Lab 4: Sorting by Brute-Force

COSC 3020: Algorithms and Data Structures

Lars Kotthoff
larsko@uwyo.edu

02 October 2019

Instructions

Attempt to finish the tasks below during the lab time. You have until Friday, 04 October 2019, 23:59h to submit the solutions to WyoCourses. You may ask your TA for feedback before submitting, but this feedback will be qualitative only.

You may *not* use external libraries in your code unless explicitly stated.

1 Permutation Sort

We talked about the complexity of the sorting problem, and used an argument over all permutations of a list to be sorted to determine its complexity. Implement a function to sort a list by systematically trying all permutations of the input list. It should have the following signature:

```
function permutationSort(a);
```

The return value should be the number of permutations that were tried until the sorted list was "discovered".

Submit your complete code, including a function that demonstrates that your implementation works with a few test inputs.

Total 6 points.

2 Runtime Analysis

What is the runtime complexity of the algorithm that you implemented? What does a best case input for your implementation look like, what does a worst case input look like? How would this complexity change if you generated permutations randomly without memory instead of systematically trying them?

Submit a PDF document describing your reasoning and the answers. Your reasoning is the most important part.

Total 4 points.

Testing

You can use `jsverify` (`https://jsverify.github.io/`) to automatically test your code. Assuming that your code is in a file `code.js`, you can use the following code to test it:

```
const fs = require('fs');
const jsc = require('jsverify');

eval(fs.readFileSync('code.js')+'');

const test =
    jsc.forall("array nat", function(arr) {
        var a1 = JSON.parse(JSON.stringify(arr));
        var a2 = JSON.parse(JSON.stringify(arr));
        var count = permutationSort(a1);
        return count >= 0 && JSON.stringify(a1) ==
            JSON.stringify(a2.sort(function(a, b)
                { return a - b; }));
    });
```

This is optional, but highly encouraged. How exactly to get this code to run is left as an exercise to the reader. No points, but the satisfaction of knowing how to do property-based testing.