NAME: ____________________________

CMPSCI 611
Advanced Algorithms
Midterm Exam Fall 2012

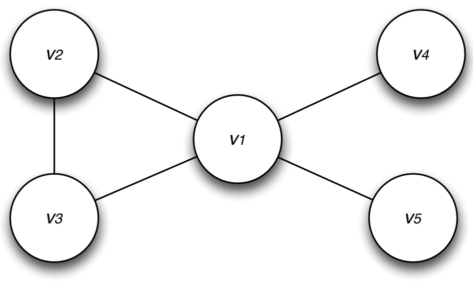
A. McGregor 16 October 2012

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. Further paper can be requested if required. However, the best answers are those that are clear and concise (and of course correct).
- The exam will finish at 12:30 pm.

1	/10
2	/10
3	/10
4	/10
5	/10
Total	/50

Question 1 (The Warm-Up Question). In this question, we consider the following undirected graph where the weight of each edge should be assumed to be one.



1. What is the size of the maximum matching in the graph?

   ANSWER: 2
2. What is the diameter of graph?

   ANSWER: 2
3. What is the adjacency matrix of the graph?

   ANSWER:

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

4. Is the graph bipartite?

   ANSWER: No
5. How many different spanning trees does the graph have?

   ANSWER: 3

Question 2 (Matroids). No proof are required in the first three parts of this question.

1. Let $E = \{a, b, c\}$ and $\mathcal{I} = \{\emptyset, \{a\}, \{b\}, \{a, c\}, \{b, c\}\}$. Is $(E, \mathcal{I})$ a subset system?

   ANSWER: No

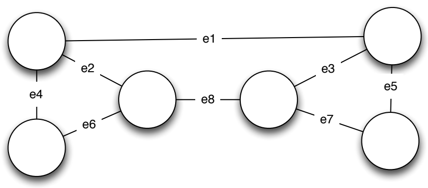
2. Let $E = \{a, b, c\}$ and $\mathcal{I} = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{a, c\}\}$. Is $(E, \mathcal{I})$ a matroid?

   ANSWER: Yes

3. State the Cardinality Theorem.

   ANSWER: $(E, \mathcal{I})$ is a matroid iff ($\forall A \subseteq E$ and $i, i' \in \mathcal{I}$ maximal in A, $|i| = |i'|$).

For the rest of the question, consider an arbitrary graph $G = (V, E)$ and let $\mathcal{I}$ be the set all subsets $E' \subseteq E$ such that each connected component of $G' = (V, E')$ has at most one cycle. For example, if G was the following graph



then $\{e_2, e_3, e_4, e_5, e_6, e_7\} \in \mathcal{I}$ but $\{e_1, e_2, e_3, e_4, e_6, e_8\} \notin \mathcal{I}$.

4. Prove that $(E, \mathcal{I})$ is a subset system for any graph.

   ANSWER: Suppose that $E' \subseteq E$ is such that each connected component of $G' = (V, E')$ has at most one cycle. Then for any $E'' \subseteq E'$, each connected component of $G'' = (V, E'')$ has at most one cycle since a) the removal of edges can not create cycles and b) the removal of edges will not result in any two existing cycles being in the same connected component. Hence $(E, \mathcal{I})$ is a subset system.

5. Prove that $(E, \mathcal{I})$ is a matroid for any graph.

   ANSWER: Consider an arbitrary subset $A \subseteq E$ and suppose that the connected components of (V, A) are $(V_1, A_1), (V_2, A_2), \ldots$. Suppose i is a maximal subset of A. Then $|i \cap A_j| = |V_j| - 1$ if (V_j, A_j) is acyclic and $|i \cap A_j| = |V_j|$ if (V_j, A_j) has a cycle. Therefore,

$$|i| = \sum_j |i \cap A_j| = |V| - \textit{(the number of } (V_j, A_j) \textit{ that are acyclic)} .$$

   Hence, all maximal subsets of A have the same size.

Question 3 (Making Change). Given an unlimited supply of coins of denominations $x_1, x_2, \ldots, x_n$, we wish to make change for a value v; that is, we wish to find a set of coins whose total value is v. This might not be possible: for instance, if the only available denominations are 5 and 10 then we can make change for 15 but not for 12. The problem we consider is:

Input: Positive integers $x_1, \ldots, x_n$ and v.

Output: Whether it is possible to make change for v using coins of denominations $x_1, \ldots, x_n$?

1. Give an $O(nv)$ time dynamic-programming algorithm. Include all necessary details.

   ANSWER: Define $A[i] = 1$ if it is possible to make change for value i and $A[u] = 0$ otherwise. Then we can compute $A[v]$ as follows

   (a) $A[0] = 1$

   (b) *For* $0 < u \leq v$, *let* $A[u] = \max_{j:x_j \leq u} A[u - x_j]$

   (c) *Return* $A[v]$

2. Prove that your algorithm is correct and takes $O(nv)$ time.

   ANSWER: The running time follows because each $A[u]$ value can be computed in $O(n)$ and there are v values to compute. The correctness follows because it is possible to make change for u iff there exists some $x_j \leq u$ such that it is possible to make change for $u - x_j$.

