

F18 CS20A Quiz 2

STEWART DULANEY

TOTAL POINTS

23 / 23

QUESTION 1

1 3 / 3

✓ - 0 pts Correct

- 1.5 pts Click here to replace this description.

- 3 pts Click here to replace this description.

QUESTION 2

4 pts

2.1 2 / 2

✓ - 0 pts Correct

- 1 pts Click here to replace this description.

- 2 pts Click here to replace this description.

2.2 2 / 2

✓ - 0 pts Correct

- 1 pts Click here to replace this description.

- 2 pts Click here to replace this description.

QUESTION 3

6 pts

3.1 1 / 1

✓ - 0 pts Correct

- 1 pts Click here to replace this description.

3.2 1 / 1

✓ - 0 pts Correct

- 1 pts incorrect

3.3 1 / 1

✓ - 0 pts Correct

- 1 pts Incorrect

3.4 1 / 1

✓ - 0 pts Correct

- 1 pts Incorrect

3.5 2 / 2

✓ - 0 pts Correct

- 1 pts Click here to replace this description.

- 2 pts Click here to replace this description.

QUESTION 4

4 No Problem 4 0 / 0

✓ - 0 pts Correct

QUESTION 5

6 pts

5.1 4 / 4

✓ - 0 pts Correct

- 1 pts Click here to replace this description.

- 2 pts Click here to replace this description.

- 4 pts Click here to replace this description.

5.2 2 / 2

✓ - 0 pts Correct

- 1 pts Click here to replace this description.

- 2 pts Click here to replace this description.

QUESTION 6

4 pts

6.1 2 / 2

✓ - 0 pts Correct

- 1 pts Click here to replace this description.

- 2 pts Incorrect

6.2 1 / 1

✓ - 0 pts Correct

- 2 pts Click here to replace this description.

6.3 1 / 1

✓ - 0 pts Correct

- 2 pts Click here to replace this description.

Name: Stewart Dulaney

ID: 1545566

Problem 1: Partition a Doubly Linked List

Consider the quicksort partition implementation for arrays in the lecture slides. Translate that partition function to operate on a doubly linked list. It takes pointers to the first and last nodes in the list and will return a pointer to the pivot node. There are several different ways of implementing such a partition function, but I want you maintain as much of the form of the original code as possible. Node definition on the right.

You may assume that partition will always be given a list with at least 2 Nodes. You can also assume that you have a function called swap that correctly swaps two integers. Recall that nodes in a linked list are at random addresses, so comparisons such as less-than or greater-than do not make sense in the context of linked lists.

```
class Node {  
public:  
    int value;  
    Node* next;  
    Node* prev;  
};
```

```
Node* partition(Node *low, Node *high) {  
    Node* pNode = low;  
    int pivot = low->value;  
    bool lowBeforeHigh = true;  
    do {  
        while ( low != high && low->value <= pivot ) {  
            low = low->next;  
        }  
        if ( low == high && low->value <= pivot ) {  
            low = low->next;  
            lowBeforeHigh = false;  
        }  
        while ( high->value > pivot ) {  
            if ( low == high ) {  
                lowBeforeHigh = false;  
            }  
            high = high->prev;  
        }  
        if ( lowBeforeHigh ) {  
            swap ( low->value, high->value );  
        }  
    } while ( lowBeforeHigh );  
    swap ( pNode->value, high->value );  
    pNode = high;  
    return pNode;  
}
```

Name: Stewart Dulaney

ID: 1545566

Problem 2: Recursion

- a. Recall that we saw a version of a recursive power function in assignment 4. However, its implementation is not intuitive. Write a recursive function that computes base^{exp} (base raised to the power of exp) that is easier to understand. You may assume that exp is nonnegative. What is the Big-O of this function?

```
// Precondition: exp >= 0
int recursivePow(int base, int exp) {
    if (exp == 0) {
        return 1;
    }
    return base * recursivePow(base, exp - 1);
}
```

