Banner number: Name:

Practice Final Exam

CSCI 3110: Design and Analysis of Algorithms

Fall 2015

Group 1		Group 2		Group 3		
Question 1.1		Question 2.1		Question 3.1		$\sum$
Question 1.2		Question 2.2		Question 3.2		
Question 1.3		Question 2.3		Question 3.3		
Question 1.4		Question 2.4				
$\sum$		$\sum$		$\sum$		

Instructions:

- The questions are divided into three groups: Group 1 (40 marks = 40%), Group 2 (36 marks = 36%), and Group 3 (24 marks = 24%). You have to answer **all questions in Groups 1 and 2** and **exactly two question in Group 3**. In the above table, put check marks in the **small** boxes beside the two questions in Group 3 you want me to mark. If you select less than or more than two questions, I will randomly choose which two to mark.
- Provide your answer in the box after each question. If you absolutely need extra space, use the backs of the pages; but try to avoid it. Keep your answers short and to the point.
- **You are not allowed to use a cheat sheet.**
- If you are asked to design an algorithm and you cannot design one that achieves the desired running time, design a slower algorithm that is correct. A correct and slow algorithm earns you 50% of the marks for the algorithm. A fast and incorrect algorithm earns 0 marks.
- When designing an algorithm, you are allowed to use algorithms and data structures you learned in class as black boxes, without explaining how they work, as long as these algorithms and data structures do not directly answer the question.
- **Read every question carefully before answering. In particular, do not waste time on an analysis if none is asked for, and do not forget to provide one if it is required.**
- **Do not forget to write your banner number and name on the top of this page.**
- **This exam has 13 pages, including this title page. Notify me immediately if your copy has fewer than 13 pages.**

Question 1.1 (Average-case and worst-case running time) **12 marks**

(a) Define the term "average-case running time of a deterministic algorithm".

(b) Define the term "worst-case running time of a deterministic algorithm".

(c) Explain the difference between a deterministic algorithm and a randomized algorithm.

(d) Explain the difference between the average-case running time of a deterministic algorithm and the expected running time of a randomized algorithm.

Question 1.2 (Asymptotic growth of functions) 5 marks

(a) Give a formal definition of the set $\Theta(f(n))$.

(b) Give a formal definition of the set $o(f(n))$.

Question 1.3 (Amortized analysis) 8 marks

Let D be a data structure whose operations have an amortized cost of at most t.

(a) Can you give a guaranteed bound on the cost of a sequence of m operations on D? If so, state this bound. Justify your answer.

(b) Can you give a guaranteed bound on the cost of an individual operation on D? If so, state this bound. Justify your answer.

(c) If t is not the amortized but the expected cost of operations on D, can you give any of the above guarantees? If so, which ones. Justify your answer.

Question 1.4 (Complexity classes) — 15 marks

(a) Formally define the complexity class P.

(b) Formally define the complexity class NP.

(c) Define what it means for a problem to be NP-hard.

(d) Define what it means for a decision problem to be NP-complete.

(e) Can an optimization problem be NP-complete? Justify your answer.

Question 2.1 (Sorting) — 6 marks

Provide an algorithm that can sort n numbers in $O(n\sqrt{n})$ time. You are allowed to use any algorithms you know, even sorting algorithms, as building blocks without stating their details. To earn full marks, your algorithm description should be as simple as possible while giving a clear description of the algorithm.

Question 2.2 (Lower bounds and reductions) — 10 marks

Given that sorting using only comparisons requires $\Omega(n \lg n)$ time, prove that there cannot exist a comparison-based priority queue implementation that supports both INSERT and DELETEMIN operations in $o(\lg n)$ time. (If such a priority queue implementation existed, you should be able to design a sorting algorithm with running time $o(n \lg n)$.)

Question 2.3 (Kruskal's algorithm) 10 marks

Given a connected graph G such that no two edges of G have the same weight, prove that Kruskal's algorithm finds a minimum spanning tree of G. You may refer to the high-level description of the algorithm:

KRUSKAL(G)

1 $T =$ a graph with the same vertex set as G and no edges
2 **while** T is not connected
3 let e be the edge in G with minimum weight among all edges whose endpoints belong to different components of T
4 add edge e to T
5 **return** T

Question 2.4 (Recurrence relations) 10 marks

Solve the following two recurrences, that is, determine a function $f(n)$ for each recurrence such that $T(n) \in \Theta(f(n))$. Use the Master Theorem to solve one of them and induction to solve the other one.

