CS180 Exam 1

Raymond Kwan

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.

- 1.5 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

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**

- 0.8 pts You check way too many points for S_y. Try to simplify your strip construction

- 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 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

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

$\frac{\log n}{\log 2}$ $\left(\frac{\log n}{\log 100}\right)^2$

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]

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

$k = \log_3 6 < 2$

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

Divide the problem into smaller subproblems, then recursively solve the smaller subproblems, and combine the solutions to get your final output.

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

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

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

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.

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

Karatsuba's trick:

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

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

3 products needed

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

	Adjacency List	Adjacency Matrix
Pro:	Space Complexity $O(\|V\|+\|E\|)$	Constant $O(1)$ time to check if two vertices are neighbors
Con:	For each vertex, have to go $O(degree(v))$ or $O(\|E\|)$ total time to get to an edge	Space: $O(\|V\|^2)$

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

A path between two vertices s,t is a set of consecutive edges such that s is connected to t.

i.e. $\{(s, v_1), (v_1, \ldots), (\ldots, v_i), (v_i, \ldots), (\ldots, v_{k-1}), (v_{k-1}, t)\}$

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]

Using Breadth First Search,

You can check, for each vertex, (in L[i])
its neighbors,
updating the DISCOVERED array
and adding the vertex to explore next if necessary
You stop when L[i+1] is empty, or when all its neighbors have been discovered.

A graph is connected only if after a full run through BFS, every vertex in the graph is marked DISCOVERED in the array.

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

MergeKSorted:

if $k \leq 2$, merge2SortedArrays

Divide the total set into 2 sets of $k/2$ arrays. (call them L_1 and L_2)

Recursively MergeKSorted(L_1), MergeKSorted(L_2) → $2T(\frac{k}{2})$

Merge2SortedArrays (L_1, L_2)

Merge2SortedArrays:

```
A[] ← empty
i, j ← 0
while i<n1 or j<n2:
    if i != n1 and L1[i] ≤ L2[j]:
        A.append(L1[i])
        i++
    else
        A.append(L2[j])
        j++
return A
```

12^{22}

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]

If $n \leq 3$, Brute force determine if any element is plurality. → $O(n)$

Divide A into 3 sets of size $n/3$ each. (call them L_1, L_2, L_3)

at most 2 elements → X_1 = Recursively FindPlurality(L_1) → $T(n/3)$

X_2 = .. FindPlurality(L_2) → $T(n/3)$

X_3 = ... FindPlurality(L_3) → $T(n/3)$

```
Result [] ← empty
For each x in [X1, X2, X3]     → Run at most 3×2=6 times
    If x is not null,
        count=0
        For element in A:
            if element ==x
                count ++
                if count > n/3
                    Result . append(x)
                    break
```

for each plurality in subarray, check if it is also plurality in total array, else don't add it to result.

$O(n)$

$$T(n) = 3T(n/3) + O(n)$$

$a=3, b=3$

$\log_3 3 = 1$

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

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 the median by x-coordinate.

Divide the set along a vertical line determined by the median x-coordinate (2 n/2 sets).

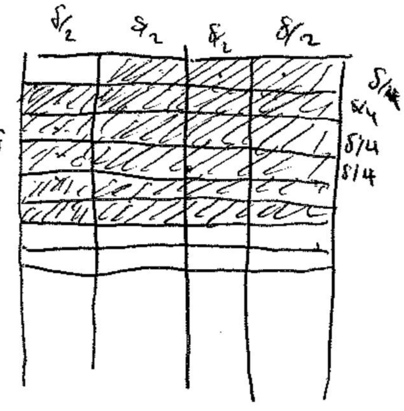
Find $\{q_x, q_y\}$, the closest pair in left half. (recursively)

Find $\{r_x, r_y\}$, the closest pair in right half. (recursively)

Set $\delta = \min\{d_4(q_x,q_y), d_4(r_x,r_y)\}$

For the points that are δ-L4 distance away from vertical median line, sort by y-coordinate.

$5 \times \frac{\delta}{4} = \frac{5}{4}\delta > \delta$



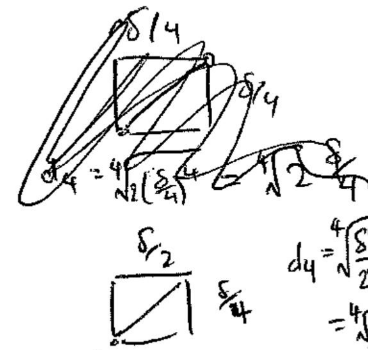
$5 \times \frac{\delta}{4} = \frac{5\delta}{4} > \delta$

$(5+1) \times 4 - 1 = 23$

For each point s_y, compute d_4 with each point t_y 23 steps ahead

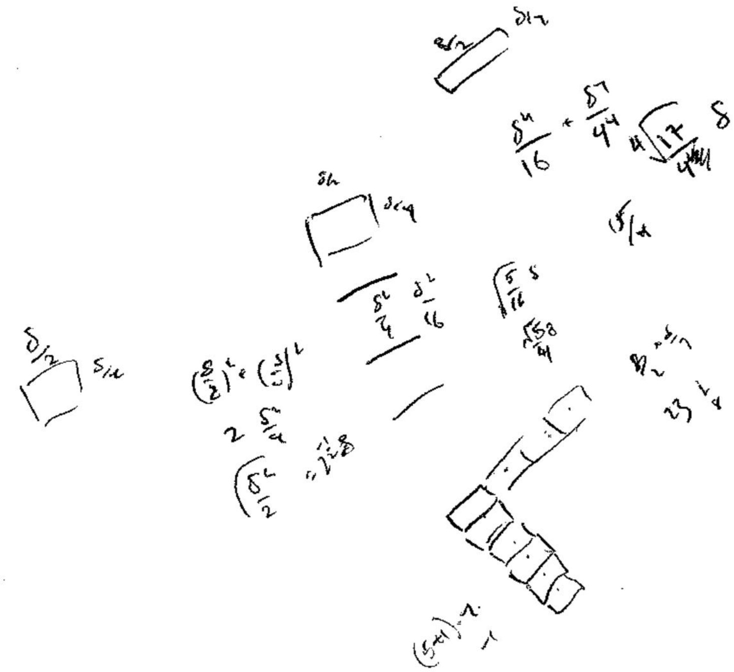
$\delta = \min(\delta, d_4(s_y, t_y))$

Return the 2 points that gave the minimum δ



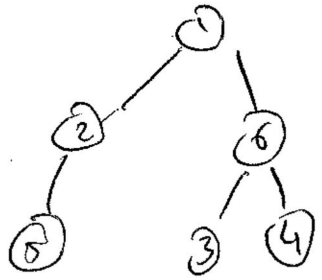
$d_4 = \sqrt[4]{\frac{\delta^4}{2^4} + \frac{\delta^4}{4^4}} = \frac{\sqrt[4]{17}}{4}\delta < \delta$

So only one point can be located in each square $\frac{\delta}{2} \times \frac{\delta}{4}$



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]



DISCOVERED

	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
$L[3] = \emptyset$						