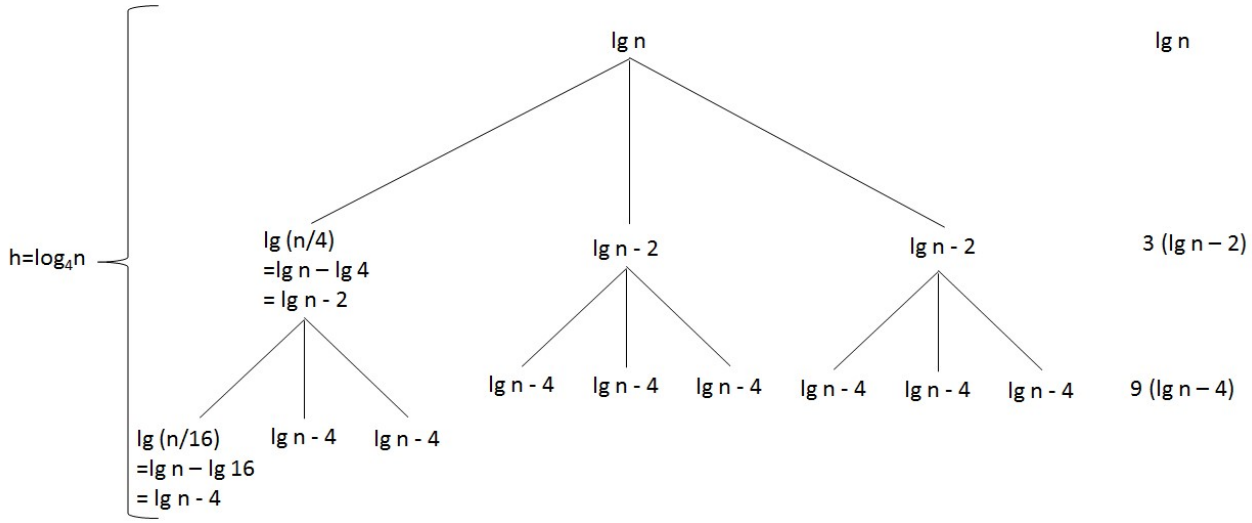
Faculty of Science

Course:	CSCI 3070U: Analysis and Design of Algorithms
Term:	Fall 2014
Course component:	Final Exam Solutions

Question 1 – Complexity (34 marks total)

1. (10 marks) Use the recursion tree method to find an estimate for `T(n) = 3T(n/4) + lg n`, written as a summation of terms representing levels in the recursion tree. Simplify the expression as much as possible in your answer.



$$T(n) = \sum_{i=0}^{log_4 n} 3^i(\lg n - 2^i)$$

2. (10 marks) Find a recurrence, T(n), for the running time of the following pseudo-code, and solve it using any method, showing the result in theta notation (as well as the original T(n)).

```
GRAB-VALS(list)
1.  if list.length == 1 then                                  O(1)
2.     return list                                            O(1)
3.  mid = list.length / 2                                     O(1)
4.  firstHalf = list[0..mid]                                  O(n)
5.  secondHalf = list[mid+1..list.length-1]                   O(n)
6.  firstGrab = GRAB-VALS(firstHalf)                          T(n/2)
7.  secondGrab = GRAB-VALS(secondHalf)                        T(n/2)
8.  grab = firstGrab + secondGrab # concatenate               O(1)
9.  for i = 1 to firstGrab.length do                          O(n)
12.     grab = grab + [secondGrab[i] * firstGrab[i])                O(1)
14. end for
15. return grab                                               O(1)
```

$T(n) = 2T(n/2) + \theta(n) = \theta(n \lg n)$

3. (10 marks) Using substitution, prove that $T(n) = \begin{cases} 2, if\ n = 2 \\ 2T\left(\frac{n}{2}\right) + \theta(n) \end{cases} = \theta(n \lg n)$.

