Midterm 1

Graded

Student

SAIM M MEHER

Total Points

98 / 150 pts

Question 1

Problem 1 — **34** / 50 pts

1.1 **1.1a** — **6** / 6 pts

✔ **+ 6 pts** n^2

The statement execution depends on how many times the inner for-loop is executed. The inner loop executes n times (from j=0 to j=n-1) for every i. Therefore, $n * n$ => n^2

+ 0 pts Incorrect - no partial credit

1.2 **1.1b** — **0** / 6 pts

+ 6 pts $2n^2$ or $4n^2$

There are 2 array accesses that happen, m[i]=false and !m1[i], for every iteration of the inner loop. So, 2 * n^2 => n^2

One to read the row and another to read the column.
In that case, there are 4 array accesses that happen, m[i]=false and !m1[i], for every iteration of the inner loop. So, $4 * n^2$ => $4n^2$

✔ **+ 0 pts** Incorrect - no partial credit

1.3 **1.1c** — **0** / 6 pts

+ 6 pts $2n^3 + 2n^2$ or $4n^3 + 4n^2$

The j loop executes n times (from j=0 to j=n-1) for every i. Therefore, n*n => n^2
• the m[i] = false is executed n^2 times: $1n^2$
• the 2 array accesses in the while-loop conditional are also executed n^3 times; $2n^3$
• m[i] = true is executed n^2 times in the worst case: n^2
$1n^2 + 2n^3 + 1n^2$ => $2n^3 + 2n^2$

One to read the row and another to read the column
The j loop executes n times (from j=0 to j=n-1) for every i. Therefore, n*n => n^2
• the m[i] = false is executed n^2 times: $2n^2$
• the 4 array accesses in the while-loop conditional are also executed n^3 times; $4n^3$
• m[i] = true is executed n^2 times in the worst case: $2n^2$
$2n^2 + 4n^3 + 2n^2$ => $4n^3 + 4n^2$

✔ **+ 0 pts** Incorrect - no partial credit

1.4 **1.1d** — **0** / 4 pts

+ 4 pts ~$2n^3$ or ~$4n^3$

✔ **+ 0 pts** Incorrect - no partial credit

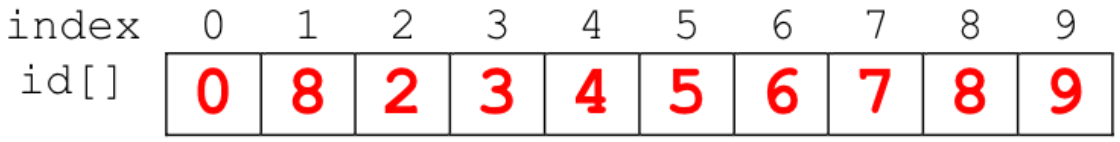
1.5 **1.1e** — **3** / 3 pts

✔ **+ 3 pts** $O(n^3)$

+ 0 pts Incorrect - no partial credit

1.6 **1.2a.i** **2** / 2 pts

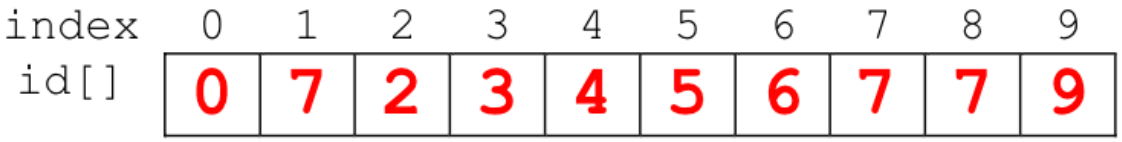
✔ **+ 2 pts** Correct



+ 0 pts Incorrect - no partial credit

1.7 **1.2a.ii** **2** / 2 pts

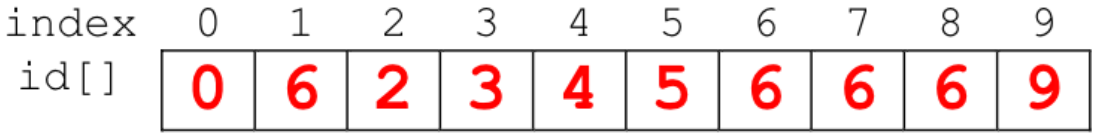
✔ **+ 2 pts** Correct or correct based on previous answer



