Algorithm and Theory of Computation

Short Questions

Answer all the questions. All short questions, but Q3, carry 2 points each. Question 3 will be evaluated for 4 points.

1. Briefly define upper, lower and tight time bound of an algorithm. How does an average time complexity related to any of these bounds?

2. In terms of run time efficiency, compare and contrast quick sort and merge sort? What is the best and the worst case time complexity of quick sort algorithm. Also state under what condition one may expect these two extreme cases?

3. Find the tight bounds of the following sums.
a. $\sum_{i=1}^{i=n} i^3\ a^i$ where is a constant greater than 1

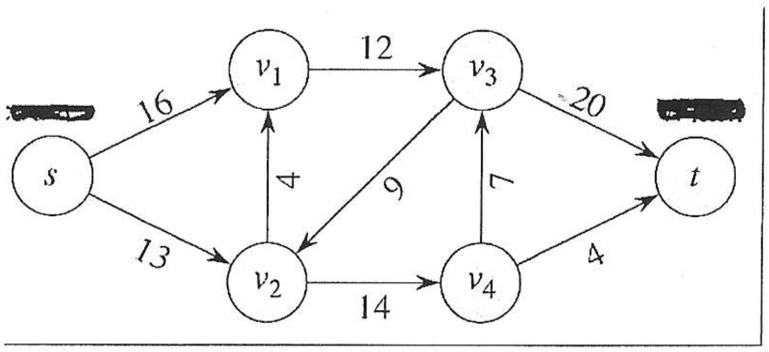
b. $\sum_{i=1}^{i=n} i \log(i^3)$

4. Mark true/false (T/F) against the following statements:
   a. Connecting any pair of nodes in a minimum spanning tree will always forms a cycle.
   b. Building strongly connected component graph of a directed graphs of N nodes takes $O(N^2)$ time.
   c. A graph formed by strongly connected component nodes, a strongly connected component graph (SCC), always is a minimum spanning tree.
   d. SCC graph will be useful in determining articulation node in a graph.

5. Use a binary tree representation to illustrate every operation of MIN HEAPSORT involved when sorting Array $A = \{5, 13, 2, 25, 7\}$ without auxiliary storage.

6. Show your construction of an optimal Huffman code for the set of 7 frequencies: **a**:2 **b**:3 **c**:5 **d**:8 **e**:13 **f**:21 **g**:34.

7. 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.



8. Briefly describe priority queue that maintains the highest key value. What is the preferred data structure for implementing priority queue? If the following keys with values 33, 22,35,40,36 and 41 were added in the given order to an empty priority queue that maintains highest key value at the root, draw the priority queue at the end of all the given set of keys. (show all the key values, and their indexes).

9. Mark true/false against the following statements:

a. A Binary search tree of size N will always find a key at most O(log N) time
b. A breadth first search algorithm can be considered as a special case of heuristic search algorithm.
c. An optimal binary search tree is not necessarily being balanced tree.
d. A dynamic programming approach uses top-down problem solving strategy to solve optimization problem.

Long Questions

Answer any 3 of 4 questions. Each long question will be marked for 20 points.

1. a. Suppose you have to find a solution to a problem that belongs to NP-complete class. Clearly summarize the steps that will help you to find the solution of the problem.

b. John, an undergraduate student recently took data structure course, sates that heuristics algorithms always solve NP-complete problems. He cites simplex methods as an example. If you agree with John justify why or why not.

c. What is the strategy behind a greedy algorithm? Will it always provide an optimal solution? If yes explain why, otherwise say why not.

d. Consider the following pseudo code for Kruskal's algorithm for solving minimum spanning tree (MST).

```
Algorithm MST (G:graph){ # Let N be the number of nodes in graph G
1    Sort the edges in non-decreasing order of cost
2    T := empty graph
3    While T has fewer then N-1 edges do{
4        Let e denotes the next edge of G (in the order of cost)
5        If T ∪{e} does not contain a cycle, then T:= T ∪{e}
```

```
        }
    }
```

Clearly mentioning the data structure you have to employ to reduce the time complexity to access and to maintain the necessary information, show the exact time taken to obtain the MST. Also show the tight bound of the algorithm. (Pay attention in detecting a cycle)

2.A. Explain what do you understand by "principle of optimality"

B. Write down the basic rule that satisfies the principle of optimality and domain related constraints to the following problems.

B1. 0-1 knapsack problem.
B2. Pairwise shortest path problem.
B3. Chain matrix multiplication problem.

C. The optimal solution to 0-1 knapsack problem belongs to NP-class. John says dynamic program formulation optimally solves 0-1 knapsack problem. If you agree with John explain why, otherwise explain why not.

3. 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. Clearly state what is understood propositional satisfiability problem?

d. Suppose there are *m* clauses and *k* 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?

e. Suppose you came across a research paper that highlights a clever heuristic strategy that the authors have used to solve an instance of 3-p sat problem which have 10,000 propositions and 10,000 clauses with 3 propositions. They have generated several instances of the similar compositions of variables and clauses. The authors, in their point of view, have demonstrated the power of their heuristic to solve an instance of NP-complete problem in polynomial time (emperically shown). What would be your insight of their findings.

4. a. An object r is accessed by its key k_r. If all the objects have an equal chanced of being accessed, what data structure will help you to have a better tight bound? (State your assumptions and provide the bound you have obtained for the proposed structure)

b. An optimal binary search tree (OPTIMAL-BST) for a given set of keys with known access probabilities ensures the minimum expected search cost for key accesses, with its pseudo code listed below. Given the set of three keys with their access probabilities of $k_1 = 0.25$, $k_2 = 0.15$, $k_3 = 0.3$, respectively, and four non-existing probabilities of $d_0 = 0.1$, $d_1 = 0.05$, $d_2 = 0.08$, $d_3 = 0.07$, construct optimal BST following dynamic programming with memoization for the given three keys and demonstrate the constructed optimal BST, which contains all three keys (k_1, k_2, k_3) and four non-existing dummies (d_0, d_1, d_2, d_3). (show your work using the three tables for expected costs: $e[i, j]$, for access weights: $w[i, j]$, and for $root[i, j]$, with i in $e[i, j]$ and $w[i, j]$ ranging from 1 to 4, j in $e[i, j]$ and $w[i, j]$ ranging from 0 to 3, and both i and j in $root[i, j]$ ranging from 1 to 3).

```
OPTIMAL-BST$(p, q, n)$
let $e[1..n+1, 0..n]$, $w[1..n+1, 0..n]$,
        and $root[1..n, 1..n]$ be new tables
for $i = 1$ to $n+1$
    $e[i, i-1] = q_{i-1}$
    $w[i, i-1] = q_{i-1}$
for $l = 1$ to $n$
    for $i = 1$ to $n - l + 1$
        $j = i + l - 1$
        $e[i, j] = \infty$
        $w[i, j] = w[i, j-1] + p_j + q_j$
        for $r = i$ to $j$
            $t = e[i, r-1] + e[r+1, j] + w[i, j]$
            if $t < e[i, j]$
                $e[i, j] = t$
                $root[i, j] = r$
return $e$ and $root$
```