Assignment 2: Divide and Conquer (10%)

Winter 2024 – CPSC 413

Due at 23:59, Mar. 8

Assignment policy:

- This is an **individual** assignment, so the work you hand in must be your own. Any external sources used must be properly cited (see below).
- Submit your answers to the written questions as a **single PDF** to the assignment on Gradescope. Code files for any programming questions will need to be uploaded separately to the corresponding category on Gradescope.
- Extensions will not be granted to individual students. Requests on behalf of the entire class will only be considered if made more than 24h before the original deadline.
- All submissions must be clearly legible. Handwritten assignments will be accepted, but marks may deducted if the TAs cannot read your writing. It is recommended to use LaTeX to typeset your work.
- Some tips to avoid plagiarism in your assignments:

  1. Cite all sources for material you hand in that is not your original work. This applies to both proof-based and coding questions. You can put the citation into the submitted PDF or as a comment in your code. For example, if you find and use code found on a website, include a comment that says, for example:

     ```
     # the following code is from https://www.quackit.com/python/tutorial/python_hello_world.cfm.
     ```

     Use the complete URL so that the marker can check the source.
  2. Citing sources avoids accusations of plagiarism and penalties for academic misconduct. However, you may still get a low grade if you submit work that is not primarily your own.
  3. Discuss and share ideas with other students as much as you like, but make sure that when you write your assignment, it is your own work. A good rule of thumb is to wait 20 minutes after talking with somebody before writing it down. If you exchange written material with another student, take notes while discussing with a fellow student, or copy from another person's screen, then this work is not yours.
  4. We will be looking for plagiarism in all submissions, possibly using automated software designed for the task.
  5. Remember, if you are having trouble with an assignment, it is always better to go to your TA and/or instructor to get help than it is to plagiarize.

Questions:

1. (2 marks) Solve each of the following recurrences by drawing the recurrence tree:

   (a) $T(n) = 2T(\frac{n}{6}) + O(n)$

   (b) $T(n) = T(n-1) + O(n)$

   (c) $T(n) = 2T(\sqrt{n}) + O(\log n)$

2. (2 marks) Consider the task of computing the product $C = AB$, where A, B, and C are $n \times n$ matrices. Solving this using the standard high school approach takes $\Theta(n^3)$ multiplications to compute the entries in the output matrix C.

   As discussed in class, the matrix multiplication be done more efficiently using Strassen's algorithm, which performs the computation in $\Theta(n^{\log_2(7)})$ multiplications. Strassen's algorithm is based on the discovery that two 2×2 matrices can be multiplied using only seven multiplications (done by performing some extra additions and subtractions).

   Suppose your three friends have spent their weekend coming up with new matrix multiplication shortcuts. Dylan has found a way to multiply two 68×68 matrices using $132,464$ multiplications. Etienne has found a way to multiply two 70×70 matrices using $143,640$ multiplications. Francesca has found a way to multiply two 72×72 matrices using $155,424$ multiplications. If they each apply their method to give a divide-and-conquer algorithm for matrix multiplication like Strassen's algorithm, who has the fastest matrix multiplication algorithm? How do they compare to Strassen's algorithm?

3. **Median-finding (two arrays):** Suppose you are given two sorted arrays of integers, A and B, each with length n. You wish to find the median of the combined array, i.e. the integer that would be in the n^{th} position if the arrays A and B were combined and sorted. For convenience, you may assume that n is odd.

   For example, the median element for the arrays below with $n = 5$ would be 4, as the sorted array containing all elements is $[1, 2, 2, 3, 4, 5, 6, 6, 7, 8]$, which has 4 as the fifth element.

   $$A = [1, 2, 4, 6, 7]$$
   $$B = [2, 3, 5, 6, 8]$$

   (a) (3 marks) Give a divide-and-conquer algorithm that finds the median element in $O(\log n)$ steps (note the comparison to your two answers on Quiz 6!). Explain briefly why your algorithm is correct.

   (b) (1 mark) Step through your algorithm on the following input arrays, with $n = 5$. Show the arrays your algorithm recurses on at each step.

   $$A = [1, 1, 3, 4, 8]$$
   $$B = [1, 2, 5, 6, 9].$$

4. (2 marks) Write a Python 3 function that, given an array of integers and two integer arguments `start` and `end`, prints the maximum possible sum of a subarray that lies between indices `start` and `end` inclusive. You do not need to return the indices of the subarray that gives this maximum sum, just the value of the sum itself. Your code **must** be written using recursion and the algorithm we covered in class. Do not optimize the code by removing the recursive calls.

   Your function should be contained in the Python 3 starter class provided on D2L. This class has a constructor with an argument named `array`, which will be the array of integers you need to process. All values will be integers between 0 and $99,999$.

   Upload your `array_checker.py` to the correct location on Gradescope. Make sure your code is clean and well-commented.

Autograder notes:

Your code will be graded by an autograder on Gradescope. For your code to work with the autograder, you will need to use the template class `array_checker.py` provided on D2L. You should write your code to fill in the `check_array()` function **without changing the arguments**, and you must return (not print) an integer indicating the maximum possible subarray sum. You do not need to change the rest of the class. Do not change the filename, and do not upload any files other than your `array_checker.py`.

You can test your code with the autograder as many times as you like before submitting. It is also recommended that you add print statements for debugging during your own initial development. There is a set of test cases included in `a2test.py`, provided on D2L, that you can use to test your code locally if you prefer.

Note that marks will be allocated to your code structure and commenting, so pay attention to this too.

---

author: jcleahy@ucalgary.ca