Prove: T(n) = 2T(n/2) + θ(n) = θ(n lg n)

Base case: For n = 2, T(n) = kn lg n = k2 lg 2 = 2k (true, for k = 1)

Assume: For all 0 <= k < n, T(k) = θ(k lg k)

Proof:

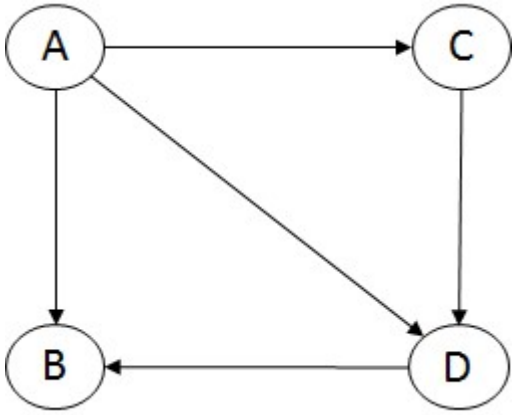
```
T(n)  = 2T(n/2) + cn
      = 2(k (n/2) lg (n/2)) + cn
      = kn (lg (n/2)) + cn
      = kn (lg n – lg 2) + cn
      = kn (lg n – 1) + cn
      = kn lg n – kn + cn
      = kn lg n + (c – k)n
      = θ(n lg n)    (for c – k >= 0, constant)
```

4. (4 marks) Order the following expressions representing running times in terms of their asymptotic order by writing the numbers 1, 2, 3, 4, 5, 6, 7, and 8 in the box beside them (where 1 represents the fastest, and 8 represents the slowest).

$18 + 7n - 4n^2 + 8n^3$	7	Cubic (8n^3)
$n \log n$	3	N log n
$2^{\log_2 2n}$	1	Linear (2n)
$n^2 \log n$	6	N^2 log n
$n \log (n / \log n)$	2	N log n – n log log n
$3n!$	8	Exponential (n!)
$4 \log n^n$	5	4 log n * log n
$n \log n^2$	4	2n log n

Question 2 – Basic Graph Theory (10 marks total)

1. (6 marks) Does the following diagram represent a DAG? Explain in one sentence why or why not.



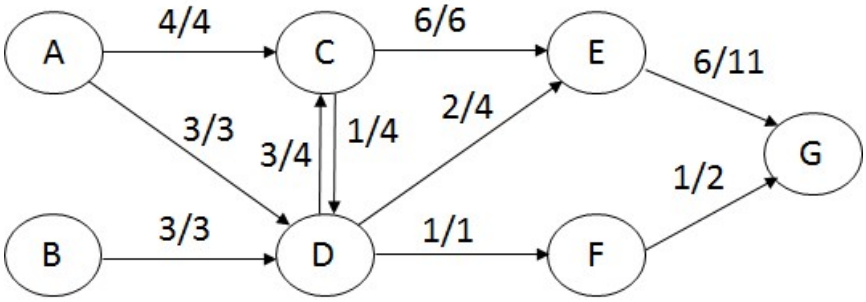
Yes, this graph is a DAG since it uses directed edges and does not contain any cycles.

2. (4 marks) Assuming G is a dense graph, what is the running time complexity of BFS(G) with respect to only $|V|$? (only the answer, in theta notation, is required)

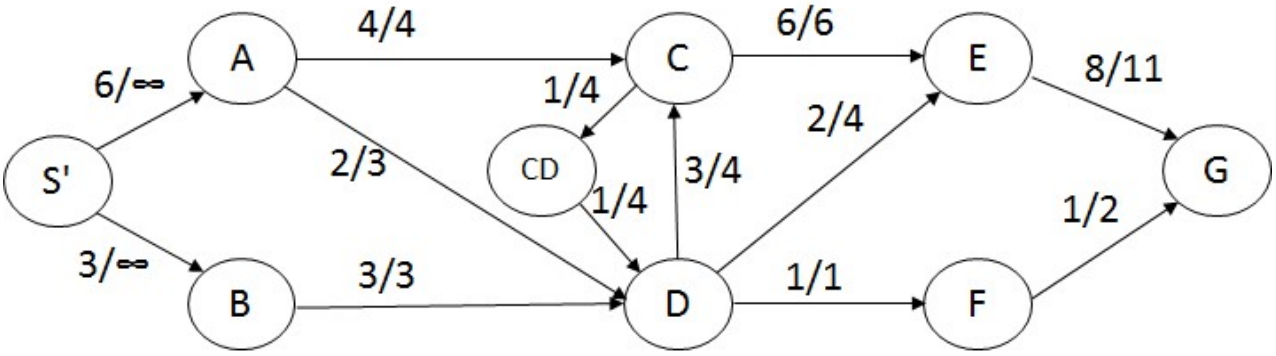
$\Theta(|V|^2)$

Question 3 – Max Flow (25 marks total)

1. (10 marks) Find and list the problems with the following flow network, and re-draw the flow network after these problems have been repaired according to the instructor-recommended method.



