CS2223: Algorithms
D-Term, 2015

Assignment 5

Teams: To be done individually
Due date: 05/01/2015 (1:50 PM)
Note: no late submission of HW5 will be accepted; we will talk about the solution of HW5 during the class of May 1st
Submission: Electronic submission only

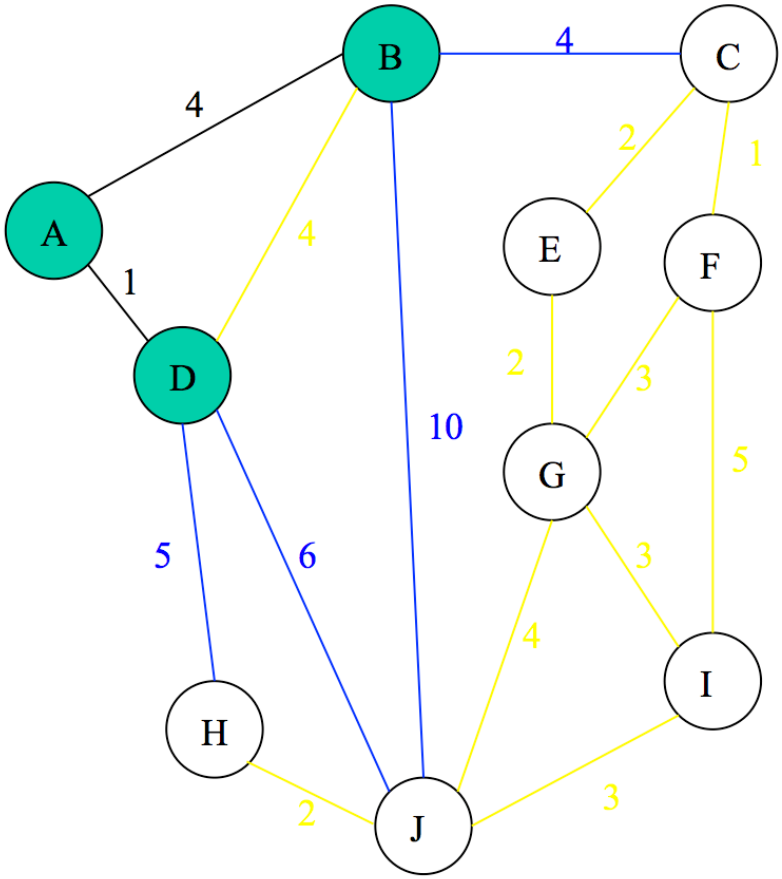
General Instructions

- ***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 5 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 "HW5". All programs plus your report (.doc or .pdf) should be zipped into a single file and that is the file to submit.

Question 1 (Minimum Spanning Tree) [20 Points]

(1) [10 points] write a pseudocode for Prim's algorithm, where we maintain the collection of vertices that are not in the tree, naively, using an unsorted array (instead of the priority queue).

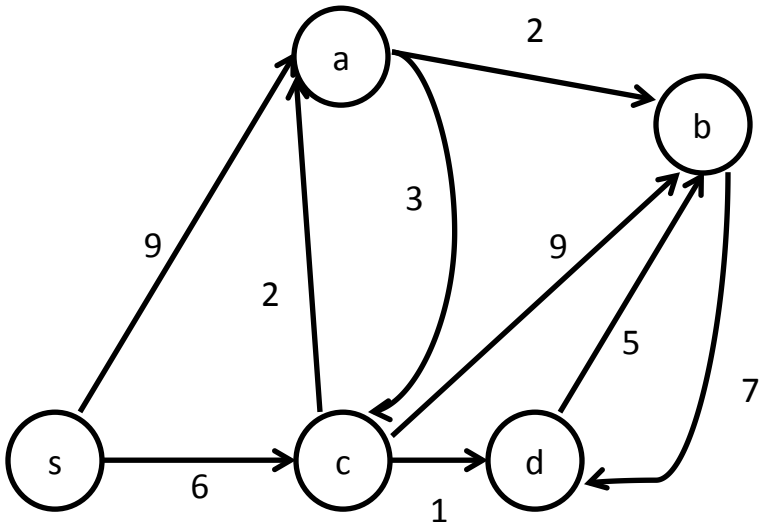
Clarification: Based upon the slides in class, here is an example of the collection of vertices that are not in the tree. The white nodes are not in the tree.