3. How would you improve your algorithm if $x_1, \ldots, x_n$, and v had a common factor?

   ANSWER: If $x_1, \ldots, x_n$, and v have a common factor t, then note that it is possible to make change for v/t from $x_1/t, x_2/t, \ldots, x_n/t$ iff it is possible to make change for v from $x_1, x_2, \ldots, x_n$. Hence, we can improve the algorithm by first dividing all values by t. The running time becomes $O(nv/t)$.

Question 4 (Finding A Shift). Suppose you are given an array $A[1 \dots n]$ of distinct sorted integers that has been circularly shifted k positions to the right. For example, $[35, 42, 5, 15, 27, 29]$ is a sorted array that has been circularly shifted $k = 2$ positions, while $[27, 29, 35, 42, 5, 15]$ has been shifted $k = 4$ positions. We can obviously find the largest element in A in $O(n)$ time.

1. Design an $O(\log n)$ time algorithm for finding the shift. You may assume n is a power of two.

   ANSWER: Note that the value of the shift is the index of the largest element. Hence, it suffices to find the index of the largest element.

   ```
   Max-Index($B[1 \dots m]$)
   (a) If $m = 1$, return 1
   (b) If $B[1] > B[m/2]$, return Max-Index($B[1 \dots m/2]$)
   (c) If $B[1] < B[m/2]$:
       i. If $B[m/2] < B[m/2+1]$, return $m/2$+Max-Index($B[m/2+1 \dots m]$)
       ii. If $B[m/2] > B[m/2+1]$, return $m/2$
   ```

2. Prove that your algorithm is correct.

   ANSWER: If $m = 1$ then the index of the maximum element in $B[1]$ is 1. If $B[1] > B[m/2]$ then the maximum element must be in the first $m/2$ coordinates since otherwise $B[1] < B[2] < \ldots < B[m/2]$ because the list was originally sorted. If $B[1] < B[m/2] < B[m/2+1]$ then the maximum element must be in the second $m/2$ coordinates since otherwise $B[m/2+1] < B[m/2+2] < \ldots < B[m] < B[1]$. If $B[m/2] > B[m/2+1]$ then the maximum element is in position $m/2$.

3. Prove that your algorithm takes $O(\log n)$ time.

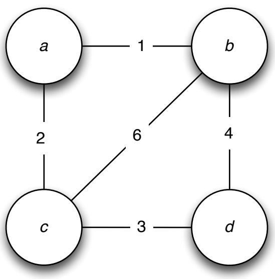
   ANSWER: The length of the list halves with each recursion. Hence, the depth of the recursion is $O(\log n)$. $O(1)$ work is required to check the values $B[1], B[m/2]$, and $B[m/2+1]$. Hence the total running time is $O(\log n)$.

Question 5 (Special Spanning Trees). Sometimes we need to find a spanning trees with additional special properties. Here's an example of such a problem.

Input: Undirected, connected graph $G = (V, E)$; edge weights w_e; subset of vertices $U \subset V$.

Output: Find the spanning tree of minimum total weight such that all nodes of U are leaves (there might be other leaves in this tree as well).

1. In the following graph, what is a) the minimum weight of a spanning tree and b) the minimum weight of a spanning tree in which node a is a leaf. The value of each edge represents the length of that edge.



ANSWER: a) 6 and b) 8.

2. Give an algorithm for this problem which runs in $O(|E|\log|E|)$ time. **Hint:** When you remove nodes U from the optimal solution, what is left? You may assume the correctness and running time of any algorithm analyzed in class.

ANSWER:

Algorithm:

(a) Use Kruskal's algorithm to find the minimum spanning tree (or forest if the graph is disconnected) of the induced graph on nodes $V \setminus U$

(b) For each node $u \in U$, add incident edge (u, v) with the smallest weight (break ties arbitrarily).

Running Time: *The running time is the $O(|E|\log|E|)$ since the running time of Kruskal's algorithm is $O(|E|\log|E|)$ and it takes $O(|E|)$ time to find the edge of smallest weight for each u.*

Correctness: *For the correctness, we first argue that, without loss of generality, the optimum solution includes the minimum spanning forest in the induced graph on nodes $V \setminus U$. Note that the optimal solution must include a maximal acyclic graph on $V \setminus U$ since adding only a single edge to each $u \in U$ can not complete a path between two nodes in $V \setminus U$. The cheapest maximal acyclic graph is the minimum spanning forest. Since we need to add exactly one edge between $V \setminus U$ and each $u \in U$, to minimize the total weight we should add the cheapest such edge.*