CS325 programming assignment 2

Due on Feb 25th midnight

**Hand in Instructions**  You are encouraged to work in group of up to 3 people. Each group should have only one submission, which should clearly indicate all team members. Also you should clearly acknowledge any resource that you used in completing this assignment. Code and report should be submitted through the TEACH website.

**Maximum Subarray**  For this project, you will design, implement and analyze (both experimentally and mathematically) three algorithms for the maximum subarray problem:

Given an array of small integers $a[1, \ldots, n]$, compute

$$\max_{i \leq j} \sum_{k=i}^{j} a[k]$$

For example, $MaxSubarray([31, -41, 59, 26, -53, 58, 97, -93, -23, 84]) = 187$.

You may use any language to implement your algorithms.

**Instructions**  Please design and implement three algorithms described below:

1. **Brute force enumeration:** Loop over every possible pair of $i$ and $j$ values and compute $\sum_{k=i}^{j} a[k]$. Keep the best sum found so far. Note that if implemented naively, this can lead to many redundant computations. For example, $\sum_{k=i}^{j} a[k]$ can be computed by adding $a[j]$ to $\sum_{k=i}^{j-1} a[k]$ in constant time, instead of computing from scratch. Your implementation should take advantage of this and avoid repeated computation.

2. **Divide and Conquer:** If we split the array into two halves, we know that the maximum subarray will either be

   • fully contained in the first half, or
   • fully contained in the second half, or
   • consisted of a suffix of the first half and a prefix of the second half.

The first two cases can be found recursively. The last case can be found in linear time.
3. Dynamic programming. Develop your algorithm based on the following idea.

The maximum subarray of $a$ either uses the last element of $a$, or does not use it.

**Project Report.** In your report, you should include the following:

**Theoretical analysis** For each of the above algorithms, provide the following:

1. pseudocode for your algorithm.
2. asymptotic run time analysis for your algorithm.
3. theoretical correctness analysis of your algorithm. Provide a proof that your algorithm is correct. This is only needed for the divide and conquer, and dynamic programming algorithms.

**Empirical Testing the correctness of your algorithm.** A sample inputs/outputs file is provided for you to test the correctness of your algorithm. A final text input file will also be provided and your report should provide your answers to these inputs so that the TA can easily verify the correctness of your algorithm.

**Empirical analysis of run time.** Please run your algorithms on input arrays of size 1k, 2k, 3k, 4k, 5k, 6k, 7k, 9k, 10k respectively. To do this, you should generate your own random inputs use a random number generator provided by your programming language. For each size, you should generate 10 random inputs and run each algorithm on it and measure the running time of each algorithm. In your report, please plot the running time as a function of the input size. Include an additional plot of the running time in a log-log plot.

In your report, present and compare the empirical run time results of the three different algorithms. Provide a discussion of the comparative benefits and drawbacks of different algorithms.