**Assignment: Algorithm Analysis**

Last modified 26 January

This assignment is due to be completed and submitted by 1 pm Wednesday, 2 February.

Here is an algorithm, foo:

01 void foo(vector<unsigned>& array)

02 {

03 size\_t n = array.size();

04

05 for (size\_t start = 0; start < n - 1; start++)

06 {

07 unsigned item = array.at(start);

08 size\_t position = start;

09

10 for (size\_t index = start + 1; index < n; index++)

11 {

12 if (array.at(index) < item)

13 {

14 position++;

15 }

16 }

17

18 if (position != start)

19 {

20 while (item == array.at(position))

21 {

22 position++;

23 }

24 swap(array.at(position), item);

25

26 while (position != start)

27 {

28 position = start;

29 for (size\_t index = start + 1; index < n; index++)

30 {

31 if (array.at(index) < item)

32 {

33 position++;

34 }

35 }

36

37 while (item == array.at(position))

38 {

39 position++;

40 }

41 swap(array.at(position), item);

42 }

43 }

44 }

45 }

Your assignment is to do the following:

* Trace the code by hand to figure out what it does, what its purpose is.
* Examine the algorithm and create an inequality that expresses the best case number of operations as a function of input size (which you must define) and an inequality that expresses the worst case number of operations as a function of input size. Based on these functions, determine to which big-O, big-Omega, and big-Theta classes the algorithm belongs.
* Write a C++ program that implements the algorithm exactly as given above, augmented with code that counts the number of basic operations.   
    
  For each thing you count, add a comment in the code explaining what you are counting. For example:
* for (unsigned outer = 0; outer < n; outer++)
* {
* count += 2; // for header

Your program must run with a single command line argument which is treated as the number of elements with which to populate a vector with randomly generated unsigned values. For generating the random values, please use the code I showed you in the first program (and which is repeated in the template).   
  
Your program must produce exactly one line to cerr that prints the input size, a space, and a count of the basic operations performed, and a single line to cout that prints the final contents of the vector, space separated, after foo has been called. A run of your program must look exactly like this:

$ ./program 5

5 1234

71590 237134 257967 259965 280757

except that you will have different values, and 1234 is probably not correct. Remember that every output stream must be terminated by endl.

* Run the program in such a way that the algorithm is exercised many times with many different arguments, and capture the results in a file. For example, to run the program 3000 times with 1000 different array sizes, you might use a command such as:
* $ for n in $(seq 1000 100 100000)
* do
* ./program $n
* ./program $n
* ./program $n
* done 2> results.dat
* Write a paper that explains all of the above: the purpose of the algorithm, the functions for best and worst case along with their derivations, the efficiency class or classes to which the algorithm belongs. Include one or more plots of input size vs. basic operations, along with one or more standard functions properly scaled, to illustrate the theoretical analysis.

Just to make sure we are all on the same page, we adopt the following conventions about basic operations:

* A vector access such as foo.at(index) is one operation.
* A vector knows its size without needing a true method call, so code such as size\_t n = foo.size() is only one operation, for the assignment.
* The built-in C++ operation swap is extremely efficient, and requires only two basic operations.

In the [current directory](https://borax.truman.edu/310/202/) there are

1. A LaTeX template that you should use as the basis for your write-up. The structure of the template is correct, but of course many of the words are nonsense that you must change. For the LaTeX to compile, you will have to supply an image file.
2. A PDF which is the result of typesetting the LaTeX, just so you can see your starting point.
3. A C++ template that you should use as the basis for your program. It does no error-checking, and you do not have to worry about erroneous input.

Make sure your C++ code adheres to the [course coding standards](https://borax.truman.edu/310/coding_rules.html).

Submit the LaTeX source code, the resulting PDF document, and the C++ source code file to the [homework submission](https://borax.truman.edu/310/submit.php) page.