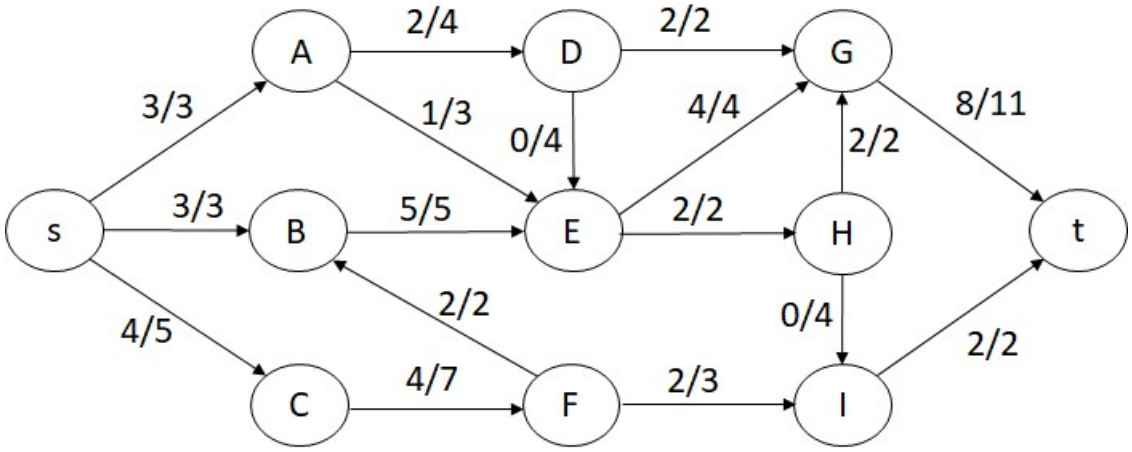
Solution:



1. Fix multi-source
2. Fix anti-parallel edge (C,D)
3. Fix flow conservation of D
4. Fix flow conservation of E

2. (15 marks) Find the max flow of the following network, showing an ordered list of augmenting paths, and each path's critical edges and bottleneck capacity as you figure out the answer.

Solution:



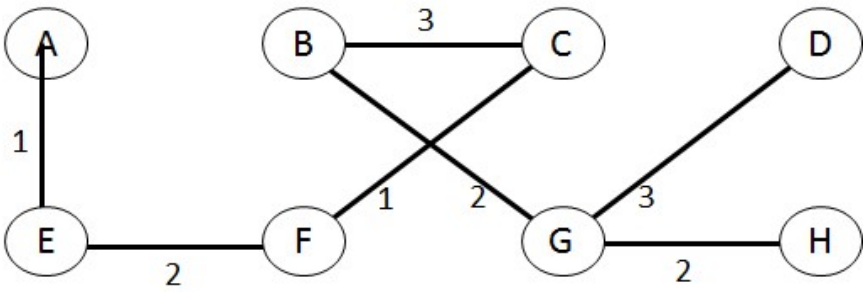
PATH	CRITICAL EDGE(S)	BOTTLENECK CAPACITY
s -> A -> E -> G -> t	(s,A) (A,E)	3
s -> B -> E -> H -> G -> t	(E,H) (H,G)	2
s -> B -> E -> G -> t	(s,B) (E,G)	1
s -> C -> F -> I -> t	(I,t)	2
s -> C -> F -> B -> E -> A -> D -> G -> t	(F,B) (B,E) (D,G)	2

Question 4 – Minimum Spanning Tree (15 marks)

Find the minimum spanning tree of the following graph, and draw it as a separate diagram. Show the list of edges in the order they were added (as a list of edges) for both Prim's algorithm (using vertex B as the starting point) and Kruskal's algorithm (thus, two lists of edges will be included), for comparison.

