Assignment 0

Min, Max, and All That

Out: January 16, 2015. **Due**: January 29, 2015.

Consider the following search problem:

MIN-MAX
Input: An array A of integers of size N (where N is a positive integer).
Output: The smallest and largest elements in A.

Assume that the cost model charges an algorithm one unit for every comparison that involves an element of the input array. This should holds even if the array element is transfer out of the array to a different variable.

We will consider two algorithms for this problem: one is recursive (as covered in class) and the other is nonrecursive. Our goal is to analyze both algorithms mathematically and experimentally.

1. Write a C++ program that solves the **MIN-MAX** problem in a recursive manner (using a divide-conquer strategy).
    a. Your program should accept as input a filename as a *command line argument*. Suppose the file containing the input is called 'data.txt'. Then, the format of 'data.txt' should be as follows:

    ```
    10
    11 19 13 7 5 17 3 2 23 29
    ```

    The first line contains the number of elements (N or size) while the second line contains the elements (separated by space).

    The output of the program should be four numbers in the following order: the size of the input array (that is, N), the minimum element, the maximum element, and the cost (the number of comparisons made by your program on the input).

    Here is an example of a sequence of compile and run (under Linux) on the above input:

    ```
    % g++ -g -c main.cpp
    % g++ -g -c minmax.cpp
    % g++ -g -o my_rec_minmax main.o minmax.o
    % ./my_rec_minmax data.txt
    10 2 29 14
    ```

    Note: the symbol '%' represents the command line prompt.

    b. Analyze the performance of your recursive program by stating and solving the recurrence formula for its cost function (see our class discussion). Investigate the following statements:

    > **True or False**: The cost of your recursive algorithm is 3/2N - 2 for N that is a power of two.
    >
    > **True or False**: The recursive algorithm performs worse than 3/2N-2 on N that are not powers of two.

    c. Prepare a plot of the cost (number of comparisons involving array elements) of your recursive C++ program as a function of the array size. The range of the size should be between 1 and (at least) 1000. On the plot, also superimpose the function 3/2N-2.

2. Write a C++ program that solves the **MIN-MAX** problem in a **non-recursive** manner (basically, use a loop instead of recursion).
    a. Same as part 1 above.

    b. Analyze the performance of your nonrecursive program. Based on this, investigate the following statements:

        > **True or False**: Your nonrecursive algorithm is as good as the recursive algorithm on N that are powers of two.
        >
        > **True or False**: Your nonrecursive algorithm outperforms the recursive algorithm on N that are not powers of two.

    c. Prepare a plot of the cost (number of comparisons involving array elements) of your nonrecursive C++ program as a function of the array size. The range of the size should be between 1 and (at least) 1000. On the plot, also superimpose the function 3/2N-2.

What to Hand-in

Source:
Submit a single compressed file by email (zip or tgz) via **moodle**.

- Your programs should compile (with g++) and run (under Linux) in the ITL lab or on the AFS system.
- For parts 1 and 2, place your C++ programs in files called 'my_rec_minmax.cpp' and 'my_iter_minmax.cpp' respectively.
- The executables should conform to the input/output conventions outlined above.
- If you have more than one source file (for each part), provide a makefile.
- Your source programs should be properly documented (don't forget to include your name in the documentation).

Testing:
Submit a **hardcopy** of a single plot showing the costs of both your recursive and nonrecursive algorithms as functions of the input size. The input sizes should range from 1 to (at least) 1000.

Analysis:
Submit a **hardcopy** of your mathematical analyses of both the recursive (see class discussion) and nonrecursive algorithms (see Chapter 1 of our textbook for ideas).
