Name_____

MATH 163/COEN 179 Winter 2014
Midterm 7 Feb 2014

1. A sorting method known as shellsort uses at worst $\Theta(n^{3/2})$ comparisons to sort an array of n entries. How does that compare to other sorts you know?

2. An array contains the entries `CRAZYQUESTION`.

   a) What would be the last two comparisons performed by insertion sort?

   b) Using our implementation of quicksort, what are the first three entries that would be used as pivots?

   c) Show the contents of the array after using the bottom-up method to build a heap.

3. Suppose you wish to store the exact value for $n!$. Typically, this will end in lots of zero bits. Rather than store the trailing zeros, you need only store the bits that precede them, and store the number of trailing zeros. For example, with

$$18! = 1011010111110111011001100101001110011000000000000000,$$

   you need only store the significant digits

$$101101011111011101100110010100111001 1$$

   and the number of zeros

$$10000$$

   (16 in binary, that is). The total number of bits before chopping is $\lfloor \lg n! \rfloor + 1$, and it turns out that the number of trailing zeros is given exactly by $n - \text{bitsum}(n)$, where $\text{bitsum}(n)$ denotes the number of 1's in the binary expression of n. (For example, 18 = 10010, so $\text{bitsum}(18) = 2$.) Let $f(n)$ represent the number of bits necessary to store the significant digits of $n!$ and let $g(n)$ represent the number of bits necessary to store the number of trailing zeros of $n!$. (For example,$f(18) = 37$, and $g(18) = 5$.) Show that

$$\frac{f(n)}{g(n)} = \Theta(n).$$

4. Using the master theorem, determine the complexity of the solutions to the following recurrences. Show enough work so that I can follow your reasoning.

   a) $T(n) = 1$ for $n < 2$; $T(n) = 8T(n/2) + 3n^2$ for $n \geq 2$;

   b) $T(n) = 1$ for $n < 2$; $T(n) = 4T(n/4) + 4n$ for $n \geq 2$;

   c) $T(n) = 1$ for $n < 2$; $T(n) = 2T(n/8) + 5\sqrt{n}$ for $n \geq 2$;

5. Including the initial call, how many times does mergesort get called while sorting an array with n entries? You may give an answer in $O(g(n))$ form, but be as precise as you can. Justify your answer.

6. We saw in class that searching an unsorted array of n entries takes at most n comparisons, but that using binary search on a sorted array takes at most the number of bits in the binary representation of n. Suppose that you are randomly searching for an array item that you know is there, and that all entries are searched with the same probability.

   a) Show that the expected number of comparisons to locate the entry using sequential search (that is, the average number of comparisons) is $(n+1)/2$. [So, the expected number of comparisons is roughly half of the worst-case number of comparisons.]

   b) Is the expected number of comparisons to locate the entry in the sorted array using binary search roughly half the number of bits used to represent n? If not $1/2$, state what the proportion of the expected number of comparisons to the worst-case number of comparisons roughly should be.