CSci 1302 Independent work. April 16 2010

Your names:

Work individually or in groups of two. Please write down your answers the best you can. It's OK if your answers are incomplete (a substantial part of your grade is participation). However, if your answer is incomplete, please explain where you are stuck.

We are using := for setting a variable to a value and = for a comparison operation. We count comparisons and additions for efficiency measurement.

Problem 1: Cocktail sort. This sorting algorithm is similar to Bubble sort with two important differences:

- on every iteration of an outer loop it moves the largest ("heaviest") element to the end of the array and the smallest ("lightest") element to the beginning.
- the algorithm has a variable "swapped" that is set to true if in the iteration of the loop there were swapped elements. If there are no swapped elements then the algorithm stops (see "stop" below).

Below is the pseudocode for the algorithm. We assume that A is an array of non-ordered elements of length n.

```
 swapped := true
cocktailSort
  while (swapped = true)
    swapped := false
    for i from 1 to n-1 do:
      if A[ i ] > A[ i + 1 ] then
        swap( A[ i ], A[ i + 1 ] )
        swapped := true
      end if
    end for
    if swapped = false then stop
    swapped := false
    for  i from n - 1 to 1 do:
      if A[ i ] > A[ i + 1 ] then
        swap( A[ i ], A[ i + 1 ] )
        swapped := true
      end if
    end for
 end  while
```

1. Show step-by-step the work of the algorithm on the array 2, 3, 5, 1, 4

2. What is the Θ of the worst case of the algorithm? When does the worst case happen? Make sure to explain how you got the approximation.

3. What is the Θ of the best case of the algorithm? When does the best case happen? Make sure to explain how you got the approximation.

Problem 2: Sum of all elements of an array. You are given an array of n integers $A[1], A[2], \ldots, A[n]$. For simplicity assume that $n = 2^k$ for some k. Consider a summation algorithm that uses a divide-and-conquer approach to sum up all elements of the array, similar to mergesort or quicksort where we repeat the algorithm for a smaller portion of the array.
The algorithm works the following way: if the array has only one element then its sum is just that element. Otherwise its sum is the sum of the first half (computed using the same sum function) plus the sum of the second half (the sums of the smaller arrays are computed using the same function `sum`).
Below is the algorithm in pseudo code:

```
sum(a[n1]...a[n2]) {
      if (n1 = n2) return a[n1]
      thesum  := sum(a[n1]...a[(n2-n1)/2]) +
                      sum(a[(n2-n1)/2 + 1]...a[n2])
      return thesum
}
```

The first call is `sum(a, a[0]...a[n-1])`.

1. Show how the algorithm finds the sum of all elements in the array 3, 2, 7, 8, 1, 4, 6, 5

2. How many levels does the algorithm have?

3. How many operations does it make at each level?

4. Guess or prove the Θ for this algorithm. Explain your answer.