NAME: ______________________________

CMPSCI 611
Advanced Algorithms
Midterm Exam Fall 2010

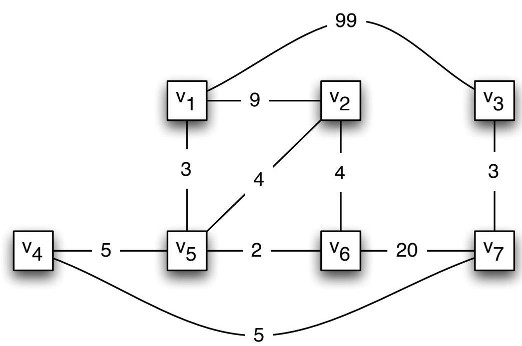
A. McGregor 19 October 2010

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 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 couple of blank pages 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.
- The exam will finish at 3:45 pm.

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

Question 1: In the first part of this question, consider the following undirected graph where the value of each edge represents the length of that edge:



1. What is the total length of the shortest path between v_1 and v_3?

   ANSWER: 3+5+5+3=16

2. What is the total length of the edges in a minimum spanning tree?

   ANSWER: Edges of a MST are $(v_5, v_6), (v_1, v_5), (v_3, v_7), (v_5, v_2), (v_4, v_5), (v_4, v_7)$. *Total weight is* $2 + 3 + 3 + 4 + 5 + 5 = 22$.

The next part of this question concerns an arbitrary weighted, undirected graph. For any two nodes u and v let $\delta_G(u, v)$ denote the length of the shortest path between u and v in the graph G. For each of the following statements, write whether they are true or false (no proofs required although including good reasoning *may* get partial credit even if you get the final answer wrong):

3. $\delta_T(u, v) = \delta_G(u, v)$ if T is a minimum spanning tree of G.

   FALSE. Consider the graph G with nodes $\{v_1, v_2, v_3\}$ *and edges* $\{(v_1, v_2), (v_2, v_3), (v_3, v_1)\}$ *where each edge has weight 1. All distances in G are 1 but in the minimum spanning tree, there is at least once pair of nodes that are distance 2 apart.*

4. $\delta_G(u, v) \leq \delta_G(u, w)$ implies $\delta_{G'}(u, v) \leq \delta_{G'}(u, w)$ where G' is the graph formed by adding 1 to each of the edge lengths in G.

   FALSE. Consider the graph with nodes $\{v, x, u, w\}$ *and edges* $\{(v, x), (x, u), (u, w)\}$ *with weights 1,1, and 2.5 respectively.*

5. $\delta_G(u, v) \leq \delta_G(u, w)$ implies $\delta_{G''}(u, v) \leq \delta_{G''}(u, w)$ where G'' is the graph formed by doubling each of the edge lengths in G.

   TRUE.

Question 2: This question is about the subset system $(E, \mathcal{I})$ where:

$$\begin{aligned} E &= \{a, b, c, d\} \\ \mathcal{I} &= \{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{a, c\}, \{b, c\}, \{c, d\}, \{a, b, c\}\} \end{aligned}$$

1. List all the maximal subsets in $\mathcal{I}$. Why can you conclude that $(E, \mathcal{I})$ is not a matroid?

   ANSWER: The maximal subsets are $\{c, d\}$ and $\{a, b, c\}$. If the subset system was a matroid, all the maximal subsets would have the same cardinality. So the subset system is not a matroid.

2. Consider the weighting function $w(a) = 1, w(b) = 2, w(c) = 3$, and $w(d) = 4$. What solution is returned by the greedy algorithm? How does this compare to the optimal solution?

   ANSWER: The greedy solution is $\{c, d\}$ and this has weight 7. This is the optimal solution!

3. Specify a weight function w on E such that the greedy algorithm doesn't return an optimal solution. Include the greedy solution and the optimal solution in your answer.

   ANSWER: Consider the weight function $w(a) = w(b) = w(c) = 2$ and $w(d) = 3$. The greedy solution is $\{c, d\}$ and this has weight 5. The optimal solution is $\{a, b, c\}$ and this has weight 6.

4. Identify two subsets $i, j \subset E$ such that $(E, \mathcal{I} + i + j)$ is a matroid.

   ANSWER: Add the subsets $\{b.d\}$ and $\{d, b, c\}$.

Question 3: Give a linear-time algorithm that takes as input a tree T and determines whether it has a perfect matching, i.e., whether there exists a subset of edges that touches each node exactly once.

ANSWER: Let V and E be the vertices and edges of T and let $n = |V|$. Consider the following algorithm:

1. $M \leftarrow \emptyset$
2. *While* $E \neq \emptyset$:
   (a) *Let $v \in V$ be a leaf (i.e., a node of degree 1) and let $(u, v) \in E$ be the unique edge incident on v*
   (b) $M \leftarrow M + (u, v)$
   (c) $V \leftarrow V - u - v$ *and remove all edges from E that are incident on u or v*
3. *If $|M| = n/2$ return "perfect matching" and if not, return "not perfect matching"*

We first note that M is always a matching because whenever an edge e is added to M, all the edges sharing an endpoint with e are removed from the graph and therefore cannot subsequently be added to M. Hence if $|M| = n/2$, then we indeed have a perfect matching. Consequently we never return "perfect matching" if T does not have a perfect matching.

