CS180 Exam 1

17.5 / 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

- ts missing considerations of space and ess times

ion of path 1 / 1

Correct

s 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 1 / 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 2.5 / 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)
  - **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

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

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

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]

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]

$a = 6$ $b = 3$ $f(n) = n^2$ $\log_b a = \log_3(6)$

$T(n) \leq 6T(\frac{n}{3}) + n^2$ $f(n) = n^{k+\varepsilon}$

$T(n) = O(f(n))$

$3^k = 6 < 3^2$

$3^k < 3^2$

$k < 2$

$\boxed{T(n) = O(n^2)}$ $\boxed{\text{TRUE}}$

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

The general technique behind divide and conquer involves splitting the main problem into subproblems that are smaller, solving those subproblems recursively with a base case, and then merging / combining the solutions of the subproblems to get the final answer

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

$a = 2$ $b = 2$ $k = \log_b a = \log_2 2 = 1$

$T(n) = n^k \log n$

$\boxed{T(n) = n \log n}$

3

$$(a_0 + a_1)(b_0 + b_1) = a_0 b_0 + \underline{a_0 b_1 + a_1 b_0} + a_1 b_1$$

$$\circledast \quad \underline{a_0 b_1 + a_1 b_0} = (a_0 + a_1)(b_0 + b_1) - a_0 b_0 - a_1 b_1$$

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.

① Compute $a_0 + a_1$ and $b_0 + b_1$

② Kmultiply (a_0, b_0), (a_1, b_1), and $(a_0 + a_1, b_0 + b_1)$ ↗ only 3 multiplications ☺

③ Solve for ⊛ above using the values in 2

④ Solve $2^n(a_1 b_1) + 2^{n/2}((a_0 b_1 \overset{\circledast}{+} a_1 b_0)) + a_0 b_0$
 (bit shift and addition)

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

ADJACENCY LIST

+	−
Takes up $O(|V| + |E|)$ space	Edge access time slower $O(|E|)$ for the vertex worst-case

ADJACENCY MATRIX

+	−
$O(1)$ edge access	$O(|V|^2)$ space complexity

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

A path in graph G is the set of edges that connects point s to point t, where $s, t \in V$ of G. Let V_P be the set of all vertices in path P_s of the graph. So all $v_i \in V_P$ shares an edge with atleast one other $v_i \in V_P$, and all of those edges $\in$ path $P_{s,t}$.
↓
so all of $v_i \in V_P$ are connected as well

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 BFS and keep track of all the DISCOVERED vertices during the algorithm. At the beginning, all the vertices are initialized to be false in discovered. If the entire graph is connected, all of the vertices in discovered will be $\underline{true}$.
$O(|V| + |E|)$

5

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

k

res = []

$O(n)$
iteration

for i in range (n):
  S = mergeSort (element i of every sorted array) $\rightarrow$ $O(k \log k)$
  res. append (s)

return res

④ for every indexed i of the sorted arrays :
  - merge sort all index i element of all k sorted array
  - add the sorted result to the final result array

7

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]

$\dfrac{n}{3} = 3$

$k = $ min # of elements for it to be plurality
$(\lfloor \text{len}(A) / 3 \rfloor + 1)$

```
def pluralityElement (A, k)
```

$O\left(\dfrac{n}{3}\right)$
```
    if len(A) == k:
        if for 'all' a_i ∈ A , a_i = a_{i+1}        ──→ all elements in A
            return A[0]                                  are the same
        else
            return None
```

$T\left(\dfrac{n}{2}\right)$
subproblems = 2
size subproblems = $\dfrac{n}{2}$
```
    q = pluralityElement ( first half of A, k )
    r = pluralityElement ( second half of A, k )
```

$O(n)$
```
    if q == None and r :
        return r
    elif r == None and q:
        return q
    elif q == None and r == None:
        return None
    else
        return |q + r         ──→ return a combined list of q and
                                                                  r
```

