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

Instructor: *Prof. Amihood Amir*
Length of Exam: 2 hours
Time: Sept. 10, 2013, 12:00
NO OUTSIDE MATERIAL ALLOWED!!!

1. Prove that the KMP automaton construction can be done in time $O(m)$, where m is the pattern length.

   Answer:

   Let me start by saying that this problem was assigned in the class so I assumed everyone did it and that this is a "gift" question. Consequently, it was also graded strictly. The proof below uses a counter, as was shown in class. It is also possible to use the potential function method.

   Proof: Assign a counter F that is incremented by 1 when a forward arrow is added and decremented by 1 when a fail arrow is followed. Denote by $F(i)$ the value of F at state i. Denote by $fail(i)$ the target state of the failure arrow of state i.

   Claim: For every state i, $F(i) > fail(i)$.

   Proof: For $i = 1$, the fail arrow goes to state 0 and $F(1) = 1 > 0 = fail(1)$.

   Assume for state i, $F(i) > fail(i)$. Consider state $i+1$. Let $fail(i) = j$. By the induction hypothesis $F(i) > j$. Since $F(i+1) = F(i) + 1$ we get $F(i+1) > j + 1$.

   If the success arrow from j has the same symbol as the success arrow from i then we are done since $fail(i+1) = j+1$. Otherwise, assume that that $fail(i+1) = s+1$, where s is the state we arrive at by following repeated fail arrows. The route from j to s involves at most $j - s$ fail arrows, thus $F(i+1) - (j-s) > j + 1 - (j-s) = s + 1 = fail(i+1)$. ∎

   Now, because of the claim F is never negative. Since the number of additions to F is at most n then the number of failure arrows followed is also at most n.

   Errors:

   1. Described algorithm and (a) claimed that $O(m)$ arrows implies $O(m)$ construction time, or (b) claimed that every fail arrow is traversed only once during automaton construction, or (c) no adequate explanation of why the time is not $O(m^2)$. **5 points**.

   2. The idea in words but no formal proof. **16 points**.

   3. Constructed a credit function but no explanation of when to reduce, no proof. **21 points**.

   4. Description of a new $O(m^2)$ algorithm. **0 points**.

   5. Correct credit function. No proof or incorrect proof. **25 points**.

   6. A glimmer a description of the idea. **13 points**.

   7. Does not show an automaton construction **0 points**.

   8. Correct credit function but construction not exact and no proof. **21 points**.

   9. Did not answer question, showed no understanding. **0 points**.

10. Problems in proof of counter.**30 points**.

11. Uses potential function method but does not define the function. **21 points**.

12. Uses potential function method but problem with function. **25 points**.

13. Assumption that $|fail(s_1) - fail(s_2)|$ equals the number of failure arrows traversed to get from s_1 to s_2. **25 points**.

14. Described and proved algorithm correctness but not running time. **10 points**.

15. Problem with induction proof on counter. **28 points**.

16. Describes new algorithm that constructs automaton based on witness table. No correctness proof. **5 points**.

17. Incorrect algorithm. **0 points**.

18. Stack algorithm. **0 points**.

19. Describes new algorithm that constructs automaton based on suffix trees and LCS. No correctness proof. **5 points**.

20. Credit function not described well. Proof incomplete. **21 points**.

2. The **Hamiltonian Cycles** problem is defined as follows.
INPUT: An undirected graph $G = (V = \{1, ..., n\}, E)$.
DECIDE: If all nodes of the graph are spanned by a set of non-intersecting cycles, i.e., if there are simple paths $v_1^1, ..., v_{n_1}^1, v_1^1;\ \ ...\ \ ; v_1^\ell, ..., v_{n_\ell}^\ell, v_1^1$ where
$\underline{v_i^k \neq v_j^m}$, unless $i = j$ and $k = m$;
$\underline{v_i^j v_{i+1}^j} \in E, \quad i = 1, ..., n-1,\ j = 1, ..., \ell$;
$\underline{v_n^j v_1^j} \in E,\ j = 1, ..., \ell$;
and $\sum_{i=1}^{\ell} n_i = n$.