It remains to show that if T has a perfect matching then we return "perfect matching". Assume M^* is such a matching. If v is a leaf and $(u, v) \in E$ then (u, v) must be in M^* since otherwise v will not be covered. Hence, without loss of generality we may add (u, v) to M and remove all other edges incident on u. After removing u and v from V, the resulting subgraph has a perfect matching and we may recurse on this graph. In this way we find all the edges of M^* and hence $|M| = |M^*| = n/2$.

Question 4: A subsequence is palindromic if it is the same whether read left to right or right to left. For instance, the sequence

$$A, C, G, T, G, T, C, A, A, A, A, T, C, G$$

has many palindromic subsequences, including A, C, G, C, A and A, A, A, A. Devise an algorithm that takes a sequence $x_1, x_2, \ldots, x_n$ and returns the length of the longest palindromic subsequence. For full marks the running time should be $O(n^2)$ but partial marks are available for less efficient solutions.

Hint: Let $A[i,k]$ be the length of the longest palindromic subsequence of $x_i, x_{i+1}, \ldots x_{i+k-1}$ and consider computing $A[1,n]$ via dynamic programming.

ANSWER: Since a subsequence of length 1 is palindromic, $A[i,1] = 1$ *for all* i. *For* $i \in [n]$ *and* $k \in \{2, \ldots, n-i+1\}$

$$A[i,k] = \begin{cases} \max(A[i,k-1], A[i+1,k-1]) & \text{if } x_i \neq x_{i+k-1} \\ 2 + A[i+1,k-2] & \text{if } x_i = x_{i+k-1} \end{cases}$$

The rationale for the formula is that the longest palindromic subsequence of $x_i, x_{i+1}, \ldots, x_{i+k-1}$ *either includes* x_i *and* x_{i+k-1} *(in which case they must be equal) or it doesn't. If it doesn't the longest palindromic subsequence of* $x_i, x_{i+1}, \ldots, x_{i+k-1}$ *is also a palindromic subsequence of* $x_{i+1}, \ldots, x_{i+k-1}$ *or* $x_i, \ldots, x_{i+k-2}$. *It it does then* $A[i,k]$ *is* $2 + A[i+1,k-2]$.

Hence $A[1,n]$ *can be constructed by the following algorithm:*

1. *For all* $i \in [n]$, *let* $A[i,1] = 1$
2. *For all* $i \in [n]$, *let* $A[i,2] = 2$ *if* $x_i = x_{i+1}$ *and* $A[i,2] = 1$ *otherwise*
3. *For* $k = 3$ *to* n*:*
   (a) *For* $i = 1$ *to* $n - k + 1$*:*

$$A[i,k] = \begin{cases} \max(A[i,k-1], A[i+1,k-1]) & \text{if } x_i \neq x_j \\ 2 + A[i+1,k-2] & \text{if } x_i = x_j \end{cases}$$

4. *Return* $A[1,n]$

The running time is clearly $O(n^2)$ *since there are* $O(n^2)$ *values to be computed and each requires* $O(1)$ *time.*

Question 5: Given a **sorted** list of **distinct integers** $A[1], A[2], \ldots, A[n]$, you want to find out whether there is an index i for which $A[i] = i$. Give an algorithm that runs in time $O(\log n)$. Remember to prove that your algorithm is correct and analyze the running time.

For example, $A = [-1, 1, 3, 5, 6, 7]$ does have such an index, i.e., $A[3] = 3$. But $A = [0, 1, 2, 5, 6, 7]$ doesn't have such an index.

ANSWER: Consider the following algorithm.

1. $a \leftarrow 1$
2. $b \leftarrow n$
3. *While* $b - a + 1 \geq 3$
   (a) $k \leftarrow \lceil (a + b)/2 \rceil$
   (b) *If* $A[k] = k$ *then return YES!*
   (c) *If* $A[k] < k$ *then* $a \leftarrow k + 1$
   (d) *If* $A[k] > k$ *then* $b \leftarrow k - 1$
4. *If* $A[a] = a, A[a + 1] = a + 1,$ *or* $A[a + 2] = a + 2$ *then return YES! Otherwise NO!*

The running time is $O(\log n)$. This follows because the value of $b - a + 1$ decreases by a factor 2 in each iteration. Let a_i and b_i be the values of a and b at the start of the i-th iteration. Then

$$b_{i+1} - a_{i+1} + 1 \leq \max(b_i - k, k - a_i) \leq (b_i - a_i + 1)/2$$

Hence, there are at most $O(\log n)$ iterations. Because each iteration takes $O(1)$ time, the total running time is as claimed.

We now prove correctness. Note that we can never report YES when the answer is NO because we only answer YES when a test $A[i] = i$ is passed for some i. It remains to prove that we never answer NO when the answer should be YES. To do this we prove that if there exists $i \in [n]$ such that $A[i] = i$, then at any point of the algorithm the values of a and b satisfy $a \leq i \leq b$. We show this by induction on the number of times the while loop is performed. Before the loop is performed, $a = 1$ and $b = n$ and so the base case is trivially true. Assume that the claim is true before the jth time the loop is performed. If $A[k] < k$ then we know all $\ell \leq k$ satisfy $A[\ell] \leq A[k] - (k - \ell) < \ell$ and $i \geq k + 1$ as required. Similarly, if $A[k] > k$ then we know all $\ell \geq k$ satisfy $A[\ell] \geq A[k] + (\ell - k) > \ell$ and hence $i \leq k - 1$ as required. Therefore, either we find the fixed point during some iteration or in the fourth line. Either way, we output YES.