

CS 373: Combinatorial Algorithms, Fall 2000

Homework 1 (due September 12, 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

Starting with Homework 1, homeworks may be done in teams of up to three people. Each team turns in just one solution, and every member of a team gets the same grade. 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.

Required Problems

1. Suppose we want to display a paragraph of text on a computer screen. The text consists of n words, where the i th word is p_i pixels wide. We want to break the paragraph into several lines, each exactly P pixels long. Depending on which words we put on each line, we will need to insert different amounts of white space between the words. The paragraph should be fully justified, meaning that the first word on each line starts at its leftmost pixel, and *except for the last line*, the last character on each line ends at its rightmost pixel. There must be at least one pixel of whitespace between any two words on the same line.

Define the *slop* of a paragraph layout as the sum over all lines, *except the last*, of the cube of the number of extra white-space pixels in each line (not counting the one pixel required between every adjacent pair of words). Specifically, if a line contains words i through j , then the amount of extra white space on that line is $P - j + i - \sum_{k=i}^j p_k$. Describe a dynamic programming algorithm to print the paragraph with minimum slop.

2. Consider the following sorting algorithm:

```

STUPIDSORT( $A[0..n-1]$ ) :
  if  $n = 2$  and  $A[0] > A[1]$ 
    swap  $A[0] \leftrightarrow A[1]$ 
  else if  $n > 2$ 
     $m \leftarrow \lceil 2n/3 \rceil$ 
    STUPIDSORT( $A[0..m-1]$ )
    STUPIDSORT( $A[n-m..n-1]$ )
    STUPIDSORT( $A[0..m-1]$ )

```

- (a) Prove that STUPIDSORT actually sorts its input.
 - (b) Would the algorithm still sort correctly if we replaced the line $m \leftarrow \lceil 2n/3 \rceil$ with $m \leftarrow \lfloor 2n/3 \rfloor$? Justify your answer.
 - (c) State a recurrence (including the base case(s)) for the number of comparisons executed by STUPIDSORT.
 - (d) Solve the recurrence, and prove that your solution is correct. [Hint: Ignore the ceiling.] Does the algorithm deserve its name?
 - *(e) Show that the number of *swaps* executed by STUPIDSORT is at most $\binom{n}{2}$.
3. The following randomized algorithm selects the r th smallest element in an unsorted array $A[1..n]$. For example, to find the smallest element, you would call RANDOMSELECT($A, 1$); to find the median element, you would call RANDOMSELECT($A, \lfloor n/2 \rfloor$). Recall from lecture that PARTITION splits the array into three parts by comparing the pivot element $A[p]$ to every other element of the array, using $n - 1$ comparisons altogether, and returns the new index of the pivot element.

```

RANDOMSELECT( $A[1..n], r$ ) :
   $p \leftarrow \text{RANDOM}(1, p)$ 
   $k \leftarrow \text{PARTITION}(A[1..n], p)$ 
  if  $r < k$ 
    return RANDOMSELECT( $A[1..k-1], r$ )
  else if  $r > k$ 
    return RANDOMSELECT( $A[k+1..n], r-k$ )
  else
    return  $A[k]$ 

```

- (a) State a recurrence for the expected running time of RANDOMSELECT, as a function of n and r .
- (b) What is the *exact* probability that RANDOMSELECT compares the i th smallest and j th smallest elements in the input array? The correct answer is a simple function of i , j , and r . [Hint: Check your answer by trying a few small examples.]