(a) $T(n) = 4T(n/3) + \Theta(n \lg n)$

(b) $T(n) = T(n/5) + T(5n/7) + \Theta(n)$

Question 3.1 (Divide and conquer) — 12 marks

Let A be an array containing n numbers (positive and negative). Develop an algorithm that finds the two indices $1 \leq i \leq j \leq n$ such that $S_{ij} := \sum_{k=i}^{j} A[k]$ is maximized. For example, in the array $A = [10, -12, 5, 7, -2, 4, -11]$, the sub-array $A[3, 6]$ has the sum $S_{3,6} = 5 + 7 - 2 + 4 = 14$ and no other sub-array contains elements that sum to a value greater than 14, so for this input the algorithm should output $(3, 6)$. The running time of your algorithm should be $O(n \lg n)$. Justify briefly why your algorithm achieves this running time and why it gives the correct answer.

Extra space for Question 3.1

Question 3.2 (Dynamic programming) — 12 marks

Let S be a sequence of m integer pairs $\langle (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m) \rangle$. Each of the values x_i and y_i, for all $1 \leq i \leq m$, is an integer between 1 and n. A *domino sequence* is a subsequence $\langle (x_{i_1}, y_{i_1}), (x_{i_2}, y_{i_2}), \ldots, (x_{i_t}, y_{i_t}) \rangle$ such that $1 \leq i_1 < i_2 < \cdots < i_t \leq n$ and, for all $1 \leq j < t$, $y_{i_j} = x_{i_{j+1}}$. Note that it isn't necessarily true that $i_{j+1} = i_j + 1$, that is, the elements of the domino sequence don't have to be consecutive in S, but they have to appear in the right order.

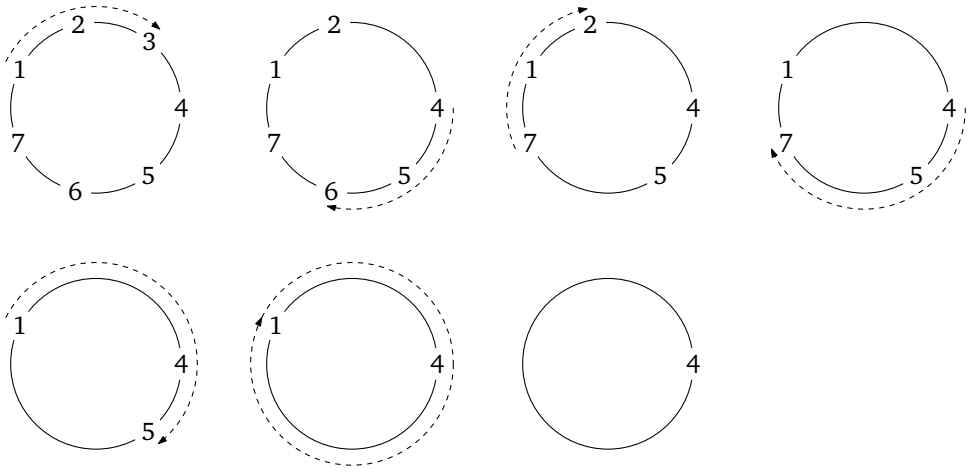
Example: For $S = \langle (1,3), (4,2), (3,5), (2,3), (3,8) \rangle$, both $\langle (1,3), (3,5) \rangle$ and $\langle (4,2), (2,3), (3,8) \rangle$ are domino sequences.

Use dynamic programming to find a longest domino sequence of S in $O(n+m)$ time. Argue briefly that the running time of your algorithm is indeed $O(n+m)$ and that its output is indeed a longest domino sequence of S.

Extra space for Question 3.2

Question 3.3 (Applications of data structures) 12 marks

Assume n people form a circle and number them in the order they appear around the circle. Now, for some integer $1 \leq m \leq n$, we repeat the following process, starting with person 1, until no people are left in the circle: Walk around the circle until reaching the mth person after the person we started at. Remove this person from the circle and continue using her successor as the starting point of the next search. The order in which we remove people from the circle is called the (n, m)-*Josephus permutation*. The following example illustrates that the $(7, 3)$-Josephus permutation is $\langle 3, 6, 2, 7, 5, 1, 4 \rangle$.



Develop an algorithm that can find the (n, m)-Josephus permutation in $O(n \lg n)$ time for any pair (n, m) such that $1 \leq m \leq n$. Justify briefly why your algorithm achieves this running time and why it gives the correct answer.

Extra space for Question 3.3