   Write an Integer Program that can solve the Hamiltonian Cycles problem.

   Answer:

   The key observation here is that there exist Hamiltonian cycles iff every node has exactly two edges participating in the cycles (since any node that has other than two edges clearly can not be part of a simple cycle). We also note that the value of the objective function is not meaningful as long as the domain is not empty, since we are considering a decision problem and not an optimization problem.

   We therefore assign a variable E_i to every edge e_i, These variables get integer value from $\{0, 1\}$. Assume there are n edges.

   The objective function is $\min \sum_{i=1}^{n} E_i$.

   Subject to the constraints:

   $E_i \in \{0, 1\}$, for $i = 1, ..., n$.

   For every node v whose adjacency list is $e_{i1}, ...e_{in_v}$:

$$\sum_{j=1}^{n_v} E_{ij} = 2.$$

Errors:

Note that here are given the points *deducted* for each mistake since there were people who had multiple mistakes.

1. No objective function given . **-10 points**.
2. No constraints given. **-30 points**.
3. No mention of which variables are integers. **-5 points**.
4. Acceptance criterion not provided. **-10 points**.
5. Constraints or objective function not linear. **-15 points**.
6. Illegal objective function. **-10 points**.
7. No ranges given for variables. **-10 points**.
8. Not all variables defined. **-10 points**.
9. Considers m as a given constant. **-10 points**.
10. Assumed that the index of the nodes in the graph is identical to its index in the cycle. **-10 points**.
11. Given constraints decide clique rather than cycle. **-10 points**.
12. Correct idea but implementation fails since variables chosen for vertices rather than edges. **-5 points**.
13. Did not understand problem definition. **-28 points**.
14. No constraint forcing every node to be in at most one cycle. **-10 points**.

3. The **2BALANCE** strategy for the *two server problem* is defined as follows: Let t_i be the distance that server i has travelled so far, and let d_i be the distance from server i to the event, for $i = 1, 2$.

   Move the server i where $2t_i + d_i$ is *smallest*.

   Prove that the 2BALANCE strategy is not competitive.

   Answer:

   Assume 2BALANCE is c-competitive. Consider the following scenario: Let $d = (c + 2)^2$. Let points A, B, C, D be plane locations $(0, 0), (0, 1), (d.0), and(d, 1)$ respectively, and let one server be on point A and one be on point B at initialization. The sequence of events is (C, D, A, B) repeated d times.

   The work of the 2BALANCE algorithm is $4d^2 = 4(c + 2)4 = 4c^4 + 32c^3 + 96c^2 + 128c + 64$.

   The optimal algorithm's work is $4d + d - 1 = 4c^2 + 17c + 17$.

   Clearly, $4c^4 + 32c^3 + 96c^2 + 128c + 64 > 4c^3 + 17c^2 + 17c$, which contradicts the assumption of c-competitiveness. ∎

   Errors:

   Note that here are given the points *deducted* for each mistake since there were people who had multiple mistakes.

   1. No explicit example of numbers contradicting the competitive value. **-4 points**.

2. Paid distance 2 instead of 4 on every quadruple. **-2 points**.

3. Incomplete phrasing of the claim. **-5 points**.

4. Starting points of servers not provided. **-4 points**.

5. Ignored the fact mentioned in class that we always assume that a server **on** the event takes care of the event. **-10 points**.

6. Example not rigorously explained. **-10 points**.

7. Example not on plane – no triangle inequality. **-20 points**.

8. Chooses the competitive ratio as a function of the distance and length of sequence, rather than vice versa. **-5 points**.

9. Error in distance calculation. **-2 points**.

10. Gives a competitive example claiming it is not competitive. **-30 points**.

11. Example not on natural grid. **-4 points**.

12. Undefined parameters. **-2 points**.

13. Did not calculate distance formulae. **-5 points**.

14. Incomplete proof of not competitive ratio. **-4 points**.

GOOD LUCK