CS180 Exam 1

TOTAL POINTS

22 / 22

QUESTION 1

Problem 1 8 pts

1.1 Asymptotic notation **1 / 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 2 / 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

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	[redacted]
UID	[redacted]
Section	Friday 12PM 1E

$\frac{\log_2 n}{(\log_{100} n)^2}$ $\frac{n^2}{n^2-2n+2}$ $\frac{2n}{2n-2}$ $\frac{2}{2} = 1$

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]

$100n^2 - 100n + 100$ n^2

$(n-1)^2+1$

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

$T(n) = 6T(n/3) + O(n^2)$

$\log_3 6 < 2$, $T(n) = O(n^2)$

True.

~~2×24+16~~

~~44+40~~ ~~160+80~~

~~= 84~~

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

Divide the original problem into smaller sub-problems

Recursively solve the smaller subproblems

Combine the solutions for the sub-problems into the Solution of the whole problem.

If the problem size is smaller than a constant, use brute-force method $\boxed{O(1)}$ (terminal vertices on the recursion tree)

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

~~$T(1)=1$~~

~~$T(2)=2\times1+20=22$~~

~~$T(4)=2\times22+40=84$~~

~~$T(8)=2\times84+80$~~

~~$=2(2(2\times1+10\times2)+10\times4)+10\times8$~~

~~$\Rightarrow T(n) = n + \log(2n+n)$~~

$T(1) = O(1)$

$T(n) = 2T(n/2) + O(n)$

$\log_2 2 = 1$

$\Rightarrow T(n) = O(n \log n)$.

$a_1 a_0$

$b_1 b_0$

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.

$$2^n (a_1 \cdot b_1) + 2^{n/2}(a_1 \cdot b_0 + a_0 \cdot b_1) + a_0 \cdot b_0$$

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

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

=> First compute $a_1 \cdot b_1$, $a_0 \cdot b_0$ and ~~$(a_1 \cdot b_0 + a_0 \cdot b_1)$~~ $(a_1 + a_0) \cdot (b_1 + b_0)$

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

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

Adjacency matrix :

~~Cons:~~

Pros: Fast, $O(1)$ time for getting an edge
Good for dense graphs
Easy to write code

Cons : Hard to add new vertices
Takes $O(n^2)$ Space to store
~~For undirected graph, each edge is stored twice~~

Adjacency list :

Pros :
Saves Space.
takes up $O(|V| + |E|)$ space
Easy to expand
Good for sparse graphs

Cons :
$O(|E|)$ time to access an edge
Not intuitive

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

A path is an ordered list of vertices

$\{V_1, V_2, \ldots, V_K\}$

Such that $V_i \in V$ for $i = 1$ to K and $(V_i, V_{i+1}) \in E$ for $i = 1$ to $K-1$

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]

Use breadth-first search algorithm.

Run the BFS algorithm on the graph.

Iterate through the ~~After~~ DISCOVERED array after ~~the~~ the search ends.

If all vertices were visited (no DISCOVERED[i] == False) then the graph is connected. Otherwise, it's not connected.

The time complexity is $O(|V| + |E|)$.

~~$2T(n/2)$~~

$2T(k/2)+O(nk)$

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 $O(nk\log k)$: $T(1) = O(1)$, $T(k) \le 2T(k/2) + O(n\cdot k)$.)

Procedure merge2arrays : (takes in two sorted arrays A_1, A_2, returns the merged sorted array A_m)

```
~~i=0, j=0~~
~~while~~
i=1 , j=1 , Am=[] (empty array)
while $i \le |A_1|$ or $j \le |A_2|$ :
    If $i > |A_1|$ : Append $A_2[j]$ to Am, $j=j+1$
    Else if $j > |A_2|$ : Append $A_1[i]$ to Am, $i=i+1$
    Else if $A_1[i] > A_2[j]$ : Append $A_2[j]$ to Am, $j=j+1$
    Else : Append $A_1[i]$ to Am, $i=i+1$
Return Am
```

Algorithm : Let L be the list of the k arrays

~~If k=1, return~~

If $k=1$, return $L[1]$.

If $k=2$, return merge2arrays $(L[1], L[2])$

Otherwise, divide L to two equal-size sublists L_1 and L_2

Recursively find the merged sorted array of L_1 and L_2, let them be A_1 and A_2 respectively.

Use merge2arrays(A_1, A_2) to merge A_1 and A_2 and this returns the single sorted array of L.

3 Problem

Given an array $A[0, 1, \ldots, n-1]$, an element $A[i]$ is said to be a *pluraility 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]

If $n \leq 3$, use brute force search ($O(1)$ time)
(return the element that appears twice or above)
Divide array A into two equal size subarrays A_l and A_r
Recursively find the pluraility elements in A_l and A_r. Let the answers be P_l and P_r, respectively.

[Because of the Pigeon-hole theorem, an array may have at most 3 plurality elements, so $|P_l| \leq 3$ and $|P_r| \leq 3$.

Let P_A be an empty array.

~~For $x \in P_l$~~
For each element x in both P_l and P_r:
    Iterate through A and count the number of appearance of x (let it be C_x)
    If $C_x > \lfloor n/3 \rfloor$, add x to P_A

P_A is the list of pluraility elements of A.

