CPSC 320 Sample Midterm 1
October 2013

[9] 1. State whether each of the following statements is true or false. Justify each of your answers briefly.

[3] a. You should use decision trees if you want to prove a lower bound on the worst-case running time of an algorithm you wrote to schedule final exams.

Solution : This is false: we use decision trees to prove lower bounds on the worst-case running time of *classes* of algorithms (all algorithms to solve a given problem). For a specific algorithm, other simpler methods should be used.

[3] b. For any given set of preferences, the Gale-Shapley algorithm always returns the same pairing, whether men or women are the ones proposing. Assume that no one likes two or more members of the opposite gender equally,

Solution : This is false. If the women's preferences do not match the men's, then who proposes may result in different stable matchings. For instance, with the following preference lists:

$$\begin{array}{llll} w_1: & m_1 & m_2 & \qquad\qquad m_1: \quad w_2 \quad w_1 \\ w_2: & m_2 & m_1 & \qquad\qquad m_2: \quad w_1 \quad w_2 \end{array}$$

then when women propose, the algorithm returns the pairs $\{w_1, m_1\}$ and $\{w_2, m_2\}$. However if men propose, then it returns the pairs $\{w_2, m_1\}$ and $\{w_1, m_2\}$.

[3] c. If the limit $\lim_{n\to\infty} f(n)/g(n)$ is equal to 0, then we can conclude that $f(n) \in O(g(n))$.

Solution : This is true. By the limit theorem, $f \in o(g)$, and since $o(g) \subseteq O(g)$ this implies that $f \in O(g)$.

[3] 2. Why is the interval scheduling algorithm we discussed in class greedy?

Solution : It is greedy because it builds its solution by repeatedly choosing the interval with the smallest finishing point, without worrying about the impact of this choice on later intervals.

[6] 3. Consider the following preference lists for four women $w_1, \ldots, w_4$ and men $m_1, \ldots, m_4$:

$$\begin{array}{llllll} w_1: & m_4 & m_2 & m_1 & m_3 & \qquad m_1: \quad w_3 \quad w_1 \quad w_2 \quad w_4 \\ w_2: & m_2 & m_4 & m_1 & m_3 & \qquad m_2: \quad w_4 \quad w_3 \quad w_2 \quad w_1 \\ w_3: & m_1 & m_3 & m_2 & m_4 & \qquad m_3: \quad w_4 \quad w_3 \quad w_1 \quad w_2 \\ w_4: & m_2 & m_4 & m_3 & m_1 & \qquad m_4: \quad w_2 \quad w_4 \quad w_1 \quad w_3 \end{array}$$

Fill in the following table that contains each of the proposals performed by the algorithm, as well as whether the man accepted or rejected the proposal. Assume that whenever several

women are not engaged, the woman with the lowest number among those will propose next. You do not need to fill out every row of the table if the algorithm terminates in fewer than 12 steps.

Solution :

	Woman name		Man name		accepts/rejects
Woman	w_1	proposes to man	m_4	who	accepts
Woman	w_2	proposes to man	m_2	who	accepts
Woman	w_3	proposes to man	m_1	who	accepts
Woman	w_4	proposes to man	m_2	who	accepts
Woman	w_2	proposes to man	m_4	who	accepts
Woman	w_1	proposes to man	m_2	who	rejects
Woman	w_1	proposes to man	m_1	who	rejects
Woman	w_1	proposes to man	m_3	who	accepts
Woman		proposes to man		who	
Woman		proposes to man		who	
Woman		proposes to man		who	
Woman		proposes to man		who	

[6] 4. Consider the two functions f, g: $\mathbf{N} \to \mathbf{R}^+$ defined by

$$f(n) = n^2$$
$$g(n) = \begin{cases} n & \text{if } n \text{ is even} \\ 5n^2 & \text{if } n \text{ is odd} \end{cases}$$

Prove that $f \in \Omega(g)$, but that $f \notin \omega(g)$.

