CS180 Exam 1

Rajiv Anisetti

TOTAL POINTS

20.85 / 22

QUESTION 1

Problem 1 8 pts

1.1 Asymptotic notation 0.6 / 1

- **0 pts** 6 out of 6
- **0.25 pts** 5 out of 6
- ✓ **- 0.4 pts** **4 out of 6**
- **0.5 pts** 3 out of 6
- **0.6 pts** 2 out of 6
- **0.75 pts** 1 out of 6
- **0.75 pts** 0 out of 6

1.2 True or False: DC 1 / 1

- ✓ **- 0 pts** **Correct**
- **0.4 pts** Wrong answer but correct formula formed
- **0.5 pts** Wrong answer with wrong formula
- **0 pts** Correct but wrong explanation

1.3 Principles of DC 1 / 1

- ✓ **- 0 pts** **Correct**
- **0.4 pts** divide not mentioned
- **0.4 pts** merge not mentioned

1.4 Solving recurrence 1 / 1

- ✓ **- 0 pts** **used master theorem**
- **0 pts** Used expansion
- **0.5 pts** wrote the master theorem components but wrong reasoning
- **0.75 pts** master theorem components are wrong
- **0.5 pts** used expansion but wrong answer
- **0.75 pts** wrong attempt for expansion

1.5 Karatsuba trick 1 / 1

- ✓ **- 0 pts** **Correct**
- **0.5 pts** wrong formation of trick
- **0.75 pts** no usage of trick at all

1.6 List vs Matrix representations 1 / 1

- ✓ **- 0 pts** **Correct**
- **0.5 pts** no mention of space
- **0.5 pts** no mention of edge access time
- **0.75 pts** missing considerations of space and edge access times

1.7 Definition of path 1 / 1

- ✓ **- 0 pts** **Correct**
- **0.5 pts** Incorrect definition / not generic

1.8 Checking if graph is connected 1 / 1

- ✓ **- 0 pts** **Correct**
- **0.7 pts** Wrong Answer
- **0.5 pts** Did not check if all vertices are discovered
- **0.5 pts** Did not check if all vertices are connected/discovered. Just checked one.

QUESTION 2

2 Sorting sorted arrays 4 / 4

- ✓ **- 0 pts** **Correct**
- **1.5 pts** using mergesort to combine 2 sorted arrays. Gives runtime O((nk) log(nk)) - more than allowed.
- **1 pts** unclear merge step
- **1.5 pts** heap ops should be stated and clarified as these were not covered in class.
- **1.75 pts** reasonable attempt but missing crucial details and/or not correct.
- **2.25 pts** Missing crucial details and/or not correct.
- **3 pts** attempt something relevent
- **3.5 pts** attempt something irrelevent
- **4 pts** empty
- **3 pts** Solution runs in time O(n k^2) time, much more than the O(nk log k) the problem was looking for.

QUESTION 3

3 Finding plurality elements 4 / 4

- ✓ **- 0 pts** **Correct**

- **0.5 pts** no base case
- **1.5 pts** no/wrong run-time analysis or no recurrence relation of the time complexity
- **1.5 pts** no/wrong counting of returned elements from the recursion in the merge part
- **1.75 pts** reasonable attempt but not returning all plurality elements
- **2.25 pts** reasonable attempt with an algorithm running in time O(n^2) or worse.
- **2.5 pts** attempt missing many details and not correct.
- **3.25 pts** not a reasonable attempt
- **4 pts** no answer

QUESTION 4

4 Closest pair L4-distance 4 / 4

✓ - **0 pts** **Correct**

- **0 pts** You check way too many points for S_y and didn't show how you derived the number. Try to simplify your strip construction./ Or show how you derive this number
- **2.25 pts** reasonable attempt but missing many crucial details and/or not correct.
- **2.5 pts** moderate attempt but missing many crucial details and/or not correct.
- **1.5 pts** Didn't state how to compute/how to organize the points in the strip S. (for example, "sort by y coordinate" or including which points in strip or the width/height of grid) or Wrong way to construct the strip and grid.
- **1.5 pts** Didn't mention how many points to look up for each S_y in the strip
- **1.5 pts** Didn't Identify the divide-conquer high-level steps correctly
- **4 pts** No answer
- **1.5 pts** wrong number of points to look up