Because $|P_l| + |P_r| \leq 6$, the merging step accesses A for at most $6 \cdot n$ times, ~~the~~ merging the sub-solutions takes $6 \cdot n = O(n)$ time.

$T(n) = 2T(n/2) + O(n)$

$\log_2 2 = 1 \implies T(n) = O(n \cdot \log n)$

$2^{\frac{1}{4}}$ $\frac{1}{2^4}$ $\frac{1}{(x^4)^4}$

x^{16}

$(x^4)^{\frac{1}{4}}$

$\frac{1}{x^{16}}$

$\left(\frac{1}{x^4}+\frac{1}{x^4}\right)^{\frac{1}{4}} \geq \frac{1}{x}$

$\left(\frac{2}{x^4}\right)^{\frac{1}{4}} \leq \frac{1}{x}$ $\frac{x^{16}}{16} \leq \frac{1}{x}$

$\frac{\frac{1}{16}}{(x^4)^{\frac{1}{4}}} \leq \frac{1}{x}$ $x^{17} \geq 16$

$x^{17} \geq 16$

$x \geq \sqrt[17]{16}$

(x^4+x^4)

$(2x^4)^{\frac{1}{4}} \leq x$ $\frac{2}{\sqrt[17]{16}} \times \frac{1}{\sqrt[17]{16}}$

$\frac{1}{16}(x^{16}) \leq x$ $\frac{2}{\frac{1}{\sqrt[17]{16}}}$

~~$x^{16} \leq 16x$~~

~~$x^{15} \leq 16$~~

len: $\frac{2}{\frac{1}{2}}$ $\frac{2}{\frac{1}{\sqrt[17]{16}}}$

$\frac{1}{x^{16}} \leq 16x$ x 2

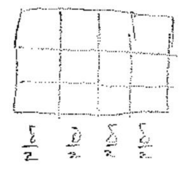
$\frac{1}{x^{17}} \geq 16$ $\frac{1}{16}$ $\frac{1}{16}$

$\frac{1}{16x} \leq \frac{1}{x}$ x $\frac{1}{16}$

$x \square$ $\left(\frac{2}{x^4}\right)^{\frac{1}{4}} \geq \frac{1}{x}$

$(2x^4)^{\frac{1}{4}}$ $\frac{\frac{1}{16}}{x} \geq \frac{1}{x}$

$\frac{1}{16}x$ $\frac{32}{128}$



$\frac{1}{16}+\frac{1}{16}$ $\frac{1}{16}^{1/4}$

$(\frac{1}{8})^{1/4}$

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]

If $|P| \leq 3$, use brute-force search ($O(1)$ time)

Otherwise, sort P according to the x-coordinates of each point

Divide P into two equal-size subarrays P_L and P_R that contains the first half and the second half of points, sorted by x-coordinates.

Recursively find the closest point pair in P_L, let it be $\{P_{L1}, P_{L2}\}$

Recursively find the closest point pair in P_R, let it be $\{P_{R1}, P_{R2}\}$

~~[illegible]~~ $\delta_L = d_4(P_{L1}, P_{L2})$ $\delta_R = d_4(P_{R1}, P_{R2})$ $\delta = \min(\delta_L, \delta_R)$

Let $L_S = [\,]$ (empty array). ~~For~~ For each point ~~[illegible]~~, $P_i \in P$, if $|d_4(P_i, P[\frac{|P|}{2}])| \leq \delta$, then add P_i to L_S.

point in the middle of the ~~[illegible]~~ x-axis

Sort L_S according to each point's y-coordinate. ~~[illegible]~~

For each point $P_i \in L_S$, consider the next ~~[illegible]~~ points

Calculate ~~L4~~ distance between P_i and that point. (using d_4)

Let δ_S be the minimum L4 distance of the points pair considered ~~[illegible]~~ in the last step. Let $\{P_{S1}, P_{S2}\}$ be that point pair.

If $\delta_S < \delta$, ~~return [illegible] δ~~ return $\{P_{S1}, P_{S2}\}$

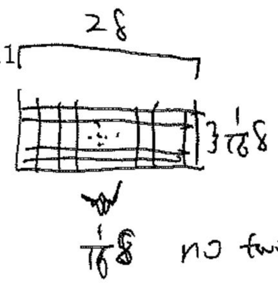
Else if $\delta_L < \delta_R$, return $\{P_{L1}, P_{L2}\}$

Else return $\{P_{R1}, P_{R2}\}$.

Consider the next $\frac{2}{\frac{1}{16}} \times$ ~~[illegible]~~ 32×4

$= 32^2 \times 4$ points ~~[illegible]~~

$= 4 \times 32^2$ points



no two points in the same grid.

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]

V	adj[V]
1	2, 6
2	1, 5, 6
3	4, 5, 6
4	3, 6
5	2, 3, 6
6	1, 2, 3, 4, 5

i	L[i]	DISCOVERED
0	[1]	[T, F, F, F, F, F]
1	[2, 6]	[T, T, F, F, F, T]
2	[5, 3, 4]	[T, T, T, T, T, T]
3	ϕ [illegible]	