# Lab 3: Divide and Conquer

COSC 3020: Algorithms and Data Structures

Lars Kotthoff
larsko@uwyo.edu

25 September 2019

## Instructions

Attempt to finish the tasks below during the lab time. You have until Friday, 27 September 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 Divide and Conquer Sum

In the lectures, we've covered merge sort, which uses a divide-and-conquer approach to sort an array of values. There are many more algorithms that take such an approach. Implement a function that computes the sum of an array of integers using divide and conquer. The function should have the following signature:

```
function divideAndConquerSum(a);
```

where `a` is the array. The recursive calls sum up the numbers in the base case, and "merge" the sums of the recursive calls otherwise. For example, the return value for the array `a = [1,5,-1,4]` is `9`.

To make it a bit more interesting, instead of splitting into two sub-arrays like in merge sort, I want you to split into *three* sub-arrays at each divide step.

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

Hint: Like in the implementation of merge sort, you may need a helper function that does the actual recursion.

Total 5 points.

## 2 Runtime Analysis

What is the runtime of the algorithm that you implemented? Provide a recurrence relation for $T(n)$ as we did for merge sort (you can ignore constant

factors) and solve it as we did in the lectures. Give the final $\Theta$ complexity.

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

Total 5 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) {
        return JSON.stringify(divideAndConquerSum(arr))
           == JSON.stringify(arr.reduce(function(a, b) {
           return a + b; }, 0));
    });
jsc.check(test);
```

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.