+ 0 pts Incorrect - no partial credit

1.8 **1.2a.iii** **2** / 2 pts

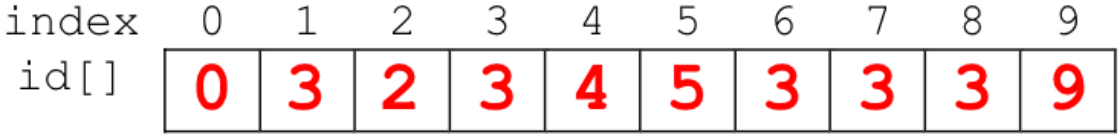
✔ **+ 2 pts** Correct or correct based on previous answer



+ 0 pts Incorrect - no partial credit

1.9 **1.2a.iv** **2** / 2 pts

✔ **+ 2 pts** Correct or correct based on previous answer



+ 0 pts Incorrect - no partial credit

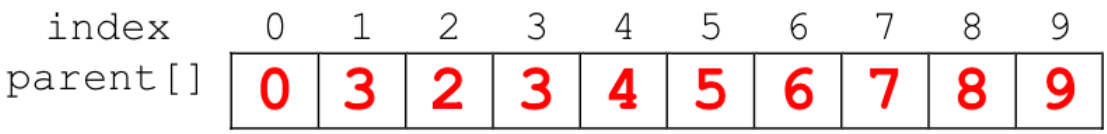
1.10 **1.2a.v** **2** / 2 pts

✔ **+ 2 pts** **6** or correct based on previous answer

+ 0 pts Incorrect - no partial credit

1.11 **1.2b.i** **3** / 3 pts

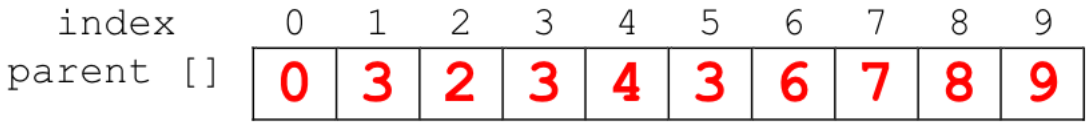
✔ **+ 3 pts** Correct



+ 0 pts Incorrect - no partial credit

1.12 **1.2b.ii** **3** / 3 pts

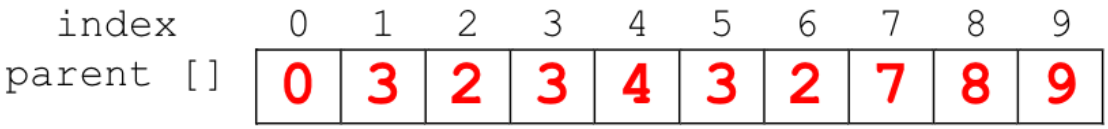
✔ **+ 3 pts** Correct or correct based on previous answer



+ 0 pts Incorrect - no partial credit

1.13 **1.2b.iii** **3** / 3 pts

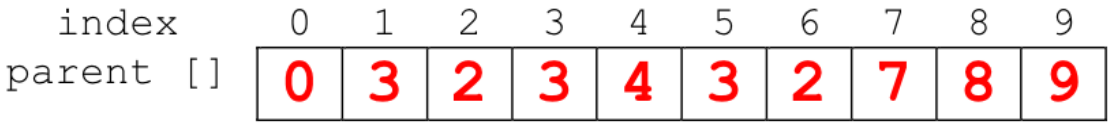
✔ **+ 3 pts** Correct or correct based on previous answer



+ 0 pts Incorrect - no partial credit

1.14 **1.2b.iv** **3** / 3 pts

✔ **+ 3 pts** Correct or correct based on previous answer



+ 0 pts Incorrect - no partial credit

1.15 **1.2b.v** **3** / 3 pts

✔ **+ 3 pts** **7** or correct based on previous answer

+ 0 pts Incorrect - no partial credit

Question 2

Problem 2 — **22** / 43 pts

2.1 **2.1** — **0** / 7 pts

+ 7 pts **A** - Insertion Sort

✔ + 0 pts Incorrect - no partial credit

2.2 **2.2** — **0** / 0 pts

✔ + 7 pts **D** - Selection Sort

+ 0 pts Incorrect - no partial credit

2.3 **2.3** — **0** / 7 pts

+ 7 pts **D** - Selection Sort

✔ + 0 pts Incorrect - no partial credit

2.4 **2.4** — **0** / 7 pts

+ 7 pts **C** - Mergesort

✔ + 0 pts Incorrect - no partial credit

2.5 **2.5** — **11** / 11 pts

✔ + 11 pts **Insertion Sort** - Takes linear time when the array is already sorted. All other algorithms take longer than that.

