input the graph $G = (V, E)$ (represented as an adjacency list) and the time for each task, and outputs a list consisting of the earliest possible time that each task can be completed. An example of a correct input-output is:



(a) Describe in 1-3 English sentences how you will determine the earliest possible completion times for the tasks.

Solution:

(b) Describe your algorithm in pseudocode. You may make use of any algorithm we've seen in class without describing how it works, but you should clearly state what you are assuming about the algorithm.

Solution:

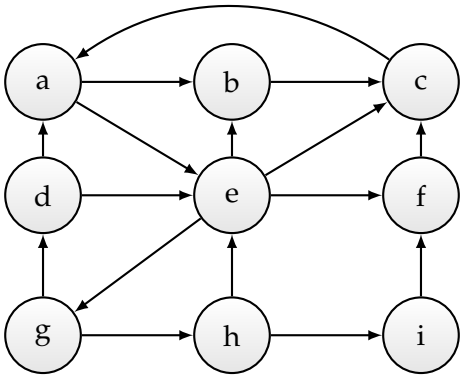
(c) Justify that your algorithm outputs the earliest possible completion time for each project. Your justification can take any form you like as long as the argument is clear and logical.

Solution:

(d) Analyze the running time of your algorithm. If your algorithm uses some algorithm from class as a subroutine, don't forget to include this running time in your analysis.

Solution:

Problem 2. *DFS and Topological Ordering*



Consider running depth-first search on this graph starting from node *a*. When there are multiple choices for the next node to visit, go in alphabetical order.

(a) Label every edge as either tree, forward, backward, or cross.

Solution:

(b) Give the post-order numbers of all vertices

Solution:

a	b	c	d	e	f	g	h	i

(c) Is this graph a DAG? If so, give a topological ordering.

Solution:

Problem 3. *Cleaning up the Streets*

You have been hired to clean up the streets of Boston using your street sweeper. In order to do the job for the lowest cost, you want to devise a way to sweep each street exactly one time in each direction. That is, for each street, you go up the street once and down the street once to sweep each side of the road. Assume all roads are two-way streets.

Specifically, your city is modeled as a graph $G = (V, E)$, where the vertices represent the intersections and edges represent the roads connecting intersections. Design an algorithm based on depth-first search that takes the graph $G = (V, E)$ and finds a street-sweeping route that goes along each edge exactly once in each direction. Your algorithm should run in $O(V + E)$ time.

(a) Explain in a few English sentences how your algorithm works.

Solution:

(b) Describe your algorithm in pseudocode.

Solution:

(c) Justify that your algorithm finds a correct street-sweeping route. Your justification may take any form you like, as long as it is clear and convincing.

Solution:

(d) Analyze the running time of your algorithm.