QUESTION 5

5 BFS trace 1.25 / 2

- **0 pts** Correct
- **1 pts** Extra lists than needed (You have mostly not considered the edges {4,6} {5,6} in line 2 of the Question)
- **1 pts** Extra lists than needed
- **1 pts** L[2] has extra elements

✓ - **0.75 pts** **L[2] order of elements wrong**

- **0.5 pts** L[1] order of elements wrong

gradescope

Exam 1. April 25, 2018

CS180: Algorithms and Complexity
Spring 2018

Guidelines:

- The exam is closed book and closed notes. Do not open the exam until instructed to do so.
- Write your solutions clearly and when asked to do so, provide complete proofs. You may use results and algorithms from class without proofs or details **except for Problem 4** as long as you state what you are using.
- I recommend taking a quick look at all the questions first and then deciding what order to tackle to them in. Even if you don't solve the problems fully, attempts that show some understanding of the questions and relevant topics will get reasonable partial credit.
- You can use extra sheets for scratch work, but you can only use the white space (it should be more than enough) on the exam sheets for your final solutions.
- Most importantly, make sure you adhere to the policies for academic honesty set out on the course webpage. The policies will be enforced strictly and any cheating reported with the score automatically becoming zero.
- Write clearly and legibly. All the best!

Problem	Points	Maximum
1		8
2		4
3		4
4		4
5		2
Total		22

Name	Rajiv Aniseth
UID	904801422
Section	1F

1 Problem

The answers to the following should fit in the white space below the question.

1. For each pair (f, g) below indicate the relation between them in terms of O, Ω, Θ. For each missing entry, write-down Y (for YES) or N (for NO) to indicate whether the relation holds (no need to justify your answers here). For example, if $f = O(g)$ but not $\Omega(g)$, then you should enter Y in the first box and N in the other two boxes. Similarly, if $f = \Theta(g)$, then you should enter Y in all the boxes. [1 point]

log

f	g	O	Ω	Θ
n^2	$n^2 - 2n + 2$	N	Y	N
$\log_2 n$	$(\log_{100} n)^2$	Y	N	N

2. Is the following True or False: Consider a divide and conquer algorithm which solves a problem on an instance of length n by making six recursive calls to instances of length $\lfloor n/3 \rfloor$ each, and combines the answers in $O(n^2)$ time. Then, the time-complexity of the algorithm is $O(n^2)$. [1 point]

$T(n) = 6T(\frac{n}{3}) + O(n^2)$ $\quad 1 \leq \log_3 6 \leq 2$

True

3. State the principles behind the divide and conquer technique for designing algorithms. [1 point]

Divide the problem into subproblems, recursively solve each subproblem, combine the solutions to create an overall solution.

4. What is the solution to the recurrence $T(1) = 1$, $T(n) = 2T(n/2) + 10n$? [1 point]

$\log_2 2 = 1$

$\therefore$ $O(T(n) = O(n \log n)$

5. Let a_0, a_1, b_0, b_1 be four integers that are k bits long. Write down Karatsuba's trick (that we used in class for fast integer multiplication) to compute the four products $a_1 \cdot b_1, a_1 \cdot b_0, a_0 \cdot b_1, a_0 \cdot b_0$ using only three multiplications and some additions and subtractions.

