CS 373: Combinatorial Algorithms, Spring 2001

http://www-courses.cs.uiuc.edu/~cs373

Homework 5 (due Tue. Apr. 17, 2001 at 11:59 pm)

Name:		
Net ID:	Alias:	U $^3\!/_4$ 1

Name:		
Net ID:	Alias:	U $^3\!/_4$ 1

Name:		
Net ID:	Alias:	U $^3\!/_4$ 1

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. Prove that finding the second smallest of n elements takes EXACTLY $n + \lceil \lg n \rceil - 2$ comparisons in the worst case. Prove for both upper and lower bounds. Hint: find the (first) smallest using an elimination tournament.

2. *Fibonacci strings* are defined as follows:

$$F_1 = \text{“b”}, \qquad F_2 = \text{“a”}, \qquad \text{and } F_n = F_{n-1}F_{n-2}, (n > 2)$$

   where the recursive rule uses concatenation of strings, so F_3 is “ab”, F_4 is “aba”. 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 ‘prefix’ (fail) function for F_7.

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

(d) Prove that the unoptimized KMP algorithm can shift $k-2$ times in a row at the same text position when searching for F_k. Again, you need to find an example text T that would cause KMP to traverse this entire chain on the same text character.

(e) How do the failure chains in parts (c) and (d) change if we use the optimized failure function instead?

3. Two-stage sorting

(a) Suppose we are given an array $A[1..n]$ of distinct integers. Describe an algorithm that splits A into n/k subarrays, each with k elements, such that the elements of each subarray $A[(i-1)k+1..ik]$ are sorted. Your algorithm should run in $O(n \log k)$ time.

(b) Given an array $A[1..n]$ that is already split into n/k sorted subarrays as in part (a), describe an algorithm that sorts the entire array in $O(n \log(n/k))$ time.

(c) Prove that your algorithm from part (a) is optimal.

(d) Prove that your alogrithm from part (b) is optimal.

4	14	7	3	1	20	11	9	5	13	12	19	10	16	17	2	8	6	18	15

$$\Downarrow (a)$$

1	3	4	7	14	5	9	11	13	20	10	12	16	17	19	2	6	8	15	18

$$\Downarrow (b)$$

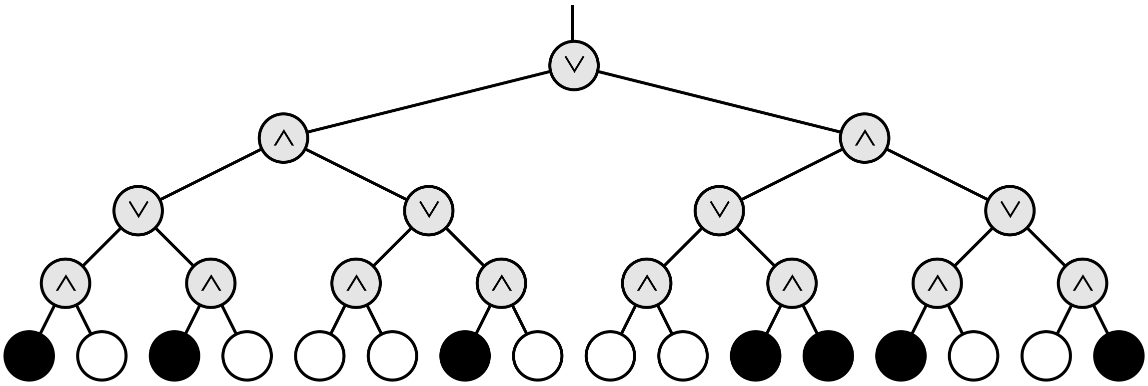
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

4. 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.)

5. Death knocks on your door once more on a warm spring day. He remembers that you are an algorithms student and that you soundly defeated him last time and are now living out your immortality. Death is in a bit of a quandry. He has been losing a lot and doesn't know why. He wants you to prove a lower bound on your deterministic algorithm so that he can reap more souls. If you have forgotten, the game goes like this: It is a complete binary tree with 4^n leaves, each colored black or white. There is a toke at the root of the tree. To play the game, you and Death took turns movin the token from its current node to one of its children. The game ends after $2n$ moves, when the token lands on a leaf. If the final leaf is black, the player dies; if it's white, you will live forever. You move first, so Death gets the last turn.

   You decided whether it's worth playing or not as follows. Imagine that the nodes at even levels (where it's your turn) are OR gates, the nodes at odd levels (where it's Death's turn) are *and* gates. Each gate gets its input from its children and passes its output to its parent. White and black stand for TRUE and FALSE. If the output at the top of the tree is TRUE, then you can win and live forever! If the output at the top of the tree is FALSE, you should've challenge Death to a game of Twister instead.

   Prove that *any* deterministic algorithm must examine *every* leaf of the tree in the worst case. Since there are 4^n leaves, this implies that any deterministic algorithm must take $\Omega(4^n)$ time in the worst case. Use an adversary argument, or in other words, assume Death cheats.



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

   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 $\Omega(V^2)$.

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).