Programming Assignment #1

TOTAL POINTS 1

1. In this programming assignment you will implement one or more of the integer multiplication algorithms described in lecture. 1 point

   To get the most out of this assignment, your program should restrict itself to multiplying only pairs of single-digit numbers. You can implement the grade-school algorithm if you want, but to get the most out of the assignment you'll want to implement recursive integer multiplication and/or Karatsuba's algorithm.

   So: what's the product of the following two 64-digit numbers?

   3141592653589793238462643383279502884197169399375105820974944592

   2718281828459045235360287471352662497757247093699959574966967627

   [TIP: before submitting, first test the correctness of your program on some small test cases of your own devising. Then post your best test cases to the discussion forums to help your fellow students!]

   [Food for thought: the number of digits in each input number is a power of 2. Does this make your life easier? Does it depend on which algorithm you're implementing?]

   The numeric answer should be typed in the space below. So if your answer is 1198233847, then just type 1198233847 in the space provided without any space / commas / any other punctuation marks.

   (We do not require you to submit your code, so feel free to use any programming language you want --- just type the final numeric answer in the following space.)

   Enter answer here

I, **Veinstin Furtado**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Save Submit

Programming Assignment #2

TOTAL POINTS 1

1. Download the following text file: 1 point

IntegerArray.txt

This file contains all of the 100,000 integers between 1 and 100,000 (inclusive) in some order, with no integer repeated.

Your task is to compute the number of inversions in the file given, where the i^{th} row of the file indicates the i^{th} entry of an array.

Because of the large size of this array, you should implement the fast divide-and-conquer algorithm covered in the video lectures.

The numeric answer for the given input file should be typed in the space below.

So if your answer is 1198233847, then just type 1198233847 in the space provided without any space / commas / any other punctuation marks. You can make up to 5 attempts, and we'll use the best one for grading.

(We do not require you to submit your code, so feel free to use any programming language you want --- just type the final numeric answer in the following space.)

[TIP: before submitting, first test the correctness of your program on some small test files or your own devising. Then post your best test cases to the discussion forums to help your fellow students!]

Enter answer here

I, **Veinstin Furtado**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Save Submit

Programming Assignment #3

TOTAL POINTS 3

1. GENERAL DIRECTIONS: 1 point

Download the following text file:

QuickSort.txt

The file contains all of the integers between 1 and 10,000 (inclusive, with no repeats) in unsorted order. The integer in the i^{th} row of the file gives you the i^{th} entry of an input array.

Your task is to compute the total number of comparisons used to sort the given input file by QuickSort. As you know, the number of comparisons depends on which elements are chosen as pivots, so we'll ask you to explore three different pivoting rules.

You should not count comparisons one-by-one. Rather, when there is a recursive call on a subarray of length m, you should simply add $m-1$ to your running total of comparisons. (This is because the pivot element is compared to each of the other $m-1$ elements in the subarray in this recursive call.)

WARNING: The Partition subroutine can be implemented in several different ways, and different implementations can give you differing numbers of comparisons. For this problem, you should implement the Partition subroutine *exactly* as it is described in the video lectures (otherwise you might get the wrong answer).

DIRECTIONS FOR THIS PROBLEM:

For the first part of the programming assignment, you should always use the first element of the array as the pivot element.

HOW TO GIVE US YOUR ANSWER:

Type the numeric answer in the space provided.

So if your answer is 1198233847, then just type 1198233847 in the space provided without any space / commas / other punctuation marks. You have 5 attempts to get the correct answer.

(We do not require you to submit your code, so feel free to use the programming language of your choice, just type the numeric answer in the following space.)

Enter answer here

2. GENERAL DIRECTIONS AND HOW TO GIVE US YOUR ANSWER: 1 point

See the first question.

DIRECTIONS FOR THIS PROBLEM:

Compute the number of comparisons (as in Problem 1), always using the final element of the given array as the pivot element. Again, be sure to implement the Partition subroutine *exactly* as it is described in the video lectures.

Recall from the lectures that, just before the main Partition subroutine, you should exchange the pivot element (i.e., the last element) with the first element.

Enter answer here

3. GENERAL DIRECTIONS AND HOW TO GIVE US YOUR ANSWER: 1 point

See the first question.

DIRECTIONS FOR THIS PROBLEM:

Compute the number of comparisons (as in Problem 1), using the "median-of-three" pivot rule. [The primary motivation behind this rule is to do a little bit of extra work to get much better performance on input arrays that are nearly sorted or reverse sorted.] In more detail, you should choose the pivot as follows. Consider the first, middle, and final elements of the given array. (If the array has odd length it should be clear what the "middle" element is; for an array with even length $2k$, use the k^{th} element as the "middle" element. So for the array 4 5 6 7, the "middle" element is the second one ---- 5 and not 6!) Identify which of these three elements is the median (i.e., the one whose value is in between the other two), and use this as your pivot. As discussed in the first and second parts of this programming assignment, be sure to implement Partition *exactly* as described in the video lectures (including exchanging the pivot element with the first element just before the main Partition subroutine).

EXAMPLE: For the input array 8 2 4 5 7 1 you would consider the first (8), middle (4), and last (1) elements; since 4 is the median of the set {1,4,8}, you would use 4 as your pivot element.

SUBTLE POINT: A careful analysis would keep track of the comparisons made in identifying the median of the three candidate elements. You should NOT do this. That is, as in the previous two problems, you should simply add $m - 1$ to your running total of comparisons every time you recurse on a subarray with length m.

Enter answer here

I, **Veinstin Furtado**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Save Submit

Programming Assignment #4

TOTAL POINTS 1

1. Download the following text file: 1 point

kargerMinCut.txt

The file contains the adjacency list representation of a simple undirected graph. There are 200 vertices labeled 1 to 200. The first column in the file represents the vertex label, and the particular row (other entries except the first column) tells all the vertices that the vertex is adjacent to. So for example, the 6^{th} row looks like : "6 155 56 52 120 .....". This just means that the vertex with label 6 is adjacent to (i.e., shares an edge with) the vertices with labels 155,56,52,120,.....,etc

Your task is to code up and run the randomized contraction algorithm for the min cut problem and use it on the above graph to compute the min cut. (HINT: Note that you'll have to figure out an implementation of edge contractions. Initially, you might want to do this naively, creating a new graph from the old every time there's an edge contraction. But you should also think about more efficient implementations.) (WARNING: As per the video lectures, please make sure to run the algorithm many times with different random seeds, and remember the smallest cut that you ever find.) Write your numeric answer in the space provided. So e.g., if your answer is 5, just type 5 in the space provided.

Enter answer here

I, **Veinstin Furtado**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Save Submit