NAME: _____

CMPSCI 611
Advanced Algorithms
First Midterm Exam Fall 2017: Draft Solutions

A. McGregor 11 October 2017

DIRECTIONS:

- Do not turn over the page until you are told to do so.
- This is a *closed book exam.* No communicating with other students, or looking at notes, or using electronic devices. You may ask the professor or TA to clarify the meaning of a question but do so in a way that causes minimal disruption.
- If you finish early, you may leave early but do so as quietly as possible. The exam script should be given to the professor.
- There are five questions. All carry the same number of marks but some questions may be easier than others. Don't spend too long on a problem if you're stuck – you may find that there are other easier questions.
- The front and back of the pages can be used for solutions. There are also a blank page at the end that can be used. If you are using these pages, clearly indicate which question you're answering. Remember that the best answers are those that are clear and concise (and of course correct). Most questions can be answered in two or three sentences.

1	/10
2	/10
3	/10
4	/8+2
5	/10
Total	/48+2

Question 1. For each of the following statements, indicate whether they are TRUE or FALSE by circling the appropriate answer. No justification is required.

1. If the running time $T(n)$ of an algorithm satisfies $T(n) = O(n^2)$ and $T(n) = \Omega(n^2)$ then it also satisfies $T(n) = \Theta(n^2)$.
   Answer: TRUE.

2. If a subset system $(E, \mathcal{I})$ doesn't satisfy the cardinality property, there exist subsets $A, B \in \mathcal{I}$ such that $|B| > |A|$ and $A + e \notin \mathcal{I}$ for all $e \in B - A$.
   Answer: TRUE.

3. If the complex number z is a 4th root of unity, then it is also an 8th root of unity.
   Answer: TRUE.

4. Consider an undirected graph where all edges have weight at least 1. If there are exactly two edges with weight 1 then any minimum spanning tree includes both of these edges.
   Answer: TRUE.

5. If E consists of n elements and $\mathcal{I}$ consists of 2^n subsets of E, then $(E, \mathcal{I})$ is a matroid.
   Answer: TRUE.

Question 2. No justification required for your answers.

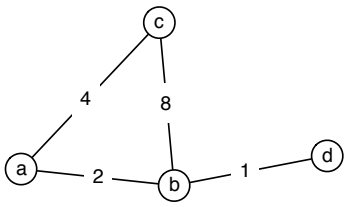
1. Consider the following undirected graph. The number on an edge indicates its weight.

   Weight of minimum spanning tree = 7

   Weight of maximum weight matching = 8

   Length of shortest path between c and d = 7

   Diameter of the graph = 7



   For diameter and shortest paths, you should treat the weight as the length of the edge.

2. What is the adjacency matrix of the graph considered above? You should ignore the weights on the edges.
   Answer:

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

3. Suppose that a divide and conquer algorithm solves a problem of size n by first solving three instances of size $n/2$ and then taking $O(n)$ additional basic steps. If the time to create the three instances is $O(n^2)$, what is the running time, $T(n)$, of the algorithm.
   Answer: $T(n) = 3T(n/2) + O(n^2)$ since $3 \leq 2^2$, $T(n) = O(n^2)$ by the Masters Theorem.

4. If $T(n) = T(n-1) + n$ for $n > 1$ and $T(1) = 1$, then $T(n) = \Theta(n^x)$ for what value of x?
   Answer: $T(n) = n + (n-1) + (n-2) + \ldots + 1 = n(n-1)/2 = \Theta(n^2)$. So $x = 2$.

Question 3. As part of an ongoing study, scientists in Iceland have been logging the noon temperature over the last n days. Let $T[i]$ be the temperature on day i for $i = 1, 2, \ldots, n$. Temperatures can be positive or negative.

1. Design an algorithm that finds, given a value of r, finds the set of r consecutive days with the maximum average temperature. Analyze the running time of your algorithm but no proof of correctness is required. The running time should not depend on r.

   Answer:

   Step 1. Let $R[1] = T[1] + T[2] + \ldots + T[r]$

   Step 2. For $i = 2, \ldots, n - r + 1$, let $R[i] = R[i-1] + T[i-1+r] - T[i-1]$

   Step 3. Return the value of j that maximizes $R[j]$.

   The returned value of j is the first day of the r consecutive days with the max average temperature. Each of the three steps takes $O(n)$ so the total running time is $O(n)$.

2. Consider the following algorithm that claims to determine whether there is a set of consecutive days whose average temperature is exactly 0.

   Step 1. Let $C[1] = T[1]$ and for $i = 2, ..., n$, let $C[i] = T[i] + C[i-1]$

   Step 2. Sort C by increasing value using MergeSort and let C' be the resulting array.

