Fundamentals of Computer Algorithms
Homework 5 Additional Problems

Prof. Matthew Moore

Due: 2018-09-25

1. (i) Implement the function `topological_sort` in the attached python file. Your algorithm should be of *linear* complexity in the vertices and edges of the graph. Submit only your function in your printed homework.
   (ii) Use `randgraph_DAG` to generate a random graph with 10 vertices. (This can take a *long* time.) Draw this graph.
   (iii) Run your `topological_sort` function on your graph from the previous part. Draw the resulting topologically sorted graph.
2. (i) Implement the function `is_DAG` in the attached python file. Your algorithm should be of linear complexity in the vertices and edges of the graph. Submit only your function in your printed homework.
   (ii) Explain why your algorithm works.
3. (i) Implement the function `findHamiltonian_DAG` in the attached python file. Your algorithm should be of *linear* complexity in the vertices and edges of the graph. Submit only your function in your printed homework.
   (ii) Use `randgraph_DAGwithHam` to generate a random graph with 10 vertices. (This can take a *very* long time.) Draw this graph.
   (iii) Run your `findHamiltonian_DAG` function on the graph from the previous part and indicate the Hamiltonian path in your drawing.
   (iv) Run your `topological_sort` function on your graph from the previous part. Draw the resulting topologically sorted graph.
4. In the interval scheduling problem we always select the interval which finishes first and which is compatible with all previous selections. Suppose instead that we select the interval which *starts last* and which is compatible with all previous selections.
   (i) Write a pseudocode algorithm with complexity $\mathcal{O}(n \log n)$ implementing this strategy.
   (ii) Does this algorithm always return an optimal solution? If it doesn't, then provide a counterexample. If it does, then prove it.
5. For each of the parts below, show by producing a counterexample that the method of selecting intervals does not produce an optimal solution to the interval scheduling problem.
   (i) Select intervals with earliest starting time and compatible with all previous selections.
   (ii) Select intervals with the shortest duration and compatible with all previous selections.