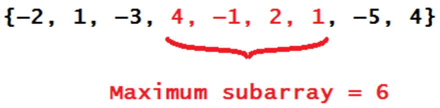
THE UNIVERSITY
OF THE WITWATERSRAND Lecturer: **Ritesh Ajoodha**

Laboratory 4: Project 1
Data Structures and Algorithms
Analysis of Algorithms
18 August , 2015
Due: 25 August, 2015

The **maximum subarray problem** is the defined as finding the largest sum of consecutive integers in an array. E.g.



```
{-2, 1, -3, 4, -1, 2, 1, -5, 4}

        Maximum subarray = 6
```

You are required to research **3 algorithms** that solve this problem:

1. The Brute force/Naïve Method
2. The Divide and Conquer Method
3. Kadane's Algorithm

There are two important submissions for this assignment: a code submission, where you simply submit all of your code you used for the analysis; and a writeup, where you state the details of the experiment. Your experiment writeup should address the following:

- The hardware used in the experiment (this includes the CPU, RAM, e.t.c.)
- The software tools used (this includes Gnuplot, G++, e.t.c.)
- The empirical and theoretical complexities of all 3 algorithms and state why they behave the way they do
- The main data structures used in each implementation
- An evaluation of input array sizes from 0- 30 000
- Each array should contain randomly generated values in the range -50 to 50. (use std::rand())
- a graphical comparison between the algorithms

Your written document should not be more than 1 page and bonus marks will be given for students who use LaTeX. One document should be submitted per group. Your program output should look something like this:

```
Input Size  Brute Force     Divide and Conquer  Kadane's Algorithm
101         2               1                   1
201         14              9                   2
301         76              13                  2
401         108             21                  3
501         190             55                  3
601         296             33                  3
701         641             49                  3
801         612             44                  3
901         825             53                  4
1001        1020            58                  4
1101        1256            65                  4
1201        1557            75                  5
1301        1365            81                  5
1401        2154            88                  6
...         ...             ...                 ...
33201       1.44158e+06     2811                79
33301       1.41839e+06     2894                81
33401       1.42892e+06     2539                95
33501       1.42674e+06     2755                81
33601       1.44132e+06     2768                81
33701       1.43997e+06     2777                77
33801       1.45012e+06     3154                97
33901       1.47716e+06     2586                109
34001       1.47672e+06     2803                75
34101       1.51964e+06     3231                102
34201       1.49281e+06     2928                81
34301       1.50697e+06     3035                90
```

When you plot the log-log graph using gnuplot. Your graphs should be very spiky, account for this 'spikiness' in your written document. You can smooth-out your graphs by calculating the average of every n consecutive values (the sliding window) in your time-series data. This is known as the **moving average** or rolling mean. Calculate the moving average for each algorithm's time-series and plot the new 'average' time-series data. Your graphs should be much smoother now, account for your choice of window size in your written document.

Additional Reading

1. Geeks for Geeks: Maximum Subarray Sum
2. Gnuplot Homepage
3. Wiki: Maximum subarray problem
4. Writing a Simple LaTeXDocument
5. Moving Average WIKI
6. Recursion
7. Getting Gnuplot on Xubuntu