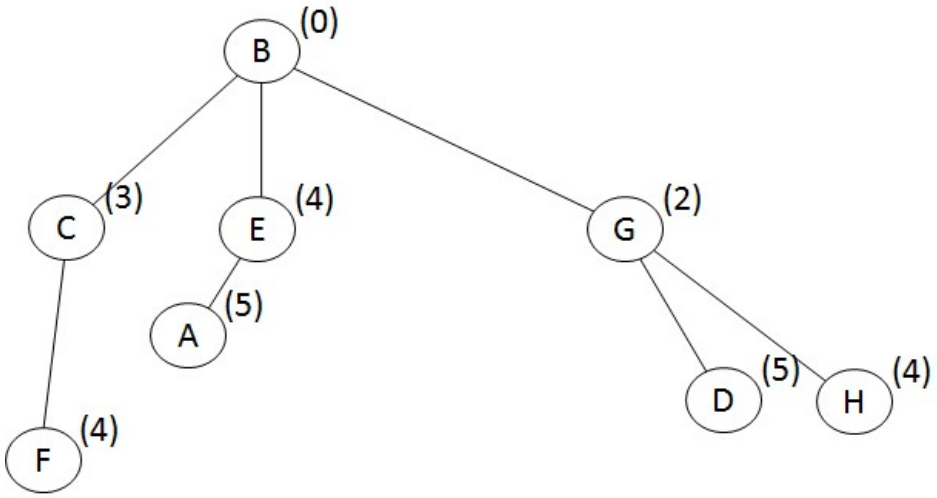
Note: Use the diagram of vertices included below, by augmenting the diagram with the edges that you have added.

ALGORITHM	**EDGES**
PRIM'S	(b,g) 2 (g,h) 2 (b,c) 3 (c,f) 1 (e,f) 2 (a,e) 1 (d,g) 3
KRUSKAL'S	(c,f) 1 (a,e) 1 (e,f) 2 (b,g) 2 (g,h) 2 (b,c) 3 (d,g) 3



Question 5 – Shortest Path (15 marks)

Using Dijkstra's algorithm, find the shortest path from B to all other vertices in the following graph. Show the results in a table with three columns: the vertex, the distance to each vertex, and a path to that vertex.



VERTEX	DISTANCE	PATH
A	5	B,E,A
B	0	B
C	3	B,C
D	5	B,G,D
E	4	B,E
F	4	B,C,F
G	2	B,G
H	4	B,G,H

Question 6 – Algorithm Strategies (45 marks total)

1. (5 marks) For each of the following algorithms, identify the algorithm strategy used (brute force, divide and conquer, greedy, dynamic programming, or none if none of these fits):

    a. Prim's Algorithm

    Divide and conquer (also acceptable: none)

    b. Dijkstra's Algorithm

    Greedy

    c. HeapSort

    None

    d. Kruskal's Algorithm

    Greedy

    e. MergeSort

    Divide and conquer

2. (20 marks) Using the greedy strategy, design an algorithm to solve the following problem, and write that algorithm in pseudo-code. Write, beside the pseudo-code, the running time complexity T(n), in terms of n (the number of integers in the list) in theta notation. For this question, you do not need to show your work, only the pseudo-code and the complexity.

   Write a function that takes a list of integers, each of which can be positive, zero, or negative, (called findMaxEvenSumSubset) that finds the subset of those integers such that the sum of those numbers is both an even number and maximal.

```
# This is (loosely) a greedy strategy
def findMaxEvenSumSubset(numbers):
    numbers.sort()
    numbers = numbers[::-1]

    subset = []
    subsetSum = 0
    for num in numbers:
        if num <= 0:
            # no more useful numbers are in the list

            # if the sum is not even, remove the smallest odd number
            if (subsetSum % 2) != 0:
                for index in range(len(subset) - 1, -1, -1):
                    if (subset[index] % 2) == 1:
                        subset.remove(subset[index])
                        break

            break
        else:
            # add the current (non-negative) value to our subset
            subset.append(num)
            subsetSum += num

    return subset
```

3. (20 marks) In this question, you are to design a dynamic programming algorithm, FIND-LONGEST-PATH, to find the longest path (not shortest path) from any vertex to any other vertex in a directed graph. The length of a path is strictly determined by the number of edges, as edges will not have any weights. Write out your algorithm in C++, Java, or Python.