   Step 3. If there exists i such that $C'[i] = C'[i+1]$, output "yes". Output "no" otherwise.

   (a) What is the running time of this algorithm?
   Answer: $O(n \log n)$ since step 1 takes $O(n)$ time, step 2 takes $O(n \log n)$ time, and step 3 can be implemented in $O(n)$ time.

   (b) Is the algorithm correct? Either provide an example where the algorithm returns the wrong answer or provide a brief proof of correctness.
   Answer: It's incorrect, e.g., $T = [10, -3, -7, 2]$ yields $C = [10, 7, 0, 2]$ which has no duplicate values even though the average of the first 3 days is 0.[1]

3. Rather than releasing all the data exactly, the scientists release a rough approximation of their reading. We say list L is a k-bucket histogram if there is *at most* $k - 1$ values of i such that $L[i] \neq L[i+1]$. For example, $L = [6, 6, 6, 2, 2, 3, 3, 3, 3, 2, 2, 4]$ is a 5-bucket histogram of length 12. Design a dynamic programming algorithm that finds the k-bucket histogram L of length n that minimizes the following quantity:

$$\text{error}(L, T) = \sum_{i=1}^{n} (L[i] - T[i])^2$$

   Hint: You may assume that for any set of real numbers S, $\sum_{e \in S}(e - x)^2$ is minimized by setting x to be average$(S) = \sum_{e \in S} e/|S|$. You may wish to consider the value $D[j, k']$ that is the minimum possible value of $\sum_{i=1}^{j}(L[i] - T[i])^2$ where L is a k'-bucket histogram.

   (a) Briefly describe your algorithm.
   Answer:

   Step 1. For all $i \leq j$, let $a_{ij} = \text{average}(T[i], T[i+1], \ldots, T[j])$

[1] For what's it worth, it correct as long as $n > 1$ and no interval starting one the first day has average zero and we still gave marks for an argument along the following lines: $C[i]$ is the sum of temperatures up to and including the ith day. So $C[i] = C[j]$ for some $i < j$ iff the sum of all temperatures on days $i + 1, i + 2, \ldots, j$ is 0, and hence the average on these days is zero. Furthermore, $C[i] = C[j]$ for some $i < j$ iff two adjacent entries in C' are equal.

Step 2. For all $i \leq j$, let $c_{ij} = \sum_{\ell=i}^{j} (T[\ell] - a_{i,j})^2$
Step 3. For $j = 1, \ldots, n$. Let $D[j, 1] = c_{1,j}$
Step 4. For $j = 1, \ldots, n$; For $k' = 2, \ldots, k$, let

$$D[j, k'] = \min_{i<j} \left(D[i, k' - 1] + c_{i+1,j} \right)$$

(b) Briefly explain why your algorithm returns the correct answer.
Answer: The best k' bucket histogram for the first i entries of T consists of, for some value of $j < i$, the best $k' - 1$ bucket histogram for the first j entries of T concatenated with $i-j$ values identical to the average of the entries $T[j+1], \ldots, T[i]$.

(c) Analyze the running time of your algorithm.
Answer: The total time is $O(n^3)$: Step 1 can be implemented in $O(n^3)$ time (or smaller if you are careful), Step 2 in $O(n^3)$ time, Step 3 in $O(n)$ time, and Step 4 in $O(n^2 k)$ time.

Question 4. Let $\mu(n)$ be the time to multiply two $n \times n$ matrices together. For simplicity, you may ignore any dependence on the size of the entries of the matrices. Given a matrix A, the square of the matrix is defined as $A^2 = A \times A$ and, more generally, for $p \geq 2$, the pth power is defined as $A^p = A \times A \times \ldots \times A$ where the product consists of p copies of A. The faster the algorithms you design, the more marks you will get. Your answers should include a *brief description* of your algorithms and an analysis of the running time (perhaps in terms of $\mu(n)$). You *do not need* to prove correctness.

1. Design an algorithm for computing the pth power of a matrix A where p is a power of two. You may assume the existence of a subroutine to multiple matrices together.
Answer: Let `power`(A, p) be the algorithm that returns the pth power of A. Let `product`(A, B) be the subroutine that returns the product of two matrices A and B. Then `power`(A, p) is defined as follows:
   - If $p = 1$, return A.
   - If $p > 1$, let $B =$ `power`$(A, p/2)$ and return `product`(B, B)

   There are $O(\log p)$ recursive calls each of which takes $\mu(n)$ time. So the total running time is $O(\mu(n) \log p)$.

