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

Instructor: *Prof. Amihood Amir*
Length of Exam: 2 hours
Time: September 11th, 2013, 08:30
NO OUTSIDE MATERIAL ALLOWED!!!

1. **Definition:** Let S be a string over alphabet Σ. S' is the *smallest period* of S if S' is the smallest string such that $S = S'^k S''$, where S'' is a prefix of S. If no such S' exists that S is *not periodic*.

   Example: $S = abcabcabcabca$. $abcabc$ is a period since $S = abcabc\ abcabc\ a$, but the smallest period is abc since $S = abc\ abc\ abc\ abc\ a$.

   Give an algorithm to find the smallest period of input string S or declare that S is not periodic. Explain your idea and analyse the time of your algorithm.

   (The more efficient the algorithm, the higher the points given.)

   ANSWER: There are many ways to answer this question. The easiest and fastest is by noting that if we delete the smallest period S' from the start of the string S, then for the remaining string S'' we have:
   $S''[i] = S[i], \quad i = 1, ..., |S''|$.

   But this means precisely that S'' is the longest prefix of S that is also a suffix. Thus the fail link of the accepting node of the KMP automaton points to state i, where $n - i$ is the length of the smallest period.

   Error Codes and their Penalties:

   1. Correct naive $O(n^2)$ algorithm. (16)
   2. Correct $O(n \log n)$ algorithm. (-4)
   3. Not used.
   4. Correct idea but wrong algorithm. (-12)
   5. Correct $O(n^3)$ algorithm. (5)
   6. Definition not understood. (-33)
   7. Errors 7. and 8. are two sides of the same coin. They are answers of the following type: Make a claim about how a suffix tree of a periodic string ought to look; Prove it; Write an algorithm for checking this condition. The overall points distribution was: 9 points for the idea, 11 points for the proof and 9 points for the algorithm. Recall that because of error code 2, 4 points are automatically deducted because the time complexity of using a suffix tree is not optimal for unbounded alphabets.
      a. If the claim is: Every edge in the suffix tree is a \$ or P\$, where P is a cyclic permutation of the period, but cyclic permutation is not mentioned. (-2)
      b. No proof given. (-11)
   8. Again there are subcases:

a. Incorrect condition. (-9)
b. No proof given. (-11)
c. Partial proof. (-5)
d. Mistake in algorithm complexity analysis. (-4)

9. Used Idea of KMP or witness table, but then technical error in algorithm. (-11)
10. Technical error in algorithm of suffix tree and LCA. (-13)
11. Has a quadratic algorithm but claims the algorithm is linear. (-20)

2. The **3-CNF** problem is defined as follows.
INPUT: A boolean formula in CNF (Conjunctive Normal Form) and no more than 3 literals in avery conjunct, i.e. $\alpha = (\alpha_{1,1} \vee \alpha_{1,2} \vee \alpha_{1,3}) \wedge (\alpha_{2,1} \vee \alpha_{2,2} \vee \alpha_{2,3}) \wedge \cdots \wedge (\alpha_{k,1} \vee \alpha_{k,2} \vee \alpha_{k,3})$, where $\alpha_{i,j}$ are atomic formulae or their negations.
DECIDE: If there is an assignment of truth values to the atomic formulae that satisfies α.

Example: $\alpha = (A \vee \sim B \vee C) \wedge (\sim D \vee \sim B \vee A) \wedge (E \vee A \vee \sim C) \wedge (A \vee D \vee E) \wedge (\sim E \vee \sim B \vee \sim D)$

Write an Integer Program that can solve the 3-CNF problem.

ANSWER: Let us assume that there are m different atomic formulae. We define m boolean variables $X_1, ..., X_m$ and assign a Boolean variable for each of the atomic variables. A "1" means that the corresponding atomic formula gets truth value T and a "0" means that the corresponding formula gets value F. We need to make sure that each conjunct has at least one satisfied atomic formula, i.e. one boolean variable that gets value 1. We also need to make sure we take care of the negations. Note that this is not an optimization function so we want to make sure that something about the optimization function we construct can help us decide the 3-CNF problem.