Solution : First we prove that $f \in \Omega(g)$. Since $5n^2 \geq n$ for every $n \geq 1$, we see that for every $n \geq 1$

$$f(n) = n^2 = (1/5)5n^2 \geq (1/5)g(n).$$

Hence $f \in \Omega(g)$, using $c = 1/5$ and $n_0 = 1$. Now, we prove that $f \notin \omega(g)$. Negating the definition of ω, we see that $f \notin \omega(g)$ if there is a $c \in \mathbf{R}^+$ such that, for every $n_0 \in \mathbf{N}, \exists n \geq n_0$ such that $f(n) < cg(n)$. Pick $c = 1$: for every odd integer, $f(n) < g(n)$. So clearly f satisfies the condition, and hence $f \notin \omega(g)$.

[5] 5. Mr. Isulo, a famous alien computer scientist, has designed a greedy algorithm to solve the *Clique* problem (you don't need to know what it is) on a type of graphs called *circular arc graphs* (you don't need to know what they are either). His algorithm starts by choosing the vertex with the most neighbours.

Mr. Isulo wants to prove the following lemma: "Every circular arc graph has a maximum clique that contains the vertex with the most neighbours", but he eventually finds a counter-example to his conjecture. What does this imply for Mr. Isulo's algorithm, and why?

Solution : All of the solutions constructed by Mr. Isulo's algorithm will contain the vertex with the most neighbours, since a greedy algorithm never removes the vertices it has chosen. Because the lemma is false, his algorithm will not return the correct answer for some graphs (that is, it doesn't work).

[11] 6. Suppose that you are given a set E with n elements, and a collection $S_1, \ldots, S_m$ of subsets of E. A *Hitting Set* for $S_1, \ldots, S_m$ is a subset E' of E such that every S_i contains at least one element of E'. For instance, if $E = \{a, b, c, d, e\}$, and the subsets of E are $S_1 = \{a, b, d\}$, $S_2 = \{a, c\}$, $S_3 = \{a, d, e\}$, $S_4 = \{b, c, d\}$ and $S_5 = \{c, e\}$ then $\{b, c, e\}$ and $\{a, c\}$ are both hitting sets.

The *Hitting Set* problem consists in finding the smallest subset E' of E that is a hitting set for $S_1, \ldots, S_m$.

[7] a. Describe a greedy algorithm to solve the hitting set problem. Your algorithm does not need to always succeed at minimizing the number of the objects selected, but it should make a good attempt at it.

Solution : Here is one possible algorithm: choose the element x that belongs to the most subsets, and add it to the hitting set H. Remove the subsets that contain x from the list, and then repeat this step until no subsets are left.

This greedy algorithm will not always return the smallest hitting set. For instance, if $E = \{a, b, c\}$ and the subsets are $\{b, d\}$, $\{a, b, e\}$, $\{a, b, f\}$, $\{a, c, g\}$, $\{a, c, h\}$, $\{c, i\}$, then the algorithm will return $\{a, b, c\}$ (possibly using d instead of b, and i instead of c, or both), but the optimal solution is $\{b, c\}$. However this problem is *NP-Complete*, which as we will see towards the end of the course means that it is very very unlikely that it can be solved efficiently (and hence no greedy algorithm is likely to work).

[4] b. Analyze the time complexity of your algorithm from part (a). Specify only as much of your implementation (for instance, data structures) as needed for your analysis. There is no need to get the most efficient possible implementation of your algorithm, but excessively pessimistic analyses (e.g. $O(n^4)$ when an algorithm could be implemented to run in $O(n \log n)$ time) will not get full marks.

Solution : Suppose we maintain the number of sets that contain each element through the execution of the algorithm, initializing them at the beginning and updating the values whenever a set is deleted. Suppose moreover that we keep the elements in a max-heap, so that each update takes $O(\log |E|)$ time. Then the running time of the algorithm will be in $O((|S_1| + |S_2| + |S_3| + ... + |S_m|) \log |E|)$.