2. Design an algorithm for computing the pth power of a matrix D where $D[i, j] = 0$ if $i \neq j$ and $D[i, i] \in \{-1, 1\}$ for all i. The algorithm only needs to output the non-zero entries of D^p. In this part of the question, you *may not assume* p is a power of two.
Answer: If p is even, output (the non-zero entries of) the identity matrix. If p is odd, output (the non-zero entries of) D. Running time is $O(n)$.

3. **Extra Credit.** Suppose you have developed an algorithm that can find the square of a matrix in $O(n^2)$ time. Using this algorithm as a subroutine, find an algorithm that multiplies two $n \times n$ different matrices together in $O(n^2)$ time.
Answer: Let $C = \begin{pmatrix} 0 & B \\ A & 0 \end{pmatrix}$. Then $C^2 = \begin{pmatrix} BA & 0 \\ 0 & AB \end{pmatrix}$. Hence if we can compute $C \times C$ in $O(n^2)$ then we have computer AB in $O(n^2)$.

Question 5. A thief breaks into a house and finds n objects that could be stolen. The ith item is valued at $v_i > 0$ dollars and weighs $w_i \in \{1, 2, \ldots, W\}$ kilograms. The thief can carry at most W kilograms where W is some positive integer.

1. Let $D[i, j]$ be the maximum total value that can be stolen when the thief can carry at most j kilograms and may pick any subset of the first i items. Write a formula for $D[i, j]$ assuming $D[i', j']$ has already been computed for all $i' < i$ and $j' \leq j$.

$$D[i,j] = \begin{cases} \min(D[i-1,j], D[i-1,j-w_i] + v_i) & \text{if } j \geq w_i \\ D[i-1,j] & \text{otherwise} \end{cases}$$

2. Based on your above answer, write a dynamic program for computing the maximum total value of a set of objects that the thief can carry. You do not need to argue correctness but do need to state the running time.

   Answer:

   Step 0. For $i = 1, \ldots, n$, Let $D[i, 0] = 0$.

   Step 1. For $i = 1, \ldots, n$; For $j = 1, \ldots, W$: Set $D[i, j]$ using the above formula

   Step 2. Return $D[n, W]$.

   The running time is $O(nW)$.

3. Consider the greedy algorithm that orders the objects by decreasing value and adds each object to the collection to be stolen if it does not violate the weight constraint. Give an example with $W = 10$ and three objects with different values such that this greedy does *not* maximize the total value stolen.

   $w_1 = 10$ $\quad v_1 = 4$ $\quad w_2 = 5$ $\quad v_2 = 3$ $\quad w_3 = 5$ $\quad v_3 = 2$

4. Suppose that the objects are *divisible*, i.e., it is possible to take any fraction of each object. For example, if one of the objects was milk, you can take all the milk, none of the milk, or any amount in between. If you take a fth fraction of object i the weight is $f \times w_i$ and the value is $f \times v_i$. Let $r_i = v_i/w_i$. Design a greedy algorithm to find the maximum total value that can be stolen. You should not assume the items are initially sorted in any particular way.

   (a) Briefly describe your algorithm.
   Answer: Sort jobs by decreasing r_i values such that after sorting $r_1 > r_2 > r_3 > \ldots$. Then add entire items in this order until we reach an item that we can't add because of the weight restriction; we then add the largest fraction of this item we can such that we violate the weight constraint.

   (b) What is the running time?
   Answer: The running time if $O(n \log n)$ because of the need to sort the items.

   (c) Prove the correctness. **Hint:** Let f_i be the fraction of item i that your algorithm takes and consider another solution where g_i is the fraction of item i that is taken. If the solutions are different, show that the solution with the g_i fractions is not optimal. For simplicity, you may assume all the v_i/w_i values are different. Your analysis does not need to involve matroids.

Answer: If $\sum_i w_i < W$, the greedy algorithm is optimal since it picks all items; hence we may assume $\sum_i f_i w_i = W$. If $\sum_i g_i w_i < W$, we could increase the value attained by the g solution and so it's not optimal. Hence, we may assume

$$\sum_i f_i w_i = \sum_i g_i w_i = W \ .$$

If $g \neq f$ then there exists $i < j$ such that $\geq f_i w_i > g_i w_i$ and $f_j w_j < g_j w_j$. Hence, we can consider increasing g_i by δ/w_i and decreasing g_j by δ/w_j for some small value of δ. This ensures the weight constraint is still satisfied but increases the total value by $\delta v_i/w_i - \delta v_j/w_j = \delta(r_i - r_j) > 0$ and hence g can't have been optimal.