The resulting IP is:

$\max \sum_{i=1}^{k} X_i$

subject to the following constraints:

For every conjunct $(A \vee B \vee C)$, let X_i, X_j, X_ℓ be the Boolean variables corresponding to atomic formulae A, B, C, then $X_i + X_j + X_\ell \geq 1$.

If some of the disjuncts in the conjunct are negations, then the constraint handles the negation in the following manner: if X_i is the Boolean variable corresponding to A and $\sim A$ occurs as a disjunct, then $1 - X_i$ will be put in the constraint (as opposed to just X_i, if there is no negation.

The final constraints are: $X_i \in \{0, 1\}, \quad i = 1, ..., m$.

NOTE: An interger program may have some variables that get integer value and some that get real values. Thus it is important to be explicit about which variables are integer and which are real.

NOTE: There is another way of handling negations. Define variables $X_1, ..., X_m$ for the Boolean atomic formulas, as before, but also variables $\overline{X_1}, ..., \overline{X_m}$ to be used wherever a negation of a boolean formula appears. So, $(A \vee B \vee \sim C)$, for example, would generate the constraint $X_i + X_j + \overline{X_\ell} \geq 1$.

Of course we then need the additional constraints:

$X_i + \overline{X_i} = 1, \quad i = 1, ..., m$, and
$X_i, \overline{X_i} \in \{0, 1\}, \quad i = 1, ..., m.$

Finally, we need to specify that if the domain is empty, i.e. there are no values for which the constraints then we do not accept, otherwise we accept.

Error Codes and their Penalties:

1. no objective function given (-5)
2. Assumes that the variable values are in $\{0, 1\}$ but that $-1 = 0$ (-10)
3. defined variables $-x$ (-5)
4. no constraints given (-30)
5. not mentioning which variable s are integers (-5)
6. negation not handled (-10)
7. different independent variables for A and $\sim A$ (-10)
8. no acceptance criteria (-10)
9. an IP that always gives the number of conjuncts (-35)
10. The constraints or objective function not linear functions (-15)
11. No ranges given for the variables. (-10)
12. Defined the variables for the literals rather than atomic variables. However, example showed that you meant that variables are defined for atomic variables. (-5)
13. Illegal objective function. (-10)
14. Accepts everything. (-25)

3. Consider the following claim:
Claim: Let σ be a segment of the input for the caching problem, where LRU experienced k faults. Let P be a page that is not in the cache when segment σ start but p is evicted within segment σ. Then even the optimal strategy will have at least one fault during execution of segment σ.

Prove the above claim or give a counter-example.

ANSWER: The claim is correct.
Proof: P is evicted from the segment, yet was not there at the start of the segment. This means that when P is evicted, all $k - 1$ other pages in the cache were accessed *after* P was last accessed. P was last accessed within segment σ. Therfore, within segment σ we have accessed k different pages, and we also accessed the page that caused P's eviction (which is different from all pages in the cache at that moment). Consequently, $k + 1$ different pages have been accessed in segment σ. No strategy can handle $k + 1$ pages in a cache of size k without at least one fault.

Error Codes and their Penalties:

1. Error in proof that k+1 pages were accessed. (-15)

2. Did not understand that you need to count the number of different pages accessed. (-20)
3. Claims that if there is a fault at segment σ it means that the number of different pages in σ is larger than the cache size. (-25)
4. Claims that LRU is FIFO. (-25)
5. Assumed that P is not in the optimal cache at the beginning of σ. (-29)
6. Proved claim for the special case of $k = 1$. (-29)
7. Proved claim for the special case of $k = 3$. (-27)
8. Error in counting $k + 1$ different page access. (-5)
9. Proved claim for the special case of a periodic segment. (-29)
10. Claims that if there are k faults in a segment then the segment is of length at least $2k$. (-29)

GOOD LUCK