

# CIS 213: Foundations of Computer Science—Lab 4

In this lab, you will be exploring the use of recursion in several example programs. Including:

- Factorial
- Selection sort
- Merge sort
- Practical Comparison of  $O(n \log n)$  versus  $O(n^2)$  sorting

Create a `cis213/lab4` directory.

Copy all files from the `~/hardnett/pub/cis213/lab4` directory. Check that you have the following files in your lab directory:

1. `fact.cc`: factorial
2. `gram.cc`: parser for simple `jexpri` grammar
3. `ssort-rand.cc`: selection sort of arrays
4. `msort-array.cc`: merge sort of an array
5. `ssort-recur.cc`: recursive selection sort

## 1 Comparing Execution Time

In this section you will compare the execution times of the `ssort-rand.cc` and `msort-array.cc`. First compile these 2 programs using `g++`.

Use the "time" command to see how long it takes to sort an array of size 5000, 6000...10000, 12000, 14000, and 16000. Open a file called `ssort.dat` and a file called `msort.dat`. Put the sizes and times in the corresponding files (on each row, the values are separated by a TAB):

5000	2.8
6000	4.1
7000	5.4
...	...
16000	36.3

use commenad like the following to run the programs and collect data.

```
host% time ssort-rand 5000 n
...
host% time msort-array 5000 n
...
```

You can plot your files with Gnuplot as follows.

```
host% gnuplot
gnuplot> plot 'ssort.dat' with lines
```

You can plot two datasets together:

```
gnuplot> plot 'msort.dat', 'ssort.dat'
```

You can also plot continuous functions. Here's how to plot  $15x^2$ :

```
gnuplot> plot 15*x**2
```

Here's how to plot the  $.01x^2 + 10$  from zero to 1000:

```
gnuplot> plot [0:1000] .01*x**2
```

Here's how to plot a function and a datafile together:

```
gnuplot> plot [0:1000] .01*x**2, 'ssort.data' with lines
```

Try various values to see what coefficients make the  $x^2$  quadratic function and the data file match up. Try the same thing with the mergesort data. *Hint:* You're going to need some small coefficients i.e. 0.000...1 values.

**Q1.** Size is on the x-axis and time is on the y-axis, What does it mean to have have algorithm like ssort where the plot is an x-squared function vs an algorithm like msort where the plot is a log(x) function?

**Q2.** Which algorithm would you use to sort large databases of millions or billions of entries? Why?

## 2 Recursive Algorithms

A recursive algorithm is implemented as a recursive function in a programming language like C/C++. A function is recursive when its definition makes a call to itself. This is analogous the the mathematical formulation of recursive functions.

### 2.1 Factorial

Consider factorial as a mathematic function:

**basis**  $1! = 1$

**recursive**  $n! = n \times (n - 1)!$

Where the C/C++ implementation as a recursive function is:

```
unsigned Factorial(unsigned n)
{
    // Base Case
    if (n == 1)
        return 1; // 1! is 1
    else // recursive case
        return n * Factorial(n-1); // n! = n * (n-1)!
}
```

Compare the mathematical formulation and the C/C++ implementation. Compile the `fact.cc` and execute it:

```
host% g++ fact.cc -o fact
```

```
host% fact 1
```

```
host% fact 2
```

```
host% fact 6
```

### 2.2 Grammars

Recursive functions may have other structures, where there are multiple base cases and multiple recursive cases.

Consider the recursive grammar:

```
<expr> := int | var | float
<expr> := <expr>;
<expr> := <expr> + <expr>
          | <expr> - <expr>
          | <expr> * <expr>
          | <expr> / <expr>
```

This is implemented with the following recursive function:

```

unsigned expr(istream& stream)
{
    string operand;
    char op;

    if (stream >> operand) {
        // ... stuff removed
        stream >> op;

        switch (op) {
            case '+': // recursive case for +
                expr(stream);
                break;
            case '-': // recursive case for -
                expr(stream);
                break;
            case '*': // recursive case for *
                expr(stream);
                break;
            case '/': // recursive case for /
                expr(stream);
                break;
            case ';': // base case for ;
                exit(0);
                break;
            default:
                cerr << "Syntax Error at char = " << op << endl;
                break;
        }
    }
}
}

```

Compare the grammar to the C/C++ parser implementation. Compile the `gram.cc` and execute it:

```
host% g++ gram.cc -o gram
```

```

host% gram
Enter the expression you want to parse/test.
End your expression with a ;.
x * y * 5 + 2 / 3;

```

```

host% gram
Enter the expression you want to parse/test.
End your expression with a ;.
x * y * 5 + 2 / 3 5;

```

Compare the output of the `gram` program with a derivation that you would do by hand. Try to make your derivation mimic the path the program used in its parsing.

### 2.3 Recursive Selection Sort

Iteration and recursion are very much related. The selection sort algorithm seen earlier, can be written recursively as follows:

```

void recursiveSelSort(int i, int n)
{
    int j, small, temp;

```

```

if (i < n-1) { /* basis is when i = n-1, in which case */
    /* the function returns without changing A */
    /* induction follows */
    small = i;
    for (j = i+1; j < n; j++)
        if (Nums[j] < Nums[small])
            small = j;
    temp = Nums[small];
    Nums[small] = Nums[i];
    Nums[i] = temp;
    recursiveSelSort(i+1, n);
}
}

```

**Q3.** What is the difference between the iterative version of the selection sort and the recursive version?

### 3 Merge Sort

The merge sort program is in the file `m-sort-array.cc`. The approach of the merge sort is to recursively sub-divide the array by 2 until sub-arrays of size 1 have been reached. When sub-arrays of 1 are reached, then they are each sorted by definition of single element array.

Details can be found on pages 75-79.

**Q4.** How many recursive calls are required to sort an array of size 10, 20, or size  $n$ ? (you may add counters to your code to figure this out).

Put your answers in a `lab4.ans` and submit:

```
host% turnin cis213 lab4
```

Be sure that you see that your `lab4.ans` file was submitted.