Algorithms

Spring 2017

Short Questions

(Answer all the questions)

1. a. Define height balanced binary tree.
   b. Write a pseudo code to determine whether a tree is height balanced?
   c. Obtain the tight bound of your algorithm.

2. Suppose you have keys of N objects stored in an array in the ascending order of key values. Also assume that there is duplicate entry in the array.
   a. Describe an efficient algorithm with the pseudo code that helps you to search for the object given the key (the algorithm must return null value if the key is not in the array).
   b. Obtain the tight upper bound for your algorithm.

3. a. Define strongly connected components as applied to directed graphs.
   b. Some potential application of strongly connected components.
   c. provide a pseudo code for obtaining strongly connected components of a directed graph. (Hint: use depth first search on appropriately transformed graphs.)

4. Find the tight for the recurrence relation below without using the master theorem (show all the steps):

$$T(n) = T(n/2) + n$$

5. a. What are the properties of min heap and max heaps.
   b. What is the preferred data structure of implementing binary heap, also justify your answer.
   c. What is the time complexity of merging two different min heaps each of size n and m.

6. Briefly describe minimum spanning tree. Sketch an algorithm to obtain a minimum spanning tree.

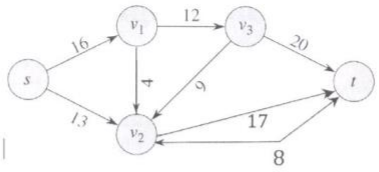
Long Questions

(Answer any 3 of 4 questions)

1. (Problem is related to dynamic programing formulation and optimal sub structure)
   a. Explain what do you understand by "principle of optimality."
   b. Consider the problem of finding the longest common subsequences (LCS) in a pair of sequences, namely, $X(x_1, x_2, ..., x_m)$ and $Y(y_1, y_2, ..., y_n)$.
      b1. A brute force approach is to generate all possible subsequences of X and see whether it is also a subsequence of Y. What is the number of possible subsequence of X?
      b2. Obtain the optimal substructure of an LCS.
   c. Consider a chain matrix multiplication problem, $m_1 \times m_2 \times m_3 \times ... \times m_n$. It is associative, but not commutative.
      c1. A brute force approach is to generate all possible ways of parenthesizing the matrix chain and compute the total number of operations. What is the number of possible grouping of the matrix chain?
      c2. Obtain the structure of the optimal parenthesization and then the recursive definition of the minimum cost of parenthesizing the product $m_i, m_{i+1}, ..., m_j$.

2.
   a. Compare and contrast P, NP, NP-complete and NP-hard.
   b. Based on current conjecture, draw a venn diagram to show the relationship among these classes of problem.
   c. Suppose there are *n* clauses and *m* propositions in a given 3p-sat problem. How many possible interpretations are there? What is the time complexity of testing the satisfiability of a given interpretation? What is the time and space complexity of testing the satisfiability of the clauses.
   d. 3-p sat problem is NP-complete, but still people have published papers by applying heuristics strategy and showing that they were able to solve it with large number of distinct propositions (say 100) and large number of clauses (say 200). To avoid any bias, they have generated the clauses and the set of propositions randomly. How would you start investigating their results? Can their results be generalized?
   e. Suppose a single NP-complete problem is solved in polynomial algorithm, what can you state about the entire NP-complete class as well as the NP-hard class.

3. The Edmonds-Karp algorithm (*EK*) follows the basic Ford-Fulkerson method with breadth-first search to choose the shortest augmenting path (in terms of the number of edges involved) for computing the maximum flow iteratively from vertex *s* to vertex *t* in a weighted directed graph. Illustrate the maximum flow computation process (including the augmenting path chosen in each iteration and its resulting residual network) via *EK* for the graph depicted below.



4. Given a set of 4 keys, with the following probabilities, determine the cost and the structure of an optimal BST (binary search tree), following the tabular, bottom-up method realized in the procedure of OPTIMAL-BST below to construct and fill $e[1..5, 0..4]$, $w[1..5, 0..4]$, and $root[1..4, 1..4]$.

i	0	1	2	3	4
p_i		0.10	0.08	0.22	0.21
q_i	0.06	0.12	0.07	0.05	0.09

Construct and fill the three tables, and show the optimal BST obtained.

```
OPTIMAL-BST(p, q, n)
 1  let e[1..n+1, 0..n], w[1..n+1, 0..n],
        and root[1..n, 1..n] be new tables
 2  for i = 1 to n + 1
 3      e[i, i − 1] = q_{i−1}
 4      w[i, i − 1] = q_{i−1}
 5  for l = 1 to n
 6      for i = 1 to n − l + 1
 7          j = i + l − 1
 8          e[i, j] = ∞
 9          w[i, j] = w[i, j − 1] + p_j + q_j
10          for r = i to j
11              t = e[i, r − 1] + e[r + 1, j] + w[i, j]
12              if t < e[i, j]
13                  e[i, j] = t
14                  root[i, j] = r
15  return e and root
```