Recursion

**Please read turn-in checklist at the end of this document before you start doing exercises.**

**Section 1: Pen-and-paper Exercises**

1. Analyze the following code and provide a "Big-O" estimate of its running time in terms of n. Explain your analysis.

function fun(int n)

{

if (n == 0)

return 0;

else

return n + fun(n-1);

}

**Note: Credit will not be given only for answers - show all your work:**

**(3 points) steps you took to get your answer.**

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

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

T(n-2) = n + T(n-3)

T(n-3) = n + T(n-4)

…

T(n-n) = T(0) = O(1)

T(n) = T(n-1) + O(1) = O(n)

**(2 points) your answer.**

O(n)

1. We are given an array A[] of n numbers in an arbitrary order. Design an algorithm to find the minimum and second minimum element in A[] using at most 3/2n -2 comparisons.
2. describe the idea behind your algorithm in English (3 points);`

Get the size of the array.if it is element 1 return both as mins, if it is elements 2 return the lower as min and second lowest as 2ndmin. Otherwise split the array in 2. Recursive call to find the min and 2ndmin. Compare the min and 2ndmin values of each half and output the 2 smallest values.

1. provide pseudocode (5 points);

1. analyze the number of comparisons used in your algorithm (2 points).

T(1) = 0

T(2) = 1

T(n) = T(n/2) + 2

T(n) = 2T(n/2) + 2

So 3/2n -2

**Note: Full credit (10 points) will be awarded for an algorithm that uses at most 3/2n -2 comparisons. Algorithms that make more comparisons will be scored out of 3 points.**

1. Using the master theorem discussed in class, find a tight bound for the solution of the following recurrence equation (3 points each).
2. T(n) = 2T(n/2) + n3
   * a=2, b=2, d=3
   * d = 3, = 1
     1. d >
   * O(n3)
3. T(n) = T(9n/10) + n
   * a=1, b=0.9, d=1
   * d = 1, = 0
     1. d >
   * O(n1)
4. T(n) = 16T(n/4) + n2
   * a=16, b=4, d=2
   * d = 2, = 2
     1. d =
   * O(n2 log n)
5. T(n) = 7T(n/3) + n2
   * a=7, b=3, d=2
   * d = 2, = 1.77
     1. d >
   * O(n2)
6. T(n) = 2T(n/4) + sqrt(n)
   * a=2, b=4, d=-1/2
   * d = -1/2, = 1/2
     1. d < = 1/2
   * O()

**Section 2: Java Implementation**

1. Implement problem 2 in Java (30 points).

Note:

Find a file called Problem2.java in assignment 4 folder.

Complete the method of dcfindmin2ndmin ().

Test your method in the main method provided following the comments.

**Full credit (30 points) will be awarded for an algorithm that uses at most 3/2n -2 comparisons. Programs that make more comparisons will be scored out of 5 points.**

**TURN-IN CHECKLIST:**

1. **Answers to Section 1 (.doc/.txt), and to Section 2 (all your source Code (.java files)). Remember to include your name, the date, and the course number in comments near the beginning of your code/report.**
2. **Create a folder and name it 'FirstName\_LastName\_assignment\_4'. In the newly created folder copy and paste your files (.doc/.txt/.java files). Then compress the folder, and submit to iLearn.**