B3043002 ALGORITHMS

Department of Computer Science and Engineering
close-book midterm exam

You may answer the questions in any order. Unless the details are requested, you may directly use anything that we have shown in class in a "black-box" way. Notice that dishonest behaviors and attempts will be punished most seriously.

1. (12%) **True/False**. To get credit, you must give reasons for your answers !!

   (a) If $f(n) = \Omega(g(n))$ and $g(n) = O(f(n))$, then $f(n) = \Theta(g(n))$.

   (b) There is some input for which Randomized Quick-Sort always runs in $\Theta(n^2)$ time.

   (c) $O(1/\log n) = 1/O(\log n)$

   (d) $f(n) = 100n + \log n$ and $g(n) = n + \log^2 n \Rightarrow f(n) = \Omega(g(n))$

2. (20%) Give asymptotically **tight** bound for $T(n)$ in each of the following recurrence:

   (a) $T(n) = 5T(n/2) + \sqrt{n}$

   (b) $T(n) = 64T(n/4) + 8^{\lg n}$

   (c) $T(n) = T(n-1) + n(n-1), T(1) = 1$

   (d) $T(n) = T(\sqrt{n}) + 1, T(2) = 1$

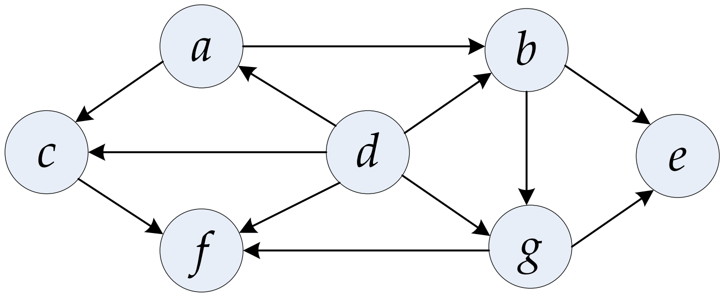
3. (10%) Give the definitions of three asymptotic notations, O, Θ and ω. Use the definition of Big-O to prove or disprove: $3n^4 + 5n + 2 = O(n^3)$.

4. (15%) What is a *stable* sorting method? We consider four sorting algorithms as Selection-Sort, Quick-Sort, Merge-Sort and Heap-Sort. Based on the comparison model, please list their best, average and worst time complexities.

5. (10%) What is an *in-place* sorting algorithm? In general, Quick-Sort is not in-place. Modify it to be an in-place sorting algorithm and analyze the time complexity. Also give an clear example to show how your algorithm works.

6. (10%) We say that a point (x_1, y_1) *dominates* (x_2, y_2) if both $x_1 > x_2$ and $y_1 > y_2$. A point is called a *maxima* if no other point dominates it. Design an $O(n \log n)$-algorithm that, given a set of n 2-D points, finds all maxima points of the set. Give a clear example to illustrate how your algorithm works.

7. (10%) Does any directed graph have a topological sort on its vertices? Design an efficient algorithm to linearize (topologically sort) the vertices in a graph. Please use your algorithm to find a topological sort step by step in the following graph. (Here, you don't need to justify your algorithm.)



8. (10%) A *d-ary heap* is like a binary heap, but (with one possible exception) non-leaf nodes have d children instead of just 2. Please answer the following questions in terms of n and d.

   (a) (2%) If the heap is represented by an array A, describe how to find the parent and the (at most) d children of element $A[i]$.

   (b) Give an efficient implementation of EXTRACT-MAX in a d-ary max-heap and analyze its running time.

   (c) Ben wants to implement Heap-Sort using the d-ary heap. He chooses $d = 1$ and argues that for this choice of d, the only operation required in the Heap-Sort algorithm is BUILD-HEAP. Since BUILD-HEAP takes $O(n)$ time, he can actually sort in $O(n)$ time!! Find the fallacy in Ben's argument. What familiar sorting algorithm is d-ary Heap-Sort really performing for $d = 1$?

9. (10%) A *bipartite graph* is a graph $G = (V, E)$ whose vertices can be partitioned into two disjoint sets ($V = V_1 \bigcup V_2$ and $V_1 \bigcap V_2 = \emptyset$) such that there are no edges between vertices in the same set (e.g, if $u, v \in V_1$, then $(u, v) \notin E(G)$). Design a linear-time algorithm to determine whether an undirected graph G is bipartite. Also give a clear example to illustrate how your algorithm works.