+ 5 pts Insertion Sort

+ 6 pts Takes linear time when the array is already sorted. All other algorithms take longer than that.

+ 0 pts All incorrect

2.6 **2.6** — **11** / 11 pts

✔ + 11 pts **B** - Quicksort or **C** - Mergesort
Quicksort and mergesort will take $nlogn$ to sort the array
Insertion sort and selection sort will take n^2 to sort the array at least 50% of the time.

+ 5 pts **B** - Quicksort or **C** - Mergesort

+ 6 pts Quicksort and mergesort will take $nlogn$ to sort the array
Insertion sort and selection sort will take n^2 to sort the array at least 50% of the time.

+ 0 pts All incorrect

Question 3

Problem 3 **35** / 50 pts

3.1 **3.1a** **5** / 5 pts

✔ **+ 5 pts** Correct

1st partition - 49 is in its final position

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	10	20	21	28	34	19	7	33	17	9	49	69	55	65	51

+ 0.5 pts index 0 = **5**

+ 0.5 pts index 1 = **10**

+ 0.5 pts index 3 = **21**

+ 0.5 pts index 10 = **9**

+ 1 pt index 11 = **49**

+ 0.5 pts index 12 = **69**

+ 0.5 pts index 13 = **55**

+ 0.5 pts index 14 = **65**

+ 0.5 pts index 15 = **51**

+ 0 pts Incorrect

3.2 **3.1b** **5** / 5 pts

✔ **+ 5 pts** Correct

2nd partition - 5 and 49 are in their final positions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	10	20	21	28	34	19	7	33	17	9	49	69	55	65	51

+ 0 pts Incorrect - no partial credit

3.3 **3.1c** **5** / 5 pts

✔ **+ 5 pts** Correct

3rd partition - 5, 10, 49 are in their final positions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	7	9	10	28	34	19	21	33	17	20	49	69	55	65	51

+ 1 pt index 1 = **7**

+ 1 pt index 2 = **9**

+ 1 pt index 3 = **10**

+ 1 pt index 7 = **21**

+ 1 pt index 10 = **20**

+ 0 pts Incorrect

3.4 **3.1d** **5** / 5 pts

✔ **+ 5 pts** Correct

4th partition - 5, 7, 10, 49 are in their final positions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	7	9	10	28	34	19	21	33	17	20	49	69	55	65	51

+ 0 pts Incorrect - no partial credit

3.5 **3.1e** **5** / 5 pts

✔ **+ 5 pts** Correct

5th partition - 5, 7, 9, 10, 28, 49 are in their final positions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	7	9	10	17	20	19	21	28	33	34	49	69	55	65	51

+ 1 pt index 4 = **17**

+ 1 pt index 5 = **20**

+ 1 pt index 8 = **28**

+ 1 pt index 9 = **33**

+ 1 pt index 10 = **34**

+ 0 pts Incorrect

3.6 **3.2a** **0** / 5 pts

+ 5 pts Index **3**

✔ **+ 0 pts** Incorrect - no partial credit

3.7 **3.2b** **5** / 5 pts

✔ **+ 5 pts** **3** partitions (Previously incorrect answer **4**)

+ 0 pts Incorrect - no partial credit

3.8 **3.3a** **0** / 5 pts

+ 5 pts **Array B**

✔ **+ 0 pts** Incorrect - no partial credit

3.9 **3.3b** **0** / 5 pts

+ 5 pts $O(nlogn)$

✔ **+ 0 pts** Incorrect - no partial credit

3.10 **3.3c** **5** / 5 pts

✔ **+ 5 pts** $O(n^2)$

+ 0 pts Incorrect - no partial credit

Question 4

2.2 (FULL POINTS) **7** / 7 pts

✔ **+ 7 pts** Correct

CS112 Data Structures - Midterm 1 - Fall 2025

Name: Saim Meher NetID: smm742

- **WRITE your name and NetID on EVERY page.**
- **DO NOT REMOVE** THE STAPLE IN YOUR EXAM.
- **DO NOT BEGIN** UNTIL INSTRUCTED TO DO SO.
- WRITE NEATLY AND CLEARLY. If we cannot read your handwriting, you will not receive credit. Please plan your space usage. No additional paper will be provided.
- This exam is worth 150 points.
- This exam has 7 pages, make sure you have all pages.

