대학원 종합시험 – 알고리즘설계및분석 (Algorithm Design and Analysis)

1. Suppose you have a problem of size n that can be divided into b subproblems of size n/b. Assume you are using a divide and conquer algorithm to solve this problem that takes n time units to combine the results from the subproblems.
   a. Write a recurrence relation T(n) for this problem.

   b. Solve your recurrence in (a) above. For the base case, assume that $T(n) = 0$ when $n \leq 1$. Give the exact bound (i.e., theta).

2. Indicate, for each pair of expressions (A, B) in the table below, whether A is O, o, Ω, ω, or ϴ of B. Assume that $k \geq 1$, $\varepsilon > 0$, and $c > 1$ are constants. Your answer should be in the form of the table with "yes" or "no" written in each box.

A	*B*	*O*	*o*	Ω	ω	ϴ
$\lg^k n$	n^{ε}					
n^k	c^n					
$n^{0.5}$	$n^{\sin n}$					
2^n	$2^{n/2}$					
$n^{\lg c}$	$c^{\lg n}$					
$\lg(n!)$	$\lg(n^n)$					

3. Suppose you have an unsorted array of n entries in a file and your machine can only load up to $k << n$ entries in memory to process. Now you want to sort the n entry file using this machine. Give the algorithm to do this task and explain the complexity of the algorithm in terms of disk I/O.

4. Take the following list of functions and arrange them in ascending order of growth rate.
   $f1(n) = n^{7.5}$
   $f2(n) = n+25000$
   $f3(n) = 2^n$
   $f4(n) = 22n$
   $f5(n) = 10^n$
   $f6(n) = 10n^3 * \lg(n)$
   $f7(n) = (\lg n)!$
   $f8(n) = \lg (n!)$
   $f9(n) = \lg^2 n$
   $f10(n) = n \lg n$

5. Give asymptotic upper and lower bounds for T(n) in each of the following recurrences. Assume that T(n) is constant for n <= 2. Make your bounds as tight as possible, and justify your answers.

   $T(n) = 2T(n/2) + n^4$

   $T(n)= T(7n/10) + n$

   $T(n) = 16T(n/4) + n^2$

   $T(n) = 7T(n/3) + n^2$

   $T(n) = 7T(n/2) + n^2$

   $T(n) = 2T(n/4) + n^{1/2}$

   $T(n) = T(n-2) + n^2$

   $T(n) = T(9n/10)+\Theta(n)$

   $T(n) = T(n-1)+\Theta(n)$

   $T(n) = T(n/4)+T(n/2)+\Theta(n^2)$

6. Answer the questions about quicksort below.
   A. Show the quicksort's best-case running time is Ω(n lg n).
   B. When does the worst-case happen? Show the worst-case running time.
   C. Show the randomized-quicksort's expected running time.
   D. Suppose that all element values are equal. What would be randomized quicksort's running time in this case?

7. Describe the Merge Sort algorithm and give the recurrence relation for the algorithm. Finally, solve the recurrence and give the upper bound of its running time.

8. Prove that any comparison sort algorithm requires Ω(n lg n).

9. Explain the stable sorting property and explain why it is important for implementing Radix sort.

10. Prove that COUNTING-SORT is stable.

11. Which of the following sorting algorithms are stable: insertion sort, merge sort, heapsort, and quicksort? Give a simple scheme that makes any sorting algorithm stable. How much additional time and space does your scheme entail?

12. Suppose you have a connected undirected graph, and you want to find out if the graph has an odd-length cycle (i.e., length of the cycle is one of 3, 5, 7... etc.). Explain how you would solve this problem using BFS.

13. Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one. (It should not output all cycles in the graph, just one of them.) The running time of your algorithms should be O(m+n) for a graph with n nodes and m edges.

14. Explain how you would use the Breadth First Search (BFS) algorithm to test the bipartiteness of a graph.

15. Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer. Describe a greedy algorithm to make change consisting of quarters (25 cents), dimes (10 cents), nickels (5 cents), and pennies (1 cents). Prove that your algorithm yields an optimal solution.

