CS-3510-C F23 Exam 1 Version A

Vidit Dharmendra Pokharna

TOTAL POINTS

91 / 110

QUESTION 1

True/False 18 pts

1.1 (i) 3 / 3

✓ **- 0 pts** *Correct*

- 3 pts Incorrect

1.2 (ii) 3 / 3

✓ **- 0 pts** *Correct*

- 3 pts Incorrect

1.3 (iii) 0 / 3

- 0 pts Correct

✓ **- 3 pts** *Incorrect*

1.4 (iv) 3 / 3

✓ **- 0 pts** *Correct*

- 3 pts Incorrect

1.5 (v) 0 / 3

- 0 pts Correct

✓ **- 3 pts** *Incorrect*

1.6 (vi) 3 / 3

✓ **- 0 pts** *Correct*

- 3 pts Incorrect

QUESTION 2

2 Comparing Runtimes 15 / 15

✓ **- 0 pts** *Correct*

- 2 pts Minor error applying Master's Theorem, but correct runtime conclusion

- 5 pts Insufficient work shown

- 3 pts All correct runtimes, incorrect/missing speed analysis

- 3 pts One incorrect runtime, correct speed analysis

- 6 pts Two incorrect runtimes, correct speed analysis

- 9 pts Three incorrect runtimes, correct speed analysis

- 6 pts One incorrect runtime, incorrect speed analysis

- 12 pts Two or more incorrect runtime and incorrect speed analysis

- 15 pts No submission

QUESTION 3

3 Shifted Sorted Array (Design + Runtime) 20 / 20

✓ **- 0 pts** *Correct*

- 2 pts Minor mistake in algorithm

- 4 pts Attempts to use binary search but incorrectly

- 5 pts Inefficient

- 6 pts Major error

- 12 pts Not a divide-and-conquer algorithm

Runtime

- **0 pts** Correct (runtime matches given algorithm)
- **2 pts** Correct recurrence, incorrect runtime
- **3 pts** No recurrence provided, correct runtime
- **5 pts** Runtime calculation incorrect for algorithm

- **20 pts** Missing

QUESTION 4

4 Merge Sorted Arrays (Design + Runtime) 19 / 22

- **0 pts** Correct

Runtime

- **0 pts** Correct (runtime matches algorithm)
- **2 pts** Correct recurrence but incorrect runtime
- ✓ **- 3 pts** *Error in recurrence or no recurrence provided*
- **5 pts** Incorrect/missing

- **2 pts** Minor mistake in algorithm
- **4 pts** Attempts to use merge sort incorrectly
- **5 pts** Inefficient
- **6 pts** Major error (does not use merge sort)
- **12 pts** Not divide-and-conquer algorithm
- **17 pts** Incorrect/Missing

QUESTION 5

Modular Arithmetic 25 pts

5.1 (i) 5 / 5

- ✓ **- 0 pts** *Correct*
- **3 pts** Incorrect, math error
- **4 pts** Incorrect, didn't attempt to use FLT
- **5 pts** Incorrect, no work shown

5.2 (ii) 5 / 5

- ✓ **- 0 pts** *Correct*
- **2 pts** Incorrect, calculation error
- **3 pts** Incorrect, multiple math errors
- **4 pts** Incorrect, didn't attempt to use FLT
- **5 pts** Incorrect, no work shown

5.3 (iii) 5 / 5

- ✓ **- 0 pts** *Correct*
- **2 pts** Incorrect, calculation error
- **3 pts** Incorrect, multiple math errors
- **4 pts** Incorrect, didn't attempt to use FLT
- **5 pts** Incorrect, no work shown

5.4 (iv) 5 / 5

- ✓ **- 0 pts** *Correct*
- **2 pts** Incorrect, calculation error
- **3 pts** Incorrect, multiple math errors
- **4 pts** Incorrect, didn't attempt to use FLT
- **5 pts** Incorrect, no work shown

5.5 (v) 5 / 5

- ✓ **- 0 pts** *Correct*
- **2 pts** Incorrect, calculation error
- **3 pts** Incorrect, multiple math errors
- **4 pts** Incorrect, didn't attempt to use FLT
- **5 pts** Incorrect, no work shown

QUESTION 6

6 Closest Points 0 / 10

- **0 pts** Correct
- **5 pts** Inefficient but still faster than $$O(n^2)$$

- 6 pts Does not correctly handle merging step

- 8 pts Attempts a Divide-and-Conquer approach

with $$n^2$$ runtime

✓ **- 10 pts** *Incorrect/Missing*

GTID: 903772087
NAME: Vidit Pokharna

Georgia Institute of Technology

Fall 2023

CS 3510 C – Design & Analysis of Algorithms
Exam 1 Version A

September 7, 2023

TIME ALLOWED: 75 MINS

Name: Vidit Pokharna

GTID: 903772087

GT Username: vpokharna3

INSTRUCTIONS TO STUDENTS

1. Please write your NAME and GTID clearly on all the pages.
2. This examination paper contains **SIX (6)** questions and comprises **ELEVEN (11)** printed pages.
3. ONLY write on the front sheets of paper that are numbered. The backs will not be scanned.
4. Calculators are **NOT** allowed.

