CS2223: Algorithms
D-Term, 2015

Assignment 4

Teams: To be done individually
Due date: 04/24/2015 (1:50 PM)
Submission: Electronic submission only

General Instructions

- ***Python Code vs. Pseudocode:*** Each question will explicitly state whether the deliverable is pseudocode or a Python program that the TA will run to give you a grade.
- ***Programming Language:*** If a question asks you to write a program, then use Python language.
    - ***Submissions:*** The submission of Homework 4 must be done electronically through the blackboard site for CS2223 available from myWPI. Login to myWPI, go to CS2223 under "My Courses", then go to "Assignments", and submit your homework under "HW4". All programs plus your report (.doc or .pdf) should be zipped into a single file and that is the file to submit.

Question 1 (Greedy Algorithms) [20 Points]

Suppose that your USB flash drive has a capacity of G gygabytes. Assume that you have the following n files that you want to copy onto the flash drive: F_1, F_2, F_3, ..., F_n. The i^{th} file F_i takes g_i gygabytes. The problem is that your USB flash drive's capacity, G, is less than the sum of all the g_i's. That is, $G < \Sigma^n_{i=1} g_i$.

1. [10 Points] Assume that you want to maximize the number of files that you copy onto your USB flash drive. Prove decisively (using induction or a proof by contradiction) if the following statement is true; or give a counterexample, if the statement is false: A greedy algorithm that selects files to copy in increasing size order (from smallest to largest) will produce an optimal solution.

2. [10 Points] Assume that you want to maximize the space utilization of your USB flash drive (that is, you want to use as much of your G gygabytes as possible). Prove decisively (using induction or a proof by contradiction) if the following statement is true; or give a counterexample, if the statement is false: A greedy algorithm that selects files to copy in decreasing size order (from largest to smallest) will produce an optimal solution.

Question 2 (Greedy Algorithms) [40 Points]

Consider the following scheduling with deadlines problem. We have a set of *n* jobs, each of which takes exactly 1 minute to execute. At any time $T = 1, 2, \ldots$ (in minutes) we can execute exactly one job. Each job *i* earns us a profit $g_i > 0$ if and only if it is executed no later than time d_i.

For example, with $n = 4$ and following values:

i	1	2	3	4
g_i	50	10	15	30
d_i	2	1	2	1

the schedules (each schedule is a sequence of jobs) to consider and the corresponding profits are:

Sequence	Profit	
1	50	
2	10	
3	15	
4	30	
1, 3	65	
2, 1	60	
2, 3	25	
3, 1	65	
4, 1	80	← optimal
4, 3	45	

The sequence 3, 2 for instance is not considered because job 2 would be executed at time $t = 2$, after its deadline $d_2 = 1$. To maximize our profit in this example, we should execute the schedule 4,1 above.

A set of jobs is called *feasible* if there exists at least one ordering of the jobs in the set that allows all the jobs in the set to be executed no later than their respective deadlines. A schedule (= sequence of jobs) is called *feasible*, if the set of jobs in the schedule is feasible.

Optimization problem: We want to find a feasible schedule that maximizes our profit.

Greedy strategy to solve this problem: Construct the schedule step by step, adding at each step the job with the highest value of g_i among those not yet considered, provided that the set of chosen jobs remains feasible.

This greedy strategy is guaranteed to construct an optimal schedule (you don't need to prove this, just take our word for it).

(1) [5 points] Follow this greedy strategy step by step with the example shown above to show that the output schedule will be 4, 1.

(2) [20 points] Write a greedy algorithm (in pseudo-code) to solve this problem using the greedy strategy provided above. Your algorithm must output an optimal feasible schedule (that is, a sequence of jobs that is feasible and maximizes our profit). Describe clearly the data structures that you're using in your pseudo-code. Note that a crucial part of your algorithm is to figure out how to determine if adding a job to the current feasible schedule will keep the schedule feasible or not.

(3) [5 points] Illustrate how your algorithm works by following your algorithm by hand step by step on the input below with $n = 4$:

i	1	2	3	4	5	6
g_i	20	15	10	7	5	3
d_i	3	1	1	3	1	3

(4) [10 Points] Analyze the time complexity of your algorithm in the worst case.

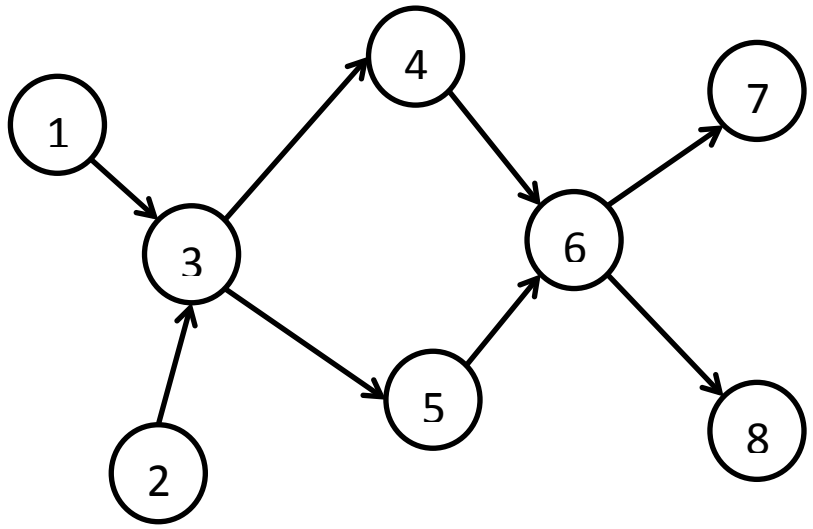
Question 3 (Greedy Algorithms) [10 Points]

Consider the *Interval Partitioning* problem described and solved on slides 15-22 of Chapter 4 of Jon Kleinberg's and Éva Tardos' Algorithm Design textbook, available at http://www.cs.princeton.edu/~wayne/kleinberg-tardos/

Provide a complete proof that the greedy algorithm presented on slide 19 runs in $O(n \log n)$. For this you need to fill in the details in the time complexity analysis on slide 20.

Question 4 (Depth-First Search) [15 Points]

Run the DFS-based algorithm for topological sort on the following graph; whenever there is a choice of vertices, pick the one with the smallest number. Start your DFS from Node (1).



(1) [10 points] Show the discovery and finishing times for each vertex.

Node	1	2	3	4	5	6	7	8
Start								
Finish								

(2) [5 points] Give the topological ordering found by the algorithm. (Do not simply write down a topological sort by inspection: that will earn no points)

Question 5 (Breadth-First Search) [15 Points]

(1) [10 points] Write a pseudocode for Breath-First-Search where we represent the input graph by an adjacency matrix (instead of adjacency list).

(2) [5 points] What is the running time of the BFS algorithm if we use adjacency matrix to represent the input graph?

Bonus Question (Number of Islands) [10 Bonus Points]

Please go to the following webpage, and read the details:

https://leetcode.com/problems/number-of-islands/

Note: in the drop menu for programming languages, select "Python". Start working on your code in the webpage, and feel free to try out submissions. Feedbacks on the correctness of your code will be given by the webpage.

Deliverables of Bonus Question

Write a Pyhon program to implement the algorithm described above. Your program should be able to pass **all** test cases in https://leetcode.com/problems/number-of-islands/ by trying to submit your code on the webpage. You can try submitting as many times as you want on leetcode webpage.

If your program can pass all the tests on leetcode, please use this code as your solution for this bonus question and submit it together with other questions in HW4.