16. Radix sort is a linear time sorting algorithm that sorts numbers digit by digit, least-significant digit first, with an auxiliary stable sort algorithm. Assume that we use counting sort as the auxiliary sorting algorithm. For your information, counting sort takes O(n + k) time to sort n numbers in the rage from 0 to k-1. Now, explain how radix sort can sort n numbers in the rage from 0 to n^d-1 in Θ(dn) time.

17. In order to implement a dynamic order statistics algorithm ("OS-Select"), suppose that we decided to augment the underlying data structure, RB-Tree. <u>Describe how we could augment</u> the standard RB-Tree to implement the functionality and <u>give the OS-Select algorithm</u> that uses the augmented information. Finally, <u>describe the cost</u> of search (for i^th^ smallest), insert and delete operations with the augmented RB-Tree.

18. For n distinct elements x1, x2, ..., xn with positive weights w1, w2, ..., wn such that $\sum_{i=1}^{n} w_i = 1$, the weighted median is the element xk satisfying

$$\sum_{x_i < x_k} w_i < \frac{1}{2} \quad \text{and} \quad \sum_{x_i > x_k} w_i \le \frac{1}{2}$$

For example, if the elements are 0.1, 0.35, 0.05, 0.1, 0.15, 0.05, 0.2 and each element equals its weight (i.e., wi = xi for i = 1, 2, ..., 7), then the median is 0.1, but the weighted median is 0.2.

Show how to compute the weighted median in O(n) expected time using an order statistics algorithm such as Rand-Select that we learned in class.

19. Explain how the cut property can be used to prove the correctness of Prim's algorithm.

20. Suppose we are given an instance of the Minimum Spanning Tree Problem on a graph G, with edge costs that are all positive and distinct. Let T be a minimum spanning tree for this instance. Now suppose we replace each edge cost c_e by its square, c_e^2, thereby creating a new instance of the problem with the same graph but different costs. Do you think that T still is a minimum spanning tree for this new instance? If you believe it is true give a short explanation. Otherwise, give a counterexample.

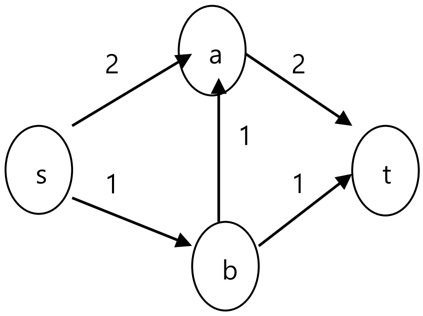
21. Describe Dijkstra's algorithm for finding shortest path, give a counter example for which it will fail, explain how the counter example can be solved using a Dynamic programming algorithm, formulate its recurrence structure, and explain the time and space requirements of the algorithm.

22. Is the Ford-Fulkerson algorithm guaranteed to terminate? If so, give a brief sketch of the proof for termination, and if not, explain why, in your own words using plain language.

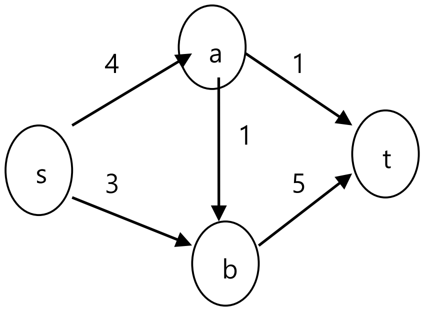
23. In order to speed up the Max Flow algorithm, we introduced the "capacity scaling" method. Briefly explain the key idea behind of it that enables the speed-up.

24. Answer the following questions.

 A. Consider the following flow network and list all the minimum s-t cuts in the network.



 B. What is the minimum capacity of an s-t cut in the following flow network?



25. Describe briefly the difference between the optimization version and the decision version of a problem. Give an example for each using Independent Set problem (i.e., write an optimization version and a decision version of the independent set problem).

26. Briefly describe how you can prove that Vertex Cover is NP-complete using the fact that 3-SAT is NP-complete.