   ***Note**: You can assume that the directed graph does not contain any cycles for this question.*

```
def findLongestPath(vertices, edges):

    allPaths= []

    # initialize the paths with edges (i.e. length 1 paths)
    for (u,v) in edges:
        allPaths.append([(u,v)])

    for (u,v) in edges:
        # see if this edge can extend any path so far
        for path in allPaths:
            # check to see if the edge connects to the start of the path
            (x,y) = path[0]
            if v == x:
                newPath = [(u,v)] + path
                allPaths.append(newPath)

            # check to see if the edge connects to the end of the path
            (x,y) = path[-1]
            if u == y:
                newPath = path + [(u,v)]
                allPaths.append(newPath)

    # figure out which path is the longest
    maxIndex = 0
    maxLength = len(allPaths[0])

    for index in range(1, len(allPaths)):
        if len(allPaths[index]) > maxLength:
            maxLength = len(allPaths[index])
            maxIndex = index

    return allPaths[maxIndex]
```

Question 7 – Theory of Computation (21 marks total)

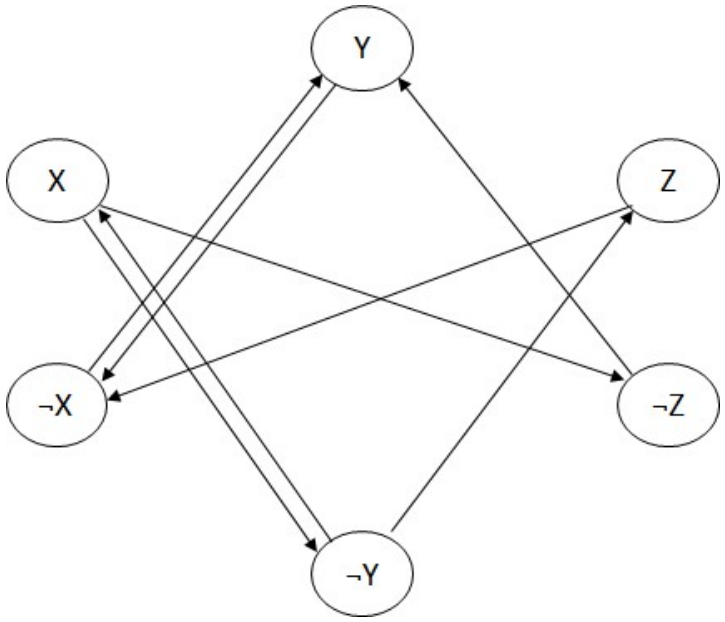
1. (6 marks) Which algorithm category (P, NP, NP-complete, NP-hard, intractable) best describes the following problems?

Problem	**Category**
Finding a path through an arbitrary graph such that every vertex is visited exactly once.	NP-hard
To find a forgotten password of a known length, k, for which you know the hashed version of that password involves guessing every possible password, and checking the hash of that guess against the password's hash.	Intractable
Given an arbitrary program, determine the average running time of the program over 100 executions of randomized input(s).	Non-computable

2. (15 marks) Find values that satisfy the following expression, using the method described in the lectures: `(¬X ∨ ¬Z) ∧ (Y ∨ Z) ∧ (X ∨ Y) ∧ (¬X ∨ ¬Y)`

   Note: Write all possible values for each variable, if there is more than one.

Variable	**Value(s)**
X	False
Y	True
Z	True, False



Question 8 – Minimum Edit Distance (15 marks)

What is the Minimum Edit Distance between the string 'abc' and the string 'ybcab'? Use Levenshtein's method, and show the complete table that results.

Note: You are not required to draw the arrows, as was shown during the lectures. If you draw then anyway, please ensure that they do not obscure the values, which are what will be marked.

		y	b	c	a	b
	0	1	2	3	4	5
a	1	1	2	3	3	4
b	2	2	1	2	3	3
c	3	3	2	1	2	3