I am in aware of and the accordance with Academic Honor Code of Georgia Tech and the Georgia Tech Code of Conduct. I'll use no external help on this test. Also, I have read all the instructions on this page.

Signature: viditpok

GTID: 903772087
NAME: Vidit Pokharna

Problem 1 *(18 points; 3 points each)*

Indicate whether the following statements are ***true*** or ***false***.

(i) $(10^{10})^n = \mathcal{O}(n!)$ $10^{10n} \leq$

Answer: true

(ii) $n^2 = \omega(n^{1.7}\log n)$ $c \cdot n^2 > n^{1.7}\log n$

Answer: true

(iii) $n^{100} = \Omega(n^{\log_2(\log_2 n)})$ false $c \cdot n^{100} \geq n^{\log_2(\log_2 n)}$ $(\log_2 n)^{(\log_2 n)}$

Answer: true

(iv) $(\sqrt{n})^3 + n\log n = \Theta(8^{\log_4(n)})$ $n^{\log_4 8} = n^{3/2}$

Answer: true $n^{3/2} + n\log n = \Theta(n^{3/2})$

(v) $\log(n) = \Theta(\log(n^{65}))$ ~~65logn~~

Answer: False

(vi) $f(n) = (n+1)^2(n-1)^2$, $T(n) = 15T(n/4) + \mathcal{O}(n^4)$
$T(n) = \mathcal{O}(f(n))$

$\log_4 15 < 2$

Answer: true $O(n^4)$

$O(n^4)$ $(n+1)^2(n-1)^2$

GTID: 903772087
NAME: Udit Pokharna

Problem 2 *(15 points)*

Suppose there are three alternative Divide-and-Conquer methods to solve a problem with input size n.

1. Divide into 4 subproblems, each with size $n/2$. Combine subproblems with additional work $\log((n!)^5) + n^3$.

2. Divide into 10 subproblems, each with size $n/8$. Combine subproblems with additional work $\sqrt{n}$.

3. Divide into 9 subproblems, each with size $n/3$. Combine subproblems with additional work $n^2 + 2^{log_{10}(n)}$

Calculate the runtime of **all three approaches**, and determine which is the fastest algorithm. Show all work.

1. $T(n) = 4 \cdot T(n/2) + O(n^3)$
$\log_b a = \log_2 4 = 2 < 3 = d$ $\boxed{O(n^3)}$

2. $T(n) = 10 \cdot T(n/8) + O(n^{1/2})$
$\log_b a = \log_8 10 > 1/2 = d$
$\boxed{O(n^{\log_8 10})}$

3. $T(n) = 9 \cdot T(n/3) + O(n^2)$
$\log_b a = \log_3 9 = 2 = d$
$\boxed{O(n^2 \log n)}$

Approach 2 is the fastest algorithm as the exponent of n in 2 is less than both 1 and 3, making it faster and of lower order

GTID: 903 7?? 087
NAME: Vidit P.

Problem 3 *(20 points)*

Suppose there is a sorted array S with n distinct integer elements. Array R is generated by choosing some number k where $1 \leq k \leq n-1$, and "rotating" the array such that all elements are shifted to the right k times. Elements that fall off the end of the array are wrapped back to the front of the array. For example, given the following array $S = [-1, 2, 3, 5, 6, 7]$ with $k = 3$, we will receive $R = [5, 6, 7, -1, 2, 3]$.

You are given an array R which is the result of some possible rotation on S and an integer value t.

(i) (15 points) Design an efficient Divide-and-Conquer algorithm to find an index i such that $R[i] = t$, or return -1 if no such index exists. Assume all indices are zero-indexed. Faster algorithms will receive more points. **You may write pseudocode OR describe your algorithm in English.**

We begin by checking the first and last element. If either matches, return the index of it. If the given array has 0, return -1. If it has length 1, check if it is t, if so, return index, if not, return -1.

Then, we check the value of middle of the array. If the value of t is between the first and middle, we call the function on the first half of the array.
If the value of t is between the last element and middle, then we call the function on the last half. If middle is equal to t, return middle index.

Note: While recursively calling the function, there must be index variables tracking the index of the first element in the recursive call so we know which i to return.

(ii) (5 points) Find your algorithm's time complexity using the Master Theorem. Show your recurrence and work.

$T(n) = 1 \cdot T(n/2) + O(1) \leftarrow$ because checking and comparing are all $O(1)$ operations

$\log_b a = \log_2 1 = 0 = d$

$\boxed{O(\log n)}$

GTID: 903772087
NAME: Vidit P.

Problem 4 *(22 points)*

Suppose you have n sorted arrays, each containing k integers (not necessarily distinct), in a list $S = [A_0, A_1, \ldots, A_{n-1}]$. You want to merge all sorted arrays into one sorted array B of size kn. For example, given $S = [[1, 2], [1, 3], [-5, 17], [2, 4]]$, the output $B = [-5, 1, 1, 2, 2, 3, 4, 17]$. Recall the `merge` subroutine from `mergesort`, that takes in two sorted arrays of length s and returns a sorted array of length $2s$ in $\mathcal{O}(s)$ time. You may use it as a black-box to solve this problem.