(2) [10 points] give a Θ analysis of the worst-case running time of the Prim's algorithms, where we maintain the collection of vertices that are not in the tree, naively, using an unsorted array (instead of the priority queue). We assume **n** represents the number of vertices and **m** for the number of edges.

Question 2 (Shortest Path) [20 Points]

Show the process of **Dijkstra's algorithm** on the graph below, where the start vertex is **s**, by showing the **d[v]** values for each vertex **v** and the growing set **S** of "settled" vertices, those that have been added to the shortest-path tree.



	S	s	a	b	c	d
Step 0	∅	0	∞	∞	∞	∞
Step 1	{ *s* }					
Step 2						
Step 3						
Step 4						
Step 5						

Question 3 (Dynamic Programming) [45 Points]

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

Assume that you want to maximize the space utilization of your USB flash drive (that is, you want to use as many of your G gigabytes as possible). We proved in Homework 4 that a greedy algorithm that selects files to download in decreasing size order (from largest to smallest) will NOT produce an optimal solution.

In this homework, you will provide an optimal dynamic-programming solution to this problem following the steps below. *Note that the output of the algorithm should merely be the maximum space utilization (a single numerical integer value, in gigabytes). The subset of files for which this maximum is attained, does not need to be returned*. Here is the input-output specification:

> **Maximum space utilization** (n, g[1...n], G)
> **Inputs:** n: positive integer; array g[1...n] of positive integer file sizes; and positive integer space capacity G.
> **Output:** Maximum space utilization $\sum_{i \in S} g_i$ that can be attained over a subset $S \subseteq \{1, \ldots, n\}$ of files without exceeding the space capacity G.

1. Dynamic Programming Solution.
   a. [20 points] Design a dynamic programming solution to the space utilization problem, using the subproblems suggested by the hint that follows.

      Hint: Consider whether or not to add the i-th file to a space of capacity j, assuming that the first i-1 files and smaller space capacities j' < j have already been considered. To this end, define a 2-dimensional (n+1) * (G+1) matrix (array) OPT[0..n][0..G], where OPT[i,j] = maximum space utilization over a subset of the first i files $F_1 \ldots F_i$, assuming a maximum space capacity of j gigabytes.

      Find a recurrence relation that expresses the solution of OPT[i,j] in terms of the solutions of slightly "smaller" problems (i.e., other cells in the matrix OPT whose values can be calculated before). Include suitable boundary conditions for the recurrence relation. At each stage argue decisively that your solution is correct.

   b. [5 points] Based on the preceding part of this question, determine in what order you will calculate the values of the cells in OPT. Explain.
2. [10 Points] Algorithm. Write a dynamic programming algorithm (in detailed pseudo-code) implementing the solution you designed above.
3. [10 Points] Time Complexity. Determine the asymptotic running time of your dynamic programming algorithm above, as a function of the input sizes n and G. Give the simplest possible Big-Theta expression for the running time.

Question 4 (Dynamic Programming) [15 Points]

Given two positive integer numbers n and m, find the number of ways in which n different elements can be partitioned into m different (non-empty) groups.

> For example, the number of ways to partition 4 elements (call them a, b, c, d) into 3 different non-empty groups is 6:

	Group 1	*Group 2*	*Group 3*
1.	a	b	c, d
2.	a, b	c	d
3.	a, c	b	d
4.	a, d	b	c
5.	b, c	a	d
6.	b, d	a	c

Let OPT[n,m] = number of ways in which n different elements can be partition into m different (non-empty) groups. For example OPT[4,3] = 6.

Find a recurrence relation that expresses the solution of OPT[n,m] in terms of the solutions of slightly "smaller" problems (i.e., other cells in the matrix OPT whose values can be calculated before). Include suitable boundary conditions for the recurrence relation. At each stage argue decisively that your solution is correct.

Bonus Question 1 (Best Time to Buy and Sell Stock 1) [10 Bonus Points]

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

https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iii/

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 Python program to implement the algorithm described above. Your program should be able to pass **all** test cases in https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iii/ 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 HW5.

Bonus Question 2 (Best Time to Buy and Sell Stock 2) [10 Bonus Points]

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

https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iv/

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 Python program to implement the algorithm described above. Your program should be able to pass **all** test cases in https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iv/ 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 HW5.

Bonus Question 3 (Dungeon Game) [10 Bonus Points]

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

https://leetcode.com/problems/dungeon-game/

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/dungeon-game/ 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 HW5.