Compute $a_1 \cdot b_1$ & $a_0 \cdot b_0$

Compute $a_0 + a_1$ & $b_0 + b_1$

Compute $(a_0 + a_1) \cdot (b_0 + b_1)$

Compute $a_0 b_1 + a_1 b_0 = (a_0 + a_1) \cdot (b_0 + b_1) - a_1 b_1 - a_0 b_0$

6. Write down some pros and cons of the adjacency-list and adjacency-matrix representations of graphs. [1 point]

Adjacency List

Pros: $O(|V| + |E|)$ space

Cons: $O(|V|)$ to check if edge is present between two vertices

Adjacency Matrix

Pros: $O(1)$ time to check for edge presence

Cons: $O(|V|^2)$ space to represent

7. Write down the definition of a path in a graph $G = (V, E)$. [1 point]

A path is a set of vertices where each vertex in the set is connected to the vertex prior to it in the set by an edge, thus all the vertices in the set are connected.

8. How can we efficiently check if a graph given in adjacency-list representation is connected? (You can refer to algorithms done in class without writing them out fully.) [1 point]

We can run BFS on the graph from any point and check if discovered[u] = true for all points $u \in V$

2 Problem

You are given k sorted arrays, each with n numbers in them. Give an algorithm for merging these arrays into a single sorted array of numbers that runs in time $O(nk \log k)$. You don't have to analyze the running time or prove correctness. [4 points]

(You can assume that the solution to the following recurrence is $O(nk \log k)$: $T(1) = O(1)$, $T(k) \leq 2T(k/2) + O(n \cdot k)$.)

If $k == 1$, merge sort

First, divide the k arrays into two sets of arrays

k_l = first $\lceil \frac{k}{2} \rceil$ arrays

k_r = second $\lfloor \frac{k}{2} \rfloor$ arrays

Recursively run algorithm on k_l & k_r

Perform a merging operation similar to merge sort where you keep track of the smallest element of k_l and k_r (maybe using a counter) to compare the smallest of both arrays in order to know which element to add first to the sorted array

Return sorted array from previous step

[2,2,1|1,6,6]
[1,2,3|4,5,6]

[1|1,1,1]

[5,5|2,4,2|2,1|1|6,1,6]

[1,1,2,4,2|2,1,2|1,2]

3 Problem

Given an array $A[0,1,\ldots,n-1]$, an element $A[i]$ is said to be a *plurality element* if more than $\lfloor n/3 \rfloor$ of its elements equal elements of A. For example, the array $A = [1,11,2,4|2,2,|1,2,4]$ has one plurality element 2; the array $A = [1,1,2|4,2,|2,1,2,1]$ has two plurality elements $1,2$; the array $A = [1,11,|2,1,2,|1,11,|2,11]$ has no plurality elements.

Given an array as input, the task is to design an efficient algorithm to tell whether the array has any plurality elements and, if so, to find **all** the plurality elements. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form "is $A[i] > A[j]$?". (Think of the array elements as mp3 files, say; so in particular, you cannot sort the elements.) However you can answer questions of the form: "is $A[i] = A[j]$" in constant time.

Give an algorithm to solve the problem. For full-credit, your algorithm should be correct and run in time $O(n \log n)$ and you should bound the run-time of the algorithm. (You don't have to prove correctness.). [4 points]

Find-Plurality Element (A)

- if $n \leq 3$, brute force search for elements → $O(1)$
- Divide the array into 2 arrays A_l, A_r where A_l = first $\lfloor \frac{n}{2} \rfloor$ and A_r = last $\lceil \frac{n}{2} \rceil$ elements } $O(n)$
- of A
- Array 1 = Find-Plurality Element (A_l) } $2T(\frac{n}{2})$
- Array 2 = Find-Plurality Element (A_r)
- Let Array 3 = Distinct elements from Arrays 1 and 2 } $O(n)$
- Go through array A keeping count of when A[i] = any of elements in Array 3 and increment their counters } $O(n)$
- If any or all of these counters $\geq \lfloor \frac{n}{3} \rfloor$ return their respective elements } $O(1)$