Problem 1 – Arrays, Time Complexity, and Union Find (50 points)

1. **(25 points)** Answer the questions below about the following method, that computes the product of two square Boolean matrices. Assume that n > 0.

```
public static boolean[][] multiply(int n,boolean[][] m1,boolean[][] m2){
    boolean[][] m=new boolean[n][n];
    for (int i=0;i<n;i++){
        for (int j=0;j<n;j++){
            m[i][j]=false;
            int k=0;
            while(k<n && (!m1[i][k] || !m2[k][j])) k++;
            if (k<n) m[i][j]=true;
        }
    }
    return m;
}
```

a. **(6 points)** How many times is the **`m[i][j]=false;`** statement executed in terms of *n*? n^2

b. **(6 points)** What is the minimum number of array accesses in terms of *n*? $4n^3$

c. **(6 point)** What is the maximum number of array accesses in terms of *n*? $5n^3$

d. **(4 point)** Write the tilde notation to represent the worst-case number of array accesses based on answer in part (c). $\sim 5n^3$

e. **(3 points)** Use the Big-O notation to represent the order of growth in terms of n, based on the answer in (c). $O(n^3)$

CS112 Data Structures - Midterm 1 - Fall 2025

Name: Saim Meher NetID: smm742

2. **(25 points)** The Union-Find API provides 2 methods listed below, where the find() method returns the component id containing p, and the union() method merges sets containing elements p and q.

```
int find(int p)
void union(int p, int q)
```

a. **(10 points)** Assume the API is implemented with the Quick-Find approach, where a one-dimensional array is used to keep track of the component IDs. When a union(p, q) is performed all the components id containing p are changed to the component id containing q.

Assume an array with 10 elements is properly initialized, write the array content after performing each of the following union() operations in sequence.

(i) **(2 points)** `union(1,8)`

index	0	1	2	3	4	5	6	7	8	9
id[]	0	8	2	3	4	5	6	7	8	9

(ii) **(2 points)** `union(1,7)`

index	0	1	2	3	4	5	6	7	8	9
id[]	0	7	2	3	4	5	6	7	7	9

(iii) **(2 points)** `union(8,6)`

index	0	1	2	3	4	5	6	7	8	9
id[]	0	6	2	3	4	5	6	6	6	9

(iv) **(2 points)** `union(6,3)`

index	0	1	2	3	4	5	6	7	8	9
id[]	0	3	2	3	4	5	3	3	3	9

(v) **(2 points)** Assuming that the above array is used to keep track of connected components in a Dynamic Connectivity Problem, how many connected components are there after the above union operations? 6

Name: Saim Meher NetID: smm742

b. **(15 points)** Assume the API is implemented with the Quick-Union (without weight) approach, where a one-dimensional array is used to keep track of the parents of the elements. When a union(p, q) is performed, the root of q becomes the parent of the root of p.

Assume an array with 10 elements is properly initialized, write the array content after performing each of the following union() operations in sequence.

(i) **(3 points)** `union(1,3)`

index	0	1	2	3	4	5	6	7	8	9
parent[]	0	3	2	3	4	5	6	7	8	9

(ii) **(3 points)** `union(5,1)`

index	0	1	2	3	4	5	6	7	8	9
parent[]	0	3	2	3	4	3	6	7	8	9

(iii) **(3 points)** `union(6,2)`

index	0	1	2	3	4	5	6	7	8	9
parent[]	0	3	2	3	4	3	2	7	8	9

(iv) **(3 points)** `union(3,5)`

index	0	1	2	3	4	5	6	7	8	9
parent[]	0	3	2	3	4	3	2	7	8	9

(v) **(3 points)** Assuming that the above array is used to keep track of connected components in a Dynamic Connectivity Problem, how many connected components after the above union operations? 7

CS112 Data Structures - Midterm 1 - Fall 2025

Name: Saim Meher NetID: smm742

Problem 2 – Sorting (50 points)

1. (**7 points**) The row on the top is the original input array of Strings to be sorted. The row on the bottom is the sorted array. The middle row is the array contents at some intermediate step during one of the algorithms listed below. Which algorithm is it?

0	1	2	3	4	5	6	7	8	9	10	11
lilo	fifo	data	type	hash	heap	sort	link	leaf	push	find	swap
data	fifo	hash	heap	lilo	link	sort	type	leaf	push	find	swap
data	fifo	find	hash	heap	leaf	lilo	link	push	sort	swap	type

a. Insertion Sort
b. Quicksort
c. Mergesort
d. Selection Sort

2. (**7 points**) The row on the top is the original input array of Strings to be sorted. The row on the bottom is the sorted array. The middle row is the array contents at some intermediate step during one of the algorithms listed below. Which algorithm is it?

