Exam Logistics:

- Answer any 6 of 7 questions. If all 7 are attempted, only the first 6 will be corrected and considered for the total score.
- Each question is worth 5 marks, and requires roughly 15 minutes' worth of effort.
- The midterm exam accounts for 15% of the final score.

Instructions for the Exam:

- Use appropriate indentation when writing pseudocode. This is strongly encouraged.
- What is expected when providing an algorithm:
  1. Algorithm must be written as pseudocode, and not descriptive format.
  2. Add comments wherever necessary to convey your ideas clearly.
  3. Do not provide code in any specific programming language.
  4. The following must be provided when asked to analyze an algorithm:
     (a) Proof of correctness.
     (b) Justification of running time.
     (c) As required, space complexity.

Questions (5 marks each):

1. For each of these cases, give an $O(V + E)$ algorithm that, given a directed graph $G = (V, E)$,:
   (a) computes its component graph. Make sure that there is at most one edge between two vertices in the component graph that your algorithm produces.
   (b) constructs another graph $G'(V, E')$ such that G and G' have the same strongly connected components, both G' and G have the same component graph, and $|E'|$ is as small as possible.
2. A PERT (Program Evaluation and Review Technique) chart is a network diagram of nodes and edges that visually maps project tasks, their dependencies, and timelines. An alternative way to represent a PERT chart is similar to the DAG in Figure 1. Vertices and edges represent tasks and sequencing constraints, respectively. For instance, edge (u, v) indicates that task u must be performed before task v. Vertices have weights, and not edges. Modify the DAG-SHORTEST-PATHS procedure that finds a longest path in a DAG with weighted vertices in linear time.
3. Write the pseudocode for binary search of a search key in a sorted array, for both iterative and recursive variants of the algorithm. Argue that the running time of binary search is $\Theta(n \lg n)$.
4. Show how to use a depth-first search of an undirected graph G to identify the connected components of G, so that the depth-first forest contains as many trees as G has connected components. More precisely, show how to modify depth-first search so that it assigns to each vertex v an integer label $v.cc$ between 1 and k, where k is the number of connected components of G, such that $u.cc = v.cc$ iff u and v belong to the same connected component.
5. Justify the asymptotically tight upper and lower bounds for $T(n)$ for the following recurrence cases.
   (a) $T(n) = 2.T(n/2) + n^4$
   (b) $T(n) = T(7n/10) + n$
   (c) $T(n) = 16.T(n/4) + n^2$
   (d) $T(n) = 7.T(n/3) + n^2$
   (e) $T(n) = 7.T(n/2) + n^2$

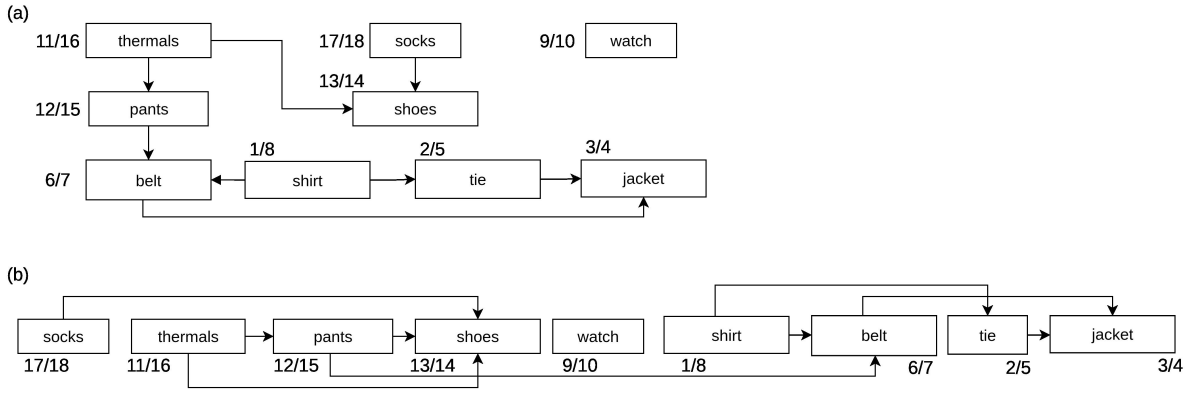

Figure 1: (a) Mr. XYZ topologically sorts his clothing when getting dressed. Each directed edge (u, v) means that garment u must be put on before garment v. The discovery and finish times from a depth-first search are shown next to each vertex. (b) The same graph is shown topologically sorted, with its vertices arranged from left to right in order of decreasing finish time. All directed edges go from left to right.

6. The problem size n is typically assumed to be a power of a convenient base (*e.g.*, 2) for running time analysis. But this need not be the case always, which leads to reformatting the problem. A few examples:

   (a) For computing the power a^n where $n \in \mathbb{Z}^+$ and $n > 1$, the brute force method is to multiply a repeatedly using $(n-1)$ multiplications. Describe an algorithm for which $O(\log n)$ multiplications suffice, including cases when n is not a power of 2.

   (b) Show how Strassen's algorithm, which is used for two $n \times n$ matrices A and B, works when n is not a power of 2.

   (*Write the algorithm, proof of its correctness, and running time analysis for each subproblem.*)

7. Consider the selection sort algorithm. What loop invariant does this algorithm maintain? Why does it need to run for only the first $(n-1)$ elements, rather than for all n elements? Give the worst- and average-case running time in O- and Θ-notations, respectively. Is the best-case running time in Ω-notation any better?