CS Comprehensive Exam
Spring 99
Analysis of Algorithms

This exam consists of **3 questions** that are worth 100 points. Please give **brief** and **precise** answers.

1. There are three cases to be considered when analyzing an algorithm: Best, Worst, and Average.

   (a) Which one of the cases is **often** studied for efficiency measurement? Explain.

   (b) Which case is **sometimes** studied for efficiency measurement but not very often? Explain.

   (c) Which case is **seldom** used as efficiency measurement? Why?

   (d) What is the best, worst and average complexity of the following algorithm for comparing whether or not two strings `s` and `t` are equal?

```
int strcmp(s, t)
char *s, *t;
{
  int i;
for (i = 0; s[i] == t[i]; i++)
  if (s[i] == '\0') return 0;
  return s[i] - t[i];
 }
```

2. (**45 points**) The number of operations in a sorting algorithm, called FOO, for sorting n numbers satisfies the recurrence

$$T(n) = 2T(n-1) + n$$

(a) Solve the above recurrence (remember to indicate the basis)

(b) What is the time complexity using big-O notation?

(c) How would you classify FOO among other sorting algorithms taking into consideration its complexity?

3. (**45 points**) The following chunk of (modified) code comes from wavelet transforms. It assumes an image that is $m \times n$ where both m and n are powers of 2. For each row $(0, \ldots n-1)$ it adds or subtracts certain elements in the row storing them in `scratch` space which is returned by the routine.

```
public double[] reconstruct (int row, int col, int n, int m) {
  double[] scratch = new double[n];
  for (int i = 0; i < n; i++) {
    scratch[i] = image[i][0];
    int index = m;
    int c = col;
    while (index > 1) {
      index = index/2;
      if (<<c is even>>) {
        int d = c/2;
        scratch[i] = scratch[i] + image[i][d+index]
      }
      else {
        scratch[i] = scratch[i] - image[i][d+index]
      }
      c = c/2;
    }
  }
  return scratch;
}
```

(a) Give a formula that describes the time complexity $T(n, m)$ of the algorithm “reconstruct”.

(b) Solve the above formula (convert it into a closed form).

(c) Indicate the asymptotic run time (order of growth) in big-O notation.