Test 1
Thu 22 Feb 2018
1. For each question below, give a short answer - a few words or symbols, maybe a sentence or two.

(a) (4 points) Give an example of an algorithm which has average time of $O(n^2)$ but is not $\Theta(n^2)$. Explain what sort of special situations would lead to the better runtime.

(b) (5 points) Give an example of an algorithm which run (on average) in $O(n \lg n)$ time, but which will occasionally do worse.
   - Name the sort
   - Give the true worst-case runtime
   - Explain why it sometimes has worse than $O(n \lg n)$ time; how does this happen?

(c) (2 points) Identify two sorting algorithms which are known to have worst case running times of $O(n \lg n)$. (You don’t have to justify why these algorithms run that quickly.)

(d) (2 points) Give an example of a sort algorithm which is a stable sort.

Give an example of a sort algorithm which is a not a stable sort.

(e) (2 points) Explain why no sorting algorithm can have performance better than $O(n)$. 
2. (a) (10 points) Give the formal definition for $O(g(n))$.

(b) (5 points) We think of a heap as a binary tree. How do we actually store it in memory? What is special about a heap which makes this possible?

(c) (5 points) Explain the difference between keys and satellite data (also known as “values”) in a sort algorithm.
3. (a) (5 points) Suppose that you have a predicate \( P(x, y) \). Write formal quantifiers to express the following English language statements. If negations are required, you are not required to simplify those negations; just write a correct expression.

“\( P(x, y) \) always returns true, no matter what parameters you give it.”

“At least one combination of \((x, y)\) will be rejected by \( P \).”

(b) (5 points) Again, convert English to a formal expression; this time, the quantifier is \( Q(x) \).

For full credit, convert this to a mathematical expression (an English language explanation is not required).

For half credit, break this expression down into simpler expressions, but still express them in English - and skip the mathematical formula.

“\( Q(x) \) is true for only one \( x \).”

4. (10 points) On the next page, I’ve given an implementation for the \textbf{merge()} step of Merge Sort. I have inserted quite a few bugs. Edit the code to make the code work correctly.

The parameters are:

- The array of input values
- The index of the start, midpoint, and end \((\text{beg}, \text{mid}, \text{end})\)

The first sub-array to merge goes from \text{beg} (inclusive) to \text{mid} (exclusive). The second sub-array to merge goes from \text{mid} (inclusive) to \text{end} (exclusive). (Don’t assume that the two sub-arrays are the same length!)

Place the merged data back into the original array; you may allocate a temporary buffer.

You may assume that all of the parameters are valid: the array is not \text{null}, and the indices are all non-negative, and the indices obey the assumption \text{beg} \leq \text{mid} \leq \text{end} \leq \text{vals.length}.
void merge(int[] vals, int beg, int mid, int end)
{
    int[] tmp = new int[vals.length];
    int out=0, L=0, R=0;
    while (L < mid || R < end)
    {
        if (vals[L] < vals[R])
            {
            tmp[out] = vals[L];
            L++;
            out++;
        }
        else
            {
            tmp[out] = vals[R];
            R++;
        }
        out++;
    }
    while (L < mid)
    {
        tmp[out] = vals[L];
        L++;
        out++;
    }
    while (R < end)
    {
        tmp[out] = vals[R];
        R++;
        out++;
    }
    for (int i=0; i<tmp.length; i++)
    {
        vals[beg+i] = tmp[i];
    }
}
5. (15 points) Use the Master Method to solve the following recurrences. If the recurrence cannot be solved by the Master Method, give a short explanation why not.

(a) \( T(n) = 8T\left(\frac{n}{3}\right) + n^2 \)

(b) \( T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n} \)

(c) \( T(n) = 2T(n - 100) + \lg n \)

(d) \( T(n) = 27T\left(\frac{n}{3}\right) + n^2 \)
Choose only one of the induction problems to do. Do not do both - if you do, we will only grade one of them!

6. (20 points) Consider a (non-empty) trinary tree, with a special restriction: every internal node has exactly three children (never less).

Using structural induction, prove that the number of leaves is always odd.

(You may use any of the structural induction strategies.)

HINT: Trees of this form can only have certain numbers of nodes: 1, 4, 7, 10...
Choose only one of the induction problems to do. Do not do both - if you do, we will only grade one of them!

Using induction, prove the following conjecture:

\[
\sum_{i=0}^{n-1} ar^i = \frac{a(1 - r^n)}{1 - r}, \text{ where } n \in \mathbb{Z}^+, r \neq 1.
\]
7. (a) (5 points) Perform Radix Sort on the data below; show the complete contents of the array after every pass. As with Project 2, use decimal digits as the “columns” of each key.

82
962
770
962
465
941
769
727

(b) (5 points) Use the median-of-3 algorithm to choose a pivot for each of the following arrays. Circle the value that you choose as the pivot.

[149, 741, 40, 300, 1, 941, 402]

[89, 7, 305, 32, 07, 953, 706]

[450, 91, 328, 47, 76, 416, 61]

[22, 929, 7, 6, 717, 41, 679]

[650, 563, 222, 68, 618, 116, 6]