

CS 325 Project 1: Maximum Sum Subarray

Your report must be typed and submitted online. Each team member's name must be listed as well as any resources used to finish the project.

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

Given an array of small integers $A[1, \dots, n]$, compute

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

For example, $\text{MAXSUBARRAY}([31, -41, 59, 26, -53, 58, 97, -93, -23, 84]) = 187$

Instructions

You may use any language you choose to implement your algorithms. All algorithms will take as input an array and output the subarray with a maximum sum along with the sum. Your algorithms are to be based on these ideas:

Algorithm 1: Enumeration. Loop over each pair of indices i, j and compute the sum $\sum_{k=i}^j A[k]$. Keep the best sum you have found so far.

Algorithm 2: Better Enumeration. Notice that in the previous algorithm the same sum is computed many times. In particular, notice that $\sum_{k=i}^j A[k]$ can be computed from $\sum_{k=i}^{j-1} A[k]$ in $O(1)$ time, rather than starting from scratch. Write a new version of the first algorithm that takes advantage of this observation.

Algorithm 3: Divide and Conquer. If we split the array into two halves, we know that the maximum subarray will either be

- Contained entirely in the first half,
- Contained entirely in the second half or
- Made of a suffix of the first half of the subarray and a prefix of the second half

Algorithm 4: Linear-time. Use the following ideas to develop a nonrecursive linear time algorithm. Start at the left end of the array and progress towards the right, keeping track of the maximum subarray sum seen so far. Knowing a maximum subarray of $A[1 \dots j]$, extend the answer to find a maximum subarray ending at index $j+1$ by using the following observation: a maximum subarray of $A[1 \dots j+1]$ is either a maximum subarray of $A[1 \dots j]$ or a subarray $A[i \dots j+1]$, for some $1 \leq i \leq j+1$. Determine a maximum subarray of the form $A[i \dots j+1]$ in constant time based on knowing a maximum subarray ending at index j .

CS 325 Project 1: Maximum Sum Subarray

Input and Output. Your program should read the input from the file `MSS_Problems.txt`. Output is to be written to `MSS_Results.txt`. The output from all four algorithms on all arrays should be in one file (there will be redundancy). We will run your program using `MSS_Problems.txt`. **Note:** All elements in the input arrays will be integers with at least one positive element in each array.

Testing for correctness. Above all else your algorithm should be correct. A file containing test sets will be posted. The file has one test case per line (10 cases). You may use this file to check if your code is correct. You should also test your code on self-generated instances. This is the format which your program should output its results. You will not submit results from these test cases.

Experimental analysis. For the experimental analysis you will plot running times as a function of input size. Every programming language should provide access to a clock (not necessarily in seconds). Run each of your algorithms on various input sizes for example $n = 100, 200, \dots, 900$ or $1000, 2000, \dots, 10000$. If your algorithm runs very fast and has a time of 0 you will need to increase the sizes of n . Likewise if the algorithm runs for over an hour you will need to decrease your input size. Each algorithm may require different sizes of n and you will need at least 10 different input sizes for each algorithm.

For each input size n and algorithm collect the running times for at least 10 different input arrays. To do this, generate random instances using a random number generator as provided by your programming language.

For example: Randomly generate ten input arrays of size 100. Run each input on each algorithm and record the times. For example

```
For i = 1:10
    A = random array 100 entries
    Start clock
    Algorithm1(A)
    pause clock
    return elapsed time
```

Note that you should not include the time to generate the instance or write the results to a file in your timing analysis.

Once all the data is collected.

- For each Algorithm, calculate the average running time for each n . Remember you have at least 10 values for each n .
- Plot the running times as a function of input size for each algorithm individually. Include all data values for each input size. Label your graphs (axes, title, etc). Fit a curve to your data? Is it linear/quadratic/logarithmic/exponential? You may find this video helpful:

CS 325 Project 1: Maximum Sum Subarray

- Create a log-log plot the running times of each algorithm http://en.wikipedia.org/wiki/Log-log_plot . Note that if the slope of a line in a loglog plot is m , then the line is of the form $O(x^m)$ on a linear plot. You may also find this video helpful: <http://www.khanacademy.org/math/algebra/logarithms/v/logarithmic-scale>

Project Report

Your report must include:

- **Names and Group Number:** List the names of all group members and the Group number at the top of the report.
- **Theoretical Run-time Analysis:** Give pseudo-code for each of the four algorithms and an analysis of the asymptotic running-times of the algorithms.
- **Proof of Correctness:** For Algorithm 3 write a proof of correction using induction.
- **Testing:** In order to get credit for this project, your code must produce correct responses. Test all algorithms against a test set.
- **Experimental Analysis:** Perform an experimental analysis and include plots as described above.
- **Extrapolation and interpretation:** Use the data from the experimental analysis to answer the following questions:
 - For each algorithm use the experimental data to estimate a function that models the relationship between running times and input sizes (n). Discuss any discrepancies between the experimental and theoretical running times.
 - For each algorithm, what is the size of the biggest instance that you can solve with your algorithm in one hour?

What to Submit

Your elected submitter must upload the following to Canvas and T.E.A.C.H.

- **Project Report** : PDF
- **Code:** Submit your fully functioning program which must run on our engr server FLIP for verification and a README with directions describing exactly how to use it, combined in a ZIP file.
- **Results:** Submit results for the MSS_Problems.txt in the form of a .txt file MSS_Results.txt