Algorithms II 89-322-01, 89-322-02, FINAL EXAM
MOED A

Instructor: *Prof. Amihood Amir*
Length of Exam: 2 hours
Time: July 7th, 08:30
NO OUTSIDE MATERIAL ALLOWED!!!

A general note about the grading process: Below you will find how many points were deducted for each error. Some answers incurred several errors, and so the amount of points deducted in such case was usually the sum of points deducted per each error, although some exceptions were made (deducting less points). In addition, some notebooks were given besides the error a number of pluses (+) or minuses (-). This indicate either additional points that were added although some errors, or additional points being deducted due to some other issue which we did not find the need to list here. There were several cases where an answer was wrong, but we felt that some points were due, and so we gave anywhere between 5 to 10 points.
Another important note. You were notified that there will be a question on the exam from last year's assignment. This was question 3[1]. Question 1 was done in the tutorial. Question 2 was mentioned in class and the instructor suggested you make sure you know how to answer all questions left as "exercises". Thus the low grades are mystifying.

1. (33 points) An undirected graph in input online edge by edge. Is there a competitive algorithm that finds the vertex cover of the graph? Describe your algorithm's idea and prove its time complexity. The better the competitive ratio (for a correct one) the more points given.

   Answer:

   The idea is to use the approximation algorithm for vertex cover.

   Algorithm: Set $V \leftarrow \emptyset$.
   Get next edge $\overline{ab}$.
   if $a, b, \notin V$ then add a and b to V.
   end Algorithm

   Time: Checking whether the nodes are in V takes constant time,

[1] There was an additional trick needed to receive full credit, but only 10 points were deducted from answers who did not deal with it properly.

so $O(1)$ time per edge. This follows from the assumption that was conveyed during the exam that all of the nodes are known in advance.

competitive Ratio: Via the same argument we have shown in class for the vertex cover approximation, the competitive ratio is 2. For each edge the optimal solution must use at least one of its vertices, while our algorithm uses both nodes.

Error Codes and Penalties:

1. An algorithm running in $\omega(1)$ time per edge. -5.
2. An algorithm which uses adds either a or b to V, but not both, or some other similar incorrect complication. -13.
3. A proof was given based on the "worst case" or on an example - this is not mathematically valid!! -10.
4. An algorithm whose decision is based on the degree of the node (this is similar to error 2). -13.
5. An algorithm which is allow to "regret" previous decisions, and remove nodes from V. Such an algorithm does not comply with the on-line model. -20.
6. An algorithm using Linear Programming. This is a very inefficient solution. -18.
7. An algorithm that for any edge always adds both a and b. A good example for a bad input for this algorithm is a star graph. -18.
8. No proof of competitiveness. -10.

2. (33 points) The following recursive algorithm is suggested for finding a maximum agreement homeomorphic subtree of k trees $T_1, ..., T_k$:

   (a) Compute a maximum agreement homeomorphic subtree of T_1 and T_2, call it M.

   (b) Compute a maximum agreement homeomorphic subtree of $M, T_3, ..., T_k$.

   Prove that the above algorithm is incorrect.

   Answer:

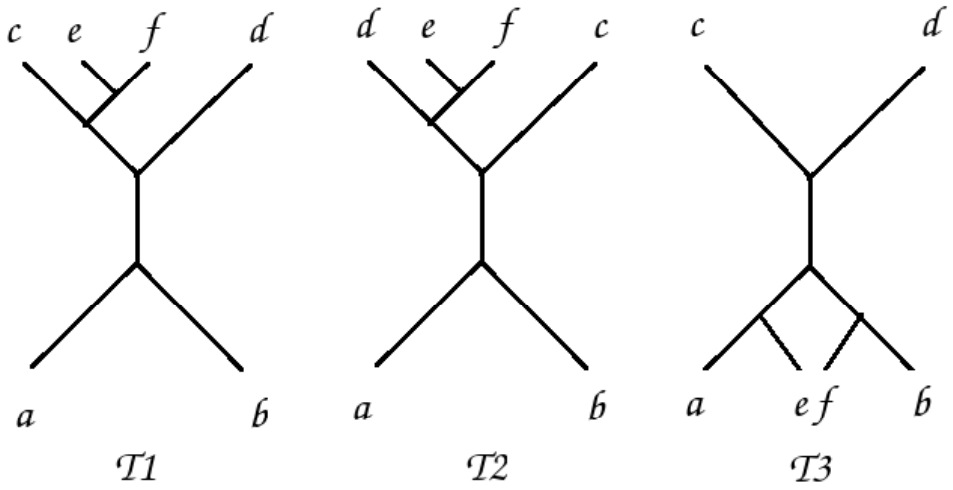

Figure 1: A counter example

Consider the trees in Figure 1:

Note that, for example $MHT(T_1, T_2) = M = \{a, b, c, e, f\}$ and then $MHT(M, T_3) = \{a, b, c\}$, but $MHT(T_1, T_2, T_3) = \{a, b, c, d\}$.

Error Codes and Penalties:

1. An example which uses trees with different leaves, or trees of different sizes. -8.
2. No example - only a general explanation for why the algorithm "could be" incorrect. -25.
3. One of the trees is just one node, or degenerated. Similar to error 1. -8.
4. Missing the optimal tree. -3.

3. (34 points) Write a Linear Program for the following problem:
Input: Undirected graph $G = (V, E)$.
Decide: whether graph G is bipartite.

Write an LP in standard form that has a solution iff G is bipartite.

Answer:

We start with an Integer Program in the general form, then we'll convert it to the LP in the standard form.

Variables: For every node $v \in V$ define a variable x_v whose value is either 0, if v is on the "left" or 1 if v is on the "right" part.

Objective Function: Note that there is no need for an objective function here, so anything goes, e.g. *maximize* x_a, for some $a \in V$.

Constraints: For every edge $\overline{ab} \in E$ write the constraint: $x_a + x_b = 1$.

Clearly, there is a solution iff the graph is bipartite. This is because if the graph is bipartite by splitting the nodes to two disjoint sets A and B, then setting the variable of all of the nodes in A to 1, and the variables of all of the nodes in B to 0 satisfies all of the equalities (each edge has one node in A and one node in B). On the other hand, if there is a satisfying assignment to the variables then there cannot exist a cycle of odd length (make sure you understand why).

Therefore, a standard form IP would be:

maximize x_a under the constraints:
For every edge $\overline{ab} \in E$:

$x_a + x_b =\leq 1$

$-x_a - x_b =\leq -1$

and $x_v \geq 0, \quad \forall v \in V$.

We need to turn this into an LP. The real problem is that it it is possible to assign $\frac{1}{2}$ to all of the variables, regardless of whether the graph is bipartite or not. In fact, this is the only "trick" in the entire exam. There are many ways of doing it. One of the simplest is to set one variable (for each component, if the graph is not connected) to 1. for example, add

$x_a \leq 1$

$-x_a \leq -1$

This forces all variables to be either 0 or 1.

Error Codes and Penalties:

1. Not in standard form. -10.
2. Did not deal properly with preventing assignments of $\frac{1}{2}$. -5
3. Providing a LP which is based on prior knowledge of the partition of the bipartite graph. -25.

4. Did not deal properly with preventing assignments of $\frac{1}{2}$ (like error 2). -5
5. LP without an IP -10.
6. An LP which is always incorrect (but some intuition is understood). -19.
7. No objective function. -2.
8. No constrains on the variables. -2.
9. Missing a direction of the correctness proof. -10.
10. Bad relaxation. -19.
11. Like Error 3.
12. No proof. -10.

4. (5 points) How does a *Ben Yeshiva* eat soup with a spoon?

   Answer: Using the law of *Lavud*.

GOOD LUCK