$T(n) \le 2 T\left(\dfrac{n}{2}\right) + \dfrac{n}{3}$
$k = 1$
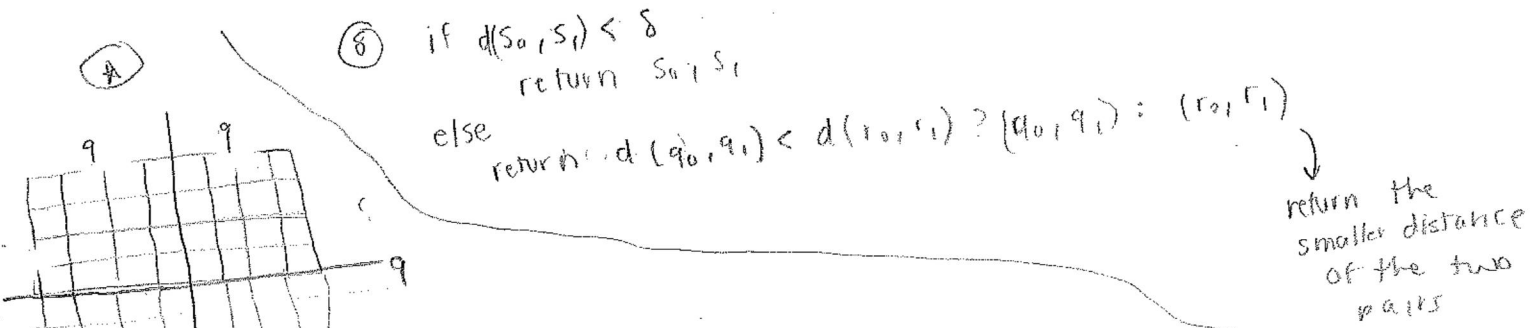$T(n) = O(n^k \log n)$
$\boxed{T(n) = O(n \log n)}$

9

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]

① Create P_x and P_y,

     set of points   set of points
     sorted by   sorted by
     x-coordinate   y coordinate

② Build $Q_x, Q_y,$ and R_x, R_y

     first $\frac{n}{2}$ points   last $\frac{n}{2}$ points
     sorted by
     x/y accordingly

③ $q_0, q_1 =$ closest-Pair-Rec (Q_x, Q_y)
     $r_0, r_1 =$ closest-pair-Rec (R_x, R_y)

④ $\delta = \min (d(q_0, q_1), d(r_0, r_1))$

⑤ $\ell = $ x-coordinate of the last point in Q_x

⑥ $S_y =$ all points in P that are δ away from ℓ (by the x-coordinate, sorted by the y-coordinate

★ ⑦ $s_0, s_1 =$ closest pair in S_y, found by looking ahead 31 points

Ⓐ

⑧ if $d(s_0, s_1) < \delta$
     return s_0, s_1
     else
       return $d(q_0, q_1) < d(r_0, r_1)$? $(q_0, q_1) : (r_0, r_1)$

return the smaller distance of the two pairs

$\frac{\delta}{4}$ $\frac{\delta}{4}$

$\sqrt[4]{\left(\frac{\delta}{4}\right)^4 + \left(\frac{\delta}{4}\right)^4}$

$\frac{\sqrt{2}\delta}{64}$ $\frac{\sqrt{2}\delta}{\sqrt{8}} < \delta$

$\sqrt{2} < \sqrt{8}$ so $\frac{\sqrt{2}}{\sqrt{8}} < 1$

no two points can be in the same square, because we already checked that δ was the minimum closest distance for Q and R. Thus, we only need to look 31 square ahead

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]
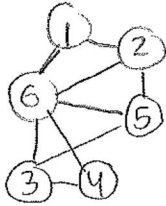
	1	2	3	4	5	6
$L[0] = [1]$	T	F	F	F	F	F
$L[1] = [2, 6]$	T	T	F	F	F	T
$L[2] = [5, 3, 4]$	T	T	T	T	T	T
$L[3] = $ NULL	T	T	T	T	T	T

13

14