HW3

Due 11:59PM Thursday Feb 10

Total points: 30

You can work on this homework in a team of two students. Write down the team members' names in the paper and the program. If your group collaborate with other teams, indicate the teams.

1. (4pts) Use mathematical induction to prove that \( n! > 2^n \) for \( n \) a positive integer greater than or equal to 4. (Note: \( n! \) is \( n \) factorial and is given by \( 1 \times 2 \times \ldots \times (n - 1) \times n \).)

2. (4pts) Show that the solution of \( T(n) = T(\lceil n/2 \rceil) + 1 \) is \( O(\lg n) \).

3. (4pts) Show that the solution of \( T(n) = 2T(\lceil n/2 \rceil) + n \) is \( O(n \lg n) \). Show that the solution of this recurrence is also \( \Omega(n \lg n) \). Conclude that the solution is \( \Theta(n \lg n) \).

4. (6pts) Design efficient algorithms (written in pseudocode) that take an array of positive numbers \( a \), and determine
   (a) (2pts) the maximum value of \( a[j] + a[i] \), with \( j \geq i \);
   (b) (2pts) the maximum value of \( a[j] - a[i] \), with \( j \geq i \);
   (c) (2pts) Tell the growth rates of the algorithms in \( \Theta \).

5. (12pts) Programming

   Analyze, and write a Java program which performs shuffles (i.e. permutations) on lists of integers. Observe that there are many ways to shuffle a given list of integers. For example the list \( (1 \ 2 \ 3) \) can be shuffled in 6 distinct ways: \( (1 \ 2 \ 3), (1 \ 3 \ 2), (2 \ 1 \ 3), (2 \ 3 \ 1), (3 \ 1 \ 2), (3 \ 2 \ 1) \). In general a list of length \( n \) has \( n! \) arrangements or permutations. One algorithm for permutation is:

   Algorithm: permute (char[] A, int begin, int end)

   INPUT: A: the input character array, begin: beginning index, end: ending index

   OUTPUT: none
if end-begin == 0 then
    print A
else
    for i ← 0 to end − begin do
        swap A[begin] and A[begin + i]
        permute(A, begin+1, end)
        swap A[begin] and A[begin+i]
    end for
end if
return

(a) (4pts) For the algorithm
   i. (0.5pts) Tell the input size.
   ii. (0.5pts) Tell the algorithm basic operation.
   iii. (0.5pts) Determine worst, average, and best cases for input of size n.
   iv. (1pts) Set up a sum if the algorithm is nonrecursive, or a recurrence relation if the
           algorithm is recursive, for the number of times the basic operation is executed.
   v. (1pts) Simplify the sum, or solve the recurrence.
   vi. (0.5pts) Tell the Big-Oh running time of the function.

(b) (8pts) Implement the algorithm. Write a java class Permute.java, which takes a command argument (int n) as input, and output the permutations of the n numbers from 1 to n inclusive. For example, if the input int is 3, then the permutations are 123, 132, 213, 231, 312, and 321. The order in which the permutations are printed is not important.

6. Turn in instructions:
   (a) Turn in your program file as "Permute.java" through turnin. "Permute.java" will have a main method that takes the input int number as an argument to the executable. So, you should run with a string of length 4 as follows:
javac Permute.java
java Permute 4

Print one permutation terminated by a newline at each line (The characters in a permutation are one after another, i.e. NOT separated by the space character). An example output for “1234” is provided as a file “permutation-sampleoutput”. You should print all possible permutations. However, the order of the permutations may be different than this output (This file is given to show the output format, you will print out to standard output not to a file).

To make sure your program truly works, you should use additional cases with different lengths for testing.

**Note:** Failure to follow the convention about the program name and usage may significantly delay grading of your assignment.