Runtime → $T(n) = 2T(\frac{n}{2}) + O(n)$ → $O(n \log n)$

4 Problem

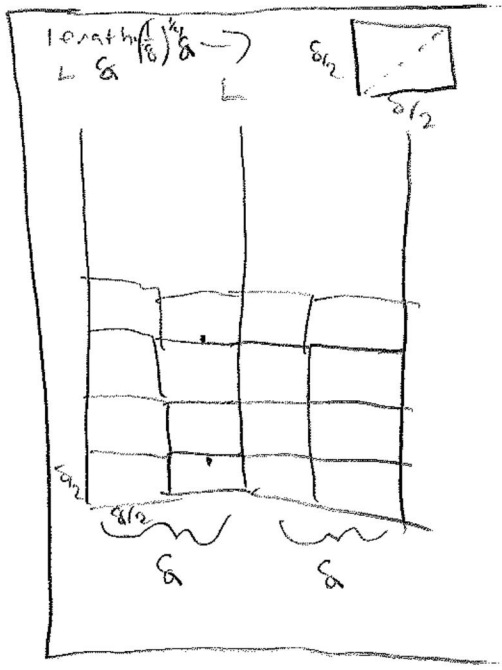
Given a set of points $P = \{p_1, \ldots, p_n\}$ in the plane, give an algorithm for finding a pair of points with the smallest possible L4-distance among the points where L4-distance between two points is defined by $d_4((x,y),(x',y')) = (|x-x'|^4 + |y-y'|^4)^{1/4}$.

For full-credit your algorithm should be correct and run in time $O(n \log n)$. You don't have to prove correctness or analyze the run-time of the algorithm. You should describe all the steps in the algorithm at a level of detail similar to what was done in class (however, you don't have to describe how to manipulate the sorted lists). [4 points]

Find-Smallest-Dist

First, get sorted lists P_x & P_y with respect to x and y coordinates

Recursive-Find (P_x, P_y)



Recursive-Find (P_x, P_y)

IF $|P| \leq 3$, brute force search

Create Q = first $\lceil \frac{n}{2} \rceil$ points in P_x

Create R = last $\lceil \frac{n}{2} \rceil$ points in P_x

Create Q_x, Q_y from P_x, P_y

Create R_x, R_y from P_x, P_y

(a,b) = Recursive-Find (Q_x, Q_y)

These are points → (c,d) = Recursive-Find (R_x, R_y)

δ = min (L4-dist (a,b), L4-dist (c,d))

Set x^* = Final point in Q_x

Set Line $L = x^*$

Create set S of points $\leq \delta$ away from the line L w/ respect to X-coordinate

Create S_y from P_y, where S_y is a sorted S w/ respect to y-coordinate

For each point u in S_y, find distance between u and next 15 points in S_y. Let d be the minimum distance found by this, and (e,f) be the points with this distance

IF $d \leq \delta$

return (e,f)

else if (L4-dist (a,b) < L4-dist (c,d))

return (a,b)

else

return (c,d)

5 Problem

Let $G = (V, E)$, where $V = \{1, 2, 3, 4, 5, 6\}$ and $E = \{\{1, 2\}, \{1, 6\}, \{2, 5\}, \{2, 6\}, \{3, 4\}, \{3, 5\}, \{3, 6\}, \{4, 6\}, \{5, 6\}\}$. Suppose that G was given to you in adjacency list representation where the elements in the adjacency list are ordered in increasing order. For example, the adjacency list of vertex 2 would be $[1, 5, 6]$. Run the BFS algorithm on G starting from the vertex 1. It suffices to show the step-by-step evolution of the lists $L[0], L[1], \ldots$ as we described in class. [2 points]

$L[0] = \{1\}$ Discovered = {T, F, F, F, F, F}

$L[1] = \{2, 6\}$ Discovered = {T, T, F, F, F, T}

$L[2] = \{3, 4, 5\}$ Discovered = {T, T, T, T, T, T}

$L[3] = \emptyset$ Discovered = {T, T, T, T, T, T}