(i) (17 points) Design an efficient Divide-and-Conquer algorithm to find B for input S. Faster algorithms will receive more points. **You may write pseudocode OR describe your algorithm in English.**

we can use recursion to solve this. First, we will check if the size of the list is 0 or 1. We will return the list for both cases.

If it has a size of 2, then we return the sorted array after calling merge on the given list

From here, we can divide our given list into two parts: S[0:n/2] and S[n/2:n]. We can call our function on both of these sublists.

In the end, the recursive calls will eventually build up a sorted array.

(ii) (5 points) Find your algorithm's time complexity using the Master Theorem. Show your recurrence and work. Note that although there are now two inputs (n, k) in your recurrence, the Master Theorem is still applicable! Think carefully about how to apply it.

$$T(n,k) = 2 \cdot T(n/2, k) + O(k)$$

$$\log_b a = \log_2 2 = 1 = d$$

$$\boxed{O(k \log n)}$$

since n is the one being divided recursively

GTID: 903772087
NAME: Vidit P.

Problem 5 *(25 points; 5 points each)*

Compute the following. All answers must be in the interval $[0, M-1]$ where M is the modulus. **Show all work.**

(i) $3^{16} \pmod 5$

Answer: 1
Work:

$3^4 \equiv 1 \bmod 15$

$(3^4)^4 \equiv 1^4 \bmod 15 \equiv 1 \bmod 15$

(ii) $2^{756} \pmod{11}$

Answer: 9
Work:

$2^{10} \equiv 1 \bmod 11$

$(2^{10})^{75} \cdot 2^6 \equiv (1)^{75} \cdot 64 \equiv 64 \bmod 11 \equiv (64-55) \bmod 11 \equiv 9 \bmod 11$

(iii) $9^{782} \pmod{79}$

Answer: 2
Work:

$9^{78} \equiv 1 \bmod 79$

$(9^{78})^{10} \cdot 9^2 \equiv (1)^{10} \cdot 81 \bmod 79 \equiv 81 \bmod 79 \equiv 2 \bmod 79$

GTID: 903772087
NAME: Vidit P.

(iv) Find an integer x such that $x^{75} \equiv 3 \pmod 5$ such that $0 \leq x \leq 4$.

Answer: 2
Work:

$x^4 \equiv 1 \bmod 5$

$(x^4)^{18} x^3 \equiv 3 \bmod 5$

$(1)^{18} x^3 \equiv 3 \bmod 5$

$x^3 \equiv 3 \bmod 5$

$x = 2$

(v) $k^{kp} \pmod p$ where $k \in \mathbb{N}$ and p is a prime such that $p > k^k$ and p doesn't divide k^k. Express your answer in terms of k and/or p.

Answer: k^k
Work:

$k^{p-1} \equiv 1 \bmod p$

$k^{kp} \bmod p \equiv (k^{p-1})^k \cdot k^k \bmod p \equiv (1^k) \cdot k^k \bmod p \equiv k^k \bmod p$

GTID: 903772087
NAME: Vidit P.

Problem 6 *(Extra Credit; 10 points)*

Suppose you are given an array of n points $P = [(p_{1,x}, p_{1,y}), (p_{2,x}, p_{2,y}), \ldots, (p_{n,x}, p_{n,y})]$ on a plane. The points in p are **sorted by x-coordinate**; for any indices i, j, if $i \le j$, then $p_{i,x} \le p_{j,x}$. You are given a function $\texttt{dist}(p, q)$ that computes the Euclidean distance between points p and q in $\mathcal{O}(1)$ time. Design a Divide-and-Conquer algorithm to find the closest pair of points. For example, given the points $[(-2,9),(0,7),(1,6),(5,6)]$ your algorithm should return the pair of points $(0,7),(1,6)$. You will receive full points if your algorithm runs in $o(n^2)$ time. You may use pseudocode OR write an explanation in English. Write your algorithm's recurrence and find its runtime.

We can use a variation of binary search.
First we check if the array is size 0 or 1 or 2.
If it's 0 or 1, we return 0, based on further specification. If it's 2, we return the dist values of the points.

~~Next, we calculate mid = (len(P)/2). We check the two points adjacent to mid using the dist function. We save the max of those two in an int value called max.~~

We call recursive calls for the first half of P and the second half of P from there.

~~We have an integer with value infinity called max. If either call produces a value greater than max we return those two points.~~

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

~~$O(n^2)$~~

GTID: 903772087
NAME: Vidit P.

we store all return dist values in an array. After all calls are made, we can find the smallest value in the array and return that index and that index plus 1 in P for the two points with closest distance.

checking final array for smallest value

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

$$\log_b a = \log_2 2 = 1 = d$$

$$\boxed{O(n \log n)}$$

END OF EXAM