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

Instructor: *Prof. Amihood Amir*
Length of Exam: 2 hours
Time: September 8th, 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).

1. (33 points) Describe an algorithm that constructs a one-dimensional witness table using suffix trees and LCA queries. What is the time complexity of your algorithm? Justify.

Answer:

Let $P[1]P[2]\cdots P[m]$ be a pattern string. Denote by $P[i..j]$ the substring $P[i]P[i+1]\cdots P[j]$. Denote by k_i the longest common prefix of $P[1..m]$ and $P[i..m]$.

By definition, $W[i] = *$ if $k_i \geq m - i + 1$. Otherwise $W[i] = k$, where $P[k] \neq P[k - i + 1]$ (i.e. the pattern and its "shift" to location i have a different symbol in that location). In particular, one may give $W[i]$ the value $i + k_i$.

Now recall that constructing a suffix tree for P, preprocessing it for LCA, and, via e.g. DFS, writing on every suffix tree node the length of the string from the root to that node, will give us the following: For every LCA query, the length of the result is the longest common suffix of the two substrings. In particular, $\ell(LCA(P[i..m], P[1..m])) = k_i$.

Time: Suffix tree construction takes time $O(m \log \sigma)$, where $\sigma = \min(|\Sigma|, m)$, where $|\Sigma|$ is the alphabet size. LCA preprocessing and DFS for length notation can be done in time $O(m)$. Subsequent queries are constant time. Thus the total time for constructing the witness table in this manner is $O(m \log \sigma)$.

Error Codes and Penalties:

1. UNUSED.
2. No algorithm provided. -20.
3. No explanation for algorithm provided. -10.
4. No suffix tree complexity mentioned. -3
5. No LCA complexity mentioned. -1.
6. No DFS complexity mentioned. -1.
7. Lack of details on suffix tree construction part of algorithm. -6.
8. Lack of details on LCA construction part of algorithm. -6.
9. Lack of details on length calculation part of algorithm. -6.

10. Lack of explanation on why the answer is the desired one. -5.
11. Infinite alphabet not handled. -2.
12. Incorrect or missing calculation of witness table value. -3.
13. Using uncompacted suffix trie. -17.
14. Suffix tree construction in quadratic time. -17.
15. Naive quadratic calculation of suffix tree. -18.

2. (33 points) The **Two Dimensional Pattern Matching with Mismatches Problem** is defined as follows:
INPUT: Test T of size $n \times n$, pattern P of size $m \times m$, and natural number k.
Find: All locations i, j in the text where the pattern occurs with at most k errors (an error is a mismatch of the text symbol with its appropriate pattern symbol).

   Write an algorithm that solves the Two Dimensional Pattern Matching with Mismatches Problem. Describe the algorithm's idea and prove its time complexity. The faster the algorithm (if correct) the more points given.

   Answer:

   We have seen two algorithms for the one dimensional pattern matching with mismatches problem. The Kangaroo method, whose time is $O(n \log \sigma + nk)$ and Abrahamson's convolutions method, whose time is $O(n\sqrt{m \log m})$. The best algorithm has time $O(n\sqrt{k \log k})$ but we have not mentioned it in class so you were not expected to generalize it (those interested in that algorithm, register to the Pattern Matching Algorithms course this year).

   The convolutions method linearizes the text by concatenating all its rows, and linearizes the pattern by concatenating its rows, padded with $n - m$ "don't cares". Then use Abrahamson. This will give time $O(n^2\sqrt{mn \log n})$. However, if we separate the text into overlapping squares of size $2m \times 2m$, the time becomes $O(n^2 m\sqrt{\log m})$.

   The Kangaroo method, would start at every text location and jump to the first error, then the second, etc. However, since we now have m rows, the time is $O(k + m)$ per location, for a total of $O(n^2(k + m))$. If $k < m$ this is a problem. The problem can be solved as follows: First do Aho and Corasick to find all exact matches of pattern rows. The time is $O(n^2 \log \sigma)$. Then for every text location, do the kangaroo jumps only on the rows where there is no exact match (assuming no more than k such rows). The rows where there is no exact match can be found by one-dimensional kangaroo jumps on the "row names" pattern. Now the time is $O(k)$ per text location for a total of $O(n^2 \log \sigma + n^2 k)$.

   Error Codes and Penalties:

   1. "Two dimensional convolution" was not discussed in class thus can not be used. -20/
   2. Infinite alphabet was not considered. -3.
   3. Showed only the $O(n^2(m + k))$ kangaroo method. -5.
   4. Showed only the $O(n^2\sqrt{mn \log n})$ convolutions method, -8.

5. Failed to recognize the fact that when padding with "don't cares" the pattern row length becomes n. -5.
6. Considers an "average" of k/m mismatches per row. We could have all errors in one row and all other rows may be fine. -8.
7. Kangaroo row-by-row for $O(n^2 mk \log \sigma)$. -10.
8. Naive $O(n^2 m^2)$ algorithm. -15.
9. Showed only the $O(n^2 m\sqrt{\log m})$ convolutions method. -5.
10. Showed only the $O(n^2 m\sqrt{\log m})$ convolutions method, but failed to pad with "don't cares". -8.
11. Abrahamson row-by-row in text times row-by-row in pattern. -10.
12. When writing the time complexity, forgot that we are dealing with two-dimensions, i.e. n^2 and m^2 and wrote complexity in terms of n and m. -5.
13. Row-by-row Kangaroo in text but failed to notice that in pattern also needs to do row-by-row. -10.
14. Kangaroo method without handling the case of $k < m$. -5.
15. Used witness table method even though no transitivity. -30.
16. No complexity analysis. -10.
17. Showed only the $O(n^2 m\sqrt{\log m})$ convolutions method, but failed to explain how to pad with "don't cares". -6.
18. Slightly wrong complexity analysis. -5.
19. UNUSED.
20. Row-by-row convolutions. -10.
21. Used automata method even though no transitivity. -30.
22. Gave the one dimensional algorithm. -30.

3. (34 points) Solve the following LP. Write what is the optimal value of the objective function and what are the variable values in the optimum. Justify your answer.

Objective function: $\max x + y - 2z$.

Constraints:
$x + y \geq 1$
$x| + 2y \leq 3$
$x \geq 0$
$y \geq 0$
$z \geq 0$

Answer:

Observe that the only constraint on variable z is that it is non-negative. However, it is subtracted from the objective function, thus the maximum value of the objective function will be the smallest value of z, i.e. 0.

Now it is possible to compute the domain of x and y and then calculate the maximum of the objective function. It is also possible to do this directly by the following observation:

We want to maximize $x + y$. But we are constrained by $x + 2y \leq 3$, i.e. $x \leq 3 - 2y$. But this means that $x + y \leq 3 - 2y + y = 3 - y$.

Again, y is non-negative so to maximize $x + y$ we need to take the smallest y possible, i.e. $y = 0$. This means that the largest x possible is $x = 3$.

the answer is: Fro values $x = 3, y = 0, z = 0$, the objective function is maximized and its value is 3. maximize

Error Codes and Penalties:

1. Failed to justify why $z = 0$. -7.
2. Failure or error in calculating the domain. -7.
3. No calculation of maximum value of objective function or error in that calculation. -7.
4. Computing solution to IP. -7.
5. Maximum objective function calculation not crisp. -5.
6. Wrong or no solution. -13.
7. No value provided for objective function or error. -3.

GOOD LUCK