CS 373: Combinatorial Algorithms, Fall 2000

Homework 4 (due October 26, 2000 at midnight)

Name:		
Net ID:	Alias:	U 3/4 1

Name:		
Net ID:	Alias:	U 3/4 1

Name:		
Net ID:	Alias:	U 3/4 1

Homeworks may be done in teams of up to three people. Each team turns in just one solution, and every memeber of a team gets the same grad. Since 1-unit graduate students are required to solve problems that are worth extra credit for other students, **1-unit grad students may not be on the same team as 3/4-unit grad students or undergraduates.**

Neatly print your name(s), NetID(s), and the alias(es) you used for Homework 0 in the boxes above. Please also tell us whether you are an undergraduate, 3/4-unit grad student, or 1-unit grad student by circling U, 3/4, or 1, respectively. Staple this sheet to the top of your homework.

RequiredProblems

1. (10 points) A certain algorithms professor once claimed that the height of an n-node Fibonacci heap is of height $O(\log n)$. Disprove his claim by showing that for a positive integer n, a sequence of Fibonacci heap operations that creates a Fibonacci heap consisting of just one tree that is a (downward) linear chain of n nodes.

2. (20 points) *Fibonacci strings* are defined as follows:

$$
\begin{aligned}
F_1 &= \mathsf{b} \\
F_2 &= \mathsf{a} \\
F_n &= F_{n-1}F_{n-2} \quad \text{for all } n > 2
\end{aligned}
$$

where the recursive rule uses concatenation of strings, so $F_3 = \mathsf{ab}$, $F_4 = \mathsf{aba}$, and so on. Note that the length of F_n is the nth Fibonacci number.

(a) Prove that in any Fibonacci string there are no two b's adjacent and no three a's.

(b) Give the unoptimized and optimized failure function for F_7.

(c) Prove that, in searching for the Fibonacci string F_k, the unoptimized KMP algorithm may shift $\lceil k/2 \rceil$ times on the same text character. In other words, prove that there is a chain of failure links $j \to fail[j] \to fail[fail[j]] \to \ldots$ of length $\lceil k/2 \rceil$, and find an example text T that would cause KMP to traverse this entire chain on the same position in the text.

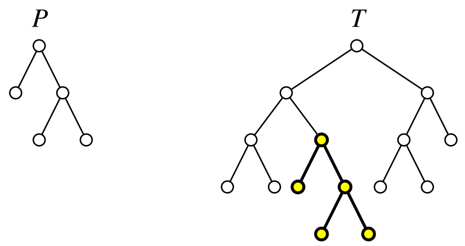
(d) What happens here when you use the optimized prefix function? Explain.

3. (10 points) Show how to extend the Rabin-Karp fingerprinting method to handle the problem of looking for a given $m \times m$ pattern in an $n \times n$ array of characters. The pattern may be shifted horizontally and vertically, but it may not be rotated.

4. (10 points)

   (a) A *cyclic rotation* of a string is obtained by chopping off a prefix and gluing it at the end of the string. For example, ALGORITHM is a cyclic shift of RITHMALGO. Describe and analyze an algorithm that determines whether one string $P[1..m]$ is a cyclic rotation of another string $T[1..n]$.

   (b) Describe and analyze an algorithm that decides, given any two binary trees P and T, whether P equals a subtree of T. We want an algorithm that compares the *shapes* of the trees. There is no data stored in the nodes, just pointers to the left and right children. *[Hint: First transform both trees into strings.]*



P occurs exactly once as a subtree of T.

5. (10 points) *[This problem is required only for graduate students taking CS 373 for a full unit; anyone else can submit a solution for extra credit.]*

   Refer to the notes for lecture 11 for this problem. The GenericSSSP algorithm described in class can be implemented using a stack for the 'bag'. Prove that the resulting algorithm can be forced to perform in $\Omega(2^n)$ relaxation steps. To do this, you need to describe, for any positive integer n, a specific weighted directed n-vertex graph that forces this exponential behavior. The easiest way to describe such a family of graphs is using an *algorithm*!

Practice Problems

1. String matching with wild-cards
   Suppose you have an alphabet for patterns that includes a 'gap' or wild-card character; any length string of any characters can match this additional character. For example if '*' is the wild-card, then the pattern foo*bar*nad can be found in foofoowangbarnad. Modify the computation of the prefix function to correctly match strings using KMP.

2. Prove that there is no comparison sort whose running time is linear for at least $1/2$ of the $n!$ inputs of length n. What about at least $1/n$? What about at least $1/2^n$?.

3. Prove that $2n - 1$ comparisons are necessary in the worst case to merge two sorted lists containing n elements each.

4. Find asymptotic upper and lower bounds to $\lg(n!)$ without Stirling's approximation (Hint: use integration).

5. Given a sequence of n elements of n/k blocks (k elements per block) all elements in a block are less than those to the right in sequence, show that you cannot have the whole sequence sorted in better than $\Omega(n \lg k)$. Note that the entire sequence would be sorted if each of the n/k blocks were individually sorted in place. Also note that combining the lower bounds for each block is not adequate (that only gives an upper bound).

6. Show how to find the occurrences of pattern P in text T by computing the prefix function of the string PT (the concatenation of P and T).

7. Lower Bounds on Adjacency Matrix Representations of Graphs
   (a) Prove that the time to determine if an undirected graph has a cycle is $\Omega(V^2)$.
   (b) Prove that the time to determine if there is a path between two nodes in an undirected graph is $\Omega(V^2)$.