# **Assignment for 4 September**

Last modified: 28 August 2020

This assignment is due to be completed and submitted by noon Friday, 4 September.

Use LaTeX to typeset the following questions and their answers. Show and explain your work in all cases.

1. Use the definitions to prove or disprove the statement 2*n*2 + 3 ∈ *O*(*n*3), and illustrate your results graphically.
2. Either prove the following assertion using the definitions or disprove it with a specific counterexample: if *T*(*n*) ∈ *O*(*S*(*n*)) then *S*(*n*) ∈ *Ω*(*T*(*n*))

For the following algorithm, explain what it computes, state what the input size for analysis is, state what basic operations should be counted for analyzing it, state exactly how many operations are executed as a function of the input size, and state the efficiency class to which it belongs. The built-in C++ swap function is extremely efficient, and you can count it as performing exactly two basic operations.  
 void foo(vector& array)

{

size\_t n = array.size();

for (size\_t pass\_indx = 0; pass\_indx < n - 1; pass\_indx++)

{

size\_t min\_position = pass\_indx;

for (size\_t compare\_indx = pass\_indx + 1; compare\_indx < n;

compare\_indx++)

{

if (array.at(compare\_indx) < array.at(min\_position))

{

min\_position = compare\_indx;

}

}

if (min\_position != pass\_indx)

{

swap(array.at(pass\_indx), array.at(min\_position));

}

}

}



Write a C++ program that implements the algorithm in problem 3, counts the number of basic operations, and outputs the input size and the count of basic operations to the cerr stream. Your program must accept the size of the array as a command line argument, it must produce exactly one line to cout that prints the contents of the array, space separated, after foo has been called, and it must produce exactly one line to cerr that prints the input size, a space, and a count of the basic operations. A run of your program must look exactly like this:  
 $ ./program 5

71590 237134 257967 259965 280757

5 12345

(except that you will have different values, and 12345 is almost certainly not correct). For this program, you will need to generate high-quality pseudorandom numbers, better than simple rand() can produce. Include the chrono system library, and use this code:  
 const unsigned MAX\_RANDOM\_VALUE = 1000000;

default\_random\_engine get\_next\_value

(static\_cast <unsigned>

(chrono::system\_clock::now().time\_since\_epoch().count()));

uniform\_int\_distribution<unsigned> uniform(0, MAX\_RANDOM\_VALUE);

Then, when you need a random number, do this:  
 unsigned random\_value = uniform(get\_next\_value);

2. Run the program in such a way that the algorithm of Problem 3 is exercised many times with many different inputs and input sizes, and capture the results in a file. Show the command(s) you used to do this. Use a plot of input size vs. basic operations, along with one or more standard functions properly scaled, to illustrate the analysis you did in Problem 3.

In the [current directory](http://borax.truman.edu/310/0904/) there is a template LaTeX document that you should use as the basis for your document, as well as the results of running pdflatex on that template. The structure of the template is correct, but of course many of the words are nonsense that you must change.

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

By the due timestamp, submit the LaTeX source code, the resulting pdf document, and the C++ source code file to the [homework submission](http://borax.truman.edu/310/submit.php) page.