The Big-O is $O(\text{exp})$

- b. Write a function `powerThree` that, given a non-negative number n, returns 3^n (3^n , or "3 raised to power n") recursively, assuming 3^n is something that can be represented as an integer. Do not use a loop, and do not use the character `'*'` (multiply) anywhere in your code. What is the Big-O of this function?

```
// Precondition: n >= 0
int powerThree(int n) {
    if (n == 0) {
        return 1;
    }
    int x = powerThree(n - 1);
    return x + x + x;
}
```

The Big-O is $O(n)$

}

Name: Stewart Dulaney

ID: 1545566

Problem 3: Big-O

- a. Suppose Algorithm A and B perform the same task. For any given input size n , algorithm A executes in $f(n) = 0.003n^2$ operations, and algorithm B executes $f(n) = 250n$ operations. Specifically, when does Algorithm A perform better than B?

$$0.003n^2 = 250n$$

$$\frac{0.003n^2}{0.003n} = \frac{250n}{0.003n}$$

$$n = 83,333.3\bar{3}$$

Alg. A and Alg. B perform the same at intersection point

∴ Alg. A performs better than Alg. B
when $n < 83,333.3\bar{3}$

- b. An algorithm that is $O(n^2)$ takes 10 seconds to complete when $n=100$. How long would you expect it to take when $n=500$?

$$(100)^2 = 10,000 \quad // \quad 10s, \text{ so } n=100 \text{ is in ms}$$

$$(500)^2 = 250,000 \text{ ms} \quad // \quad 250s$$

We would expect it to take 250 seconds when $n = 500$

- c. What is the Big-O of the following code segment:

```
for (int i = 0; i < n; i++) (n) (fixed)
    for (int j = 0; j < 4 * i; j++) i_max = n-1, rounded up to n → (4n)
        sum++;
```

$$n \cdot 4n$$

$$4n^2$$

$$O(n^2)$$

- d. What is the Big-O of the following function?

```
int gobidygoop(int n, int p) {
    int ac = 1;
    for (int i = 0; i < n; i++) { (n) (fixed)
        int k = p;
        while (k > 1) { (log4 p) (fixed)
            ac *= i + k;
            k /= 4;
        }
    }
}
```

$$n \cdot \log_4 p$$

$$O(n \cdot \log_4 p)$$

Name: Stewart Dulaney

ID: 1545566

e. Consider the following pseudo code:

```

Get value for n
Set the value of k to 1
While k is less than or equal to n  (n) (Fixed)
    Set the value of j to twice of k
    While j is greater or equal to 1  k_max = n → log₂ 2n
        Print the value of j
        Set the value of j to one half its former value;
    Increase k by 1
    
```

n

k

j

// Assume n, k, j have type int so integer division is used
What is the Big-O of this pseudocode? What does this print if n is 4?

$n = \log_2 2n$

Output:

2 1 4 2 1 6 3 1 8 4 2 1

$O(n \log_2 n)$

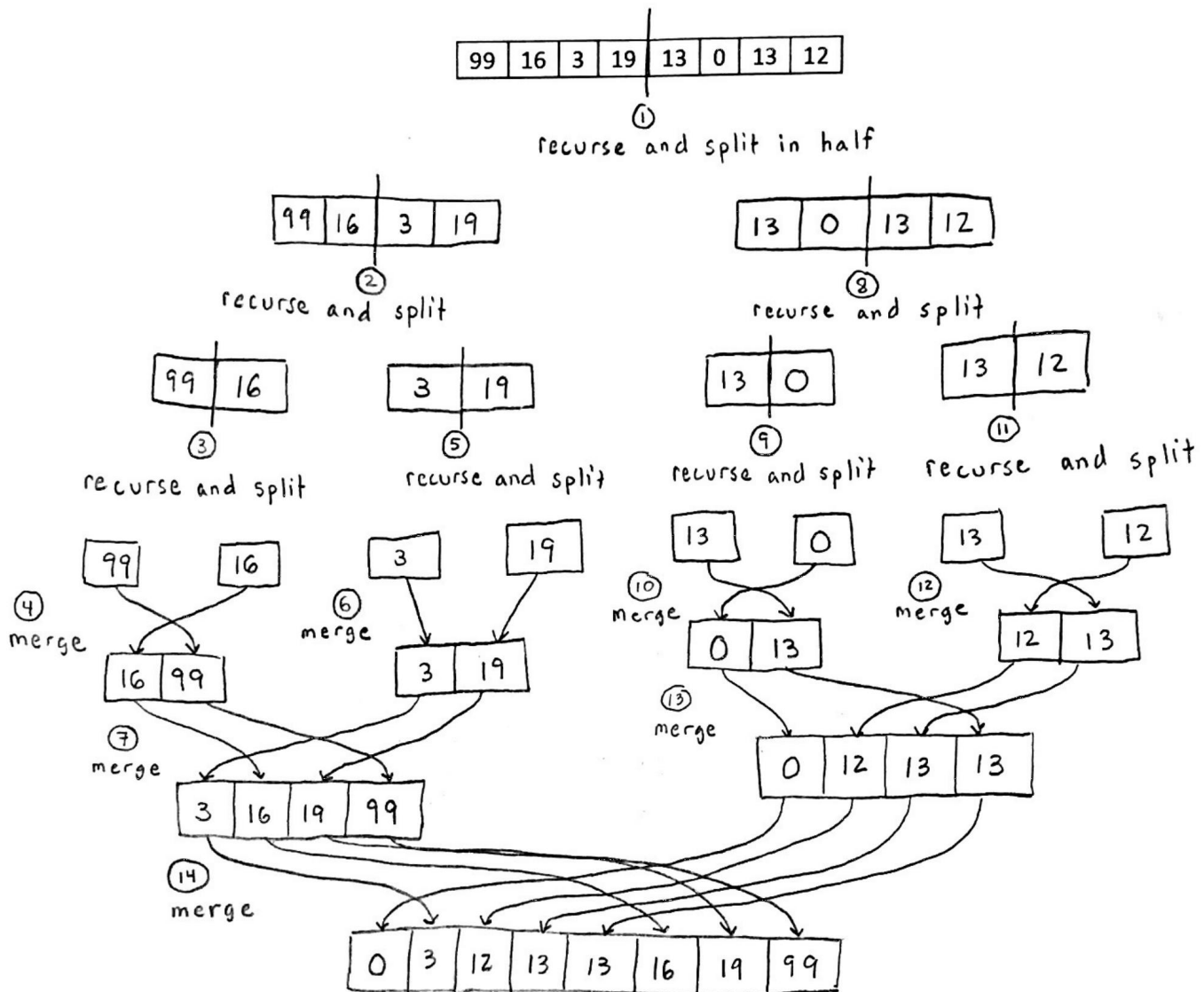
Problem 5: Sorting

- a. Fill out the following table of sorting properties, if there is no special condition for a particular case then leave it blank:

Sorting Algorithm	Selection	Insertion	Bubble	Quick	Merge
Average Complexity	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n \log n)$	$O(n \log n)$
Worse Complexity	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n \log n)$
Condition for Worse				input already or mostly sorted OR in reverse order	
Best Complexity	$O(n^2)$	$O(n)$	$O(n)$	$O(n \log n)$	$O(n \log n)$
Condition for Best		input already or mostly sorted	input already or mostly sorted		

Name: Stewart Dulaney ID: 1545566

b. Sort the following array using the Mergesort algorithm. Show each recursive step, including the merge.

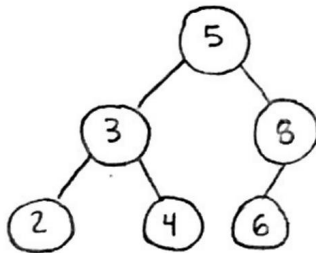


Name: Stewart Dulaney

ID: 1545566

Problem 6: Tree Traversal

- a. Suppose we define an empty tree to have height 0. Draw a ^{① ✓} complete binary search tree with ^{② ✓} height 3 whose in-order traversal is 2,3,4,5,6,8. ^{③ ✓}



- b. What is the post-order traversal of this tree?

2, 4, 3, 6, 8, 5

- c. What is the pre-order traversal of this tree?

5, 3, 2, 4, 8, 6