0	1	2	3	4	5	6	7	8	9	10	11	12	13
T	X	D	R	S	L	V	R	Z	T	F	K	I	A
A	D	F	I	S	L	R	V	Z	T	X	K	R	T
A	D	F	I	K	L	R	R	S	T	T	V	X	Z

a. Insertion Sort
b. Quicksort
c. Mergesort
d. Selection Sort

3. (**7 points**) The row on the top is the original input array of Strings to be sorted. The row on the bottom is the sorted array. The middle row is the array contents at some intermediate step during one of the algorithms listed below. Which algorithm is it?

0	1	2	3	4	5	6	7	8	9	10	11	12	13
W	E	M	C	Y	Q	A	T	V	D	J	R	B	E
A	B	C	D	E	Q	W	T	V	M	J	R	Y	E
A	B	C	D	E	E	J	M	Q	R	T	V	W	Y

a. Insertion Sort
b. Quicksort
c. Mergesort
d. Selection Sort

Name: Saim Meher NetID: smm742

4. (**7 points**) The row on the top is the original input array of Strings to be sorted. The row on the bottom is the sorted array. The middle row is the array contents at some intermediate step during one of the algorithms listed below. Which algorithm is it?

0	1	2	3	4	5	6	7	8	9
W	E	C	A	T	V	D	J	B	E
A	C	E	T	W	D	J	V	B	E
A	B	C	D	E	E	J	T	V	W

a. Insertion Sort
b. Quicksort
c. Mergesort
d. Selection Sort

5. (**11 points**) Which sorting algorithm would you use, Insertion Sort, Selection Sort, Mergesort or Quicksort, to sort a large array that is known to be almost sorted? Justify your answer.

To sort a large array that is almost sorted, I would use insertion sort because it has a big-O time complexity of O(n) when the array is sorted.

6. (**11 points**) Suppose you are given a large array to sort. About 50% of the time the array will already be sorted. The other 50% of the time, the array items are in a completely random order. Which of the following sorting algorithms would you use? Justify your answer.

a. Insertion Sort
b. Quicksort
c. Mergesort
d. Selection Sort

I would use mergesort because it has a worst and best time complexity of O(nlogn) so the array being completely random or already sorted doesn't matter.

Name: Saim Meher NetID: smm742

Problem 3 – Quicksort (50 points)

1. **(25 points)** Assume this Quicksort code is used to sort the unsorted array below. Write the array content after each of the 1st, 2nd, 3rd, 4th and 5th partitions, assuming the first element of the array/subarray is used as the pivot (partition point) for the partitions.

```
public static void quicksort (int[] a, int lo, int hi) {
    if ( hi <= lo ) return;
    int p = partition(a, lo, hi);
    quicksort(a, lo, p-1);
    quicksort(a, p+1, hi);
}
```

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	49	51	20	65	28	34	19	7	33	17	55	69	5	9	21	10

a. **(5 points)** After 1st call to partition()

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	10	20	21	28	34	19	7	33	17	9	49	69	55	65	51

b. **(5 points)** After 2nd call to partition()

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	10	20	21	28	34	19	7	33	17	9	49	69	55	65	51

c. **(5 points)** After 3rd call to partition()

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	7	9	10	28	34	19	21	33	17	20	49	69	55	65	51

d. **(5 points)** After 4th call to partition()

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	7	9	10	28	34	19	21	33	17	20	49	69	55	65	51

e. **(5 points)** After 5th call to partition()

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	7	9	10	17	20	19	21	28	33	34	49	69	55	65	51

Name: Saim Meher NetID: smm742

2. **(10 points)** Assume we run Quicksort to sort the array below in ascending order.

a. **(5 points)** Which array index splits the array into two subarrays after the first partition? 2

b. **(5 points)** How many partitions are performed to complete the Quicksort? 3

index	0	1	2	3	4	5
	6	6	6	6	6	6

3. **(15 points)** Assume we run Quicksort to sort the arrays A and B below in ascending order.

a. **(5 points)** Which one needed more compares to complete the sorting? Both needed the same.

b. **(5 points)** If *n* represents the number of elements in an array, and all the elements have the same values, use Big-O notation to represent the number of compares to perform Quicksort on the array. $O(n^2)$

c. **(5 points)** If *n* represents the number of elements in an array, and the array elements are already in ascending order before the sorting, use Big-O notation to represent the number of compares to perform Quicksort on the array. $O(n^2)$

array A

0	1	2	3	4	5
6	6	6	6	6	6

array B

0	1	2	3	4	5
5	6	7	8	9	10

Saim Meher

smm742