Summer 20 CS 5006

Name Saumeng Chea Date: 06/15/20

Each problem is worth 5 points unless otherwise state. Please do not leave questions blank, write down your thoughts to at least attempt to earn partial credit–good luck, you have learned so much!

1. What is the main difference between the array and linked list. That is, discuss how is the memory in each structured. Note: A picture in addition to a brief explanation may be helpful!
Arrays are index based data structure where each element associated w/ an index. On other hand, Linked List relies on reference where each node consists of the data and reference to that previous and next element. As for the memory, an array consumes contigious memory location allocation at a compile time, where as Linked List, the memory is being assigned as and when data is added to it.
2. In Big-Oh notation, label the complexity of the following:
   - How long does it take to update an arbitrary item in an array? O(1)
   - How long does it take to update an item in a singly linked list. O(n)

3. In Big-Oh notation, label the complexity of the following:
   - Performing a Linear Search in a sorted array: O(n)
   - Performing a Linear Search in an unsorted array: O(n)
   - Performing a Binary Search in a sorted array: O(log n)

4.
```
// Doubly-linked list node
typedef struct DLL_Node{
    struct DLL_Node* next;
    struct DLL_Node* prev;
    int data;
}DLL_Node_t;

// Singly-linked list node
typedef struct Node{
    struct Node* next;
    int data;
}Node_t;
```

In our homework we implemented both a singly and doubly-linked list(DLL). Fill out the table below comparing the asymptotic complexities of each data structure. Assume you have pointers to the head and tail in both implementations.

Big-Oh complexity			
Data Structure	insert	push_back	pop_back
Doubly Linked List	O(n)	O(1)	O(1)
Singly Linked List	O(n)	O(1)	O(n)

5. List one advantage and one disadvantage of using a doubly-linked list versus a singly-linked list?

The advantage of DLL vs. SLL is that DLL can be traversed in both direction - Forward or backward. The disadvantage is that DLL required extra storage for previous pointer and SLL don't. SLL only occupied less memory.

6. You are given a collection of 1,000,000,000 books that are already sorted. They are labeled Book1, Book2, Book3, Book4 (book 1 is less than book 2, book 3, less than book 4, etc.). Your colleague says organizing these books in a binary search tree (BST) data structure will be useful for looking up books later.

   - Your colleague begins inserting each book in order, but you are worried you will not get Log(N) lookup when inserting books in already sorted order.
   - **Answer:** Why or why not will you achieve Log(N) when inserting books in sorted order?
   - Note: It may be helpful to draw a picture of the 'tree' growing with each insertion.
   - Assume in the BST a left-child of the parent is less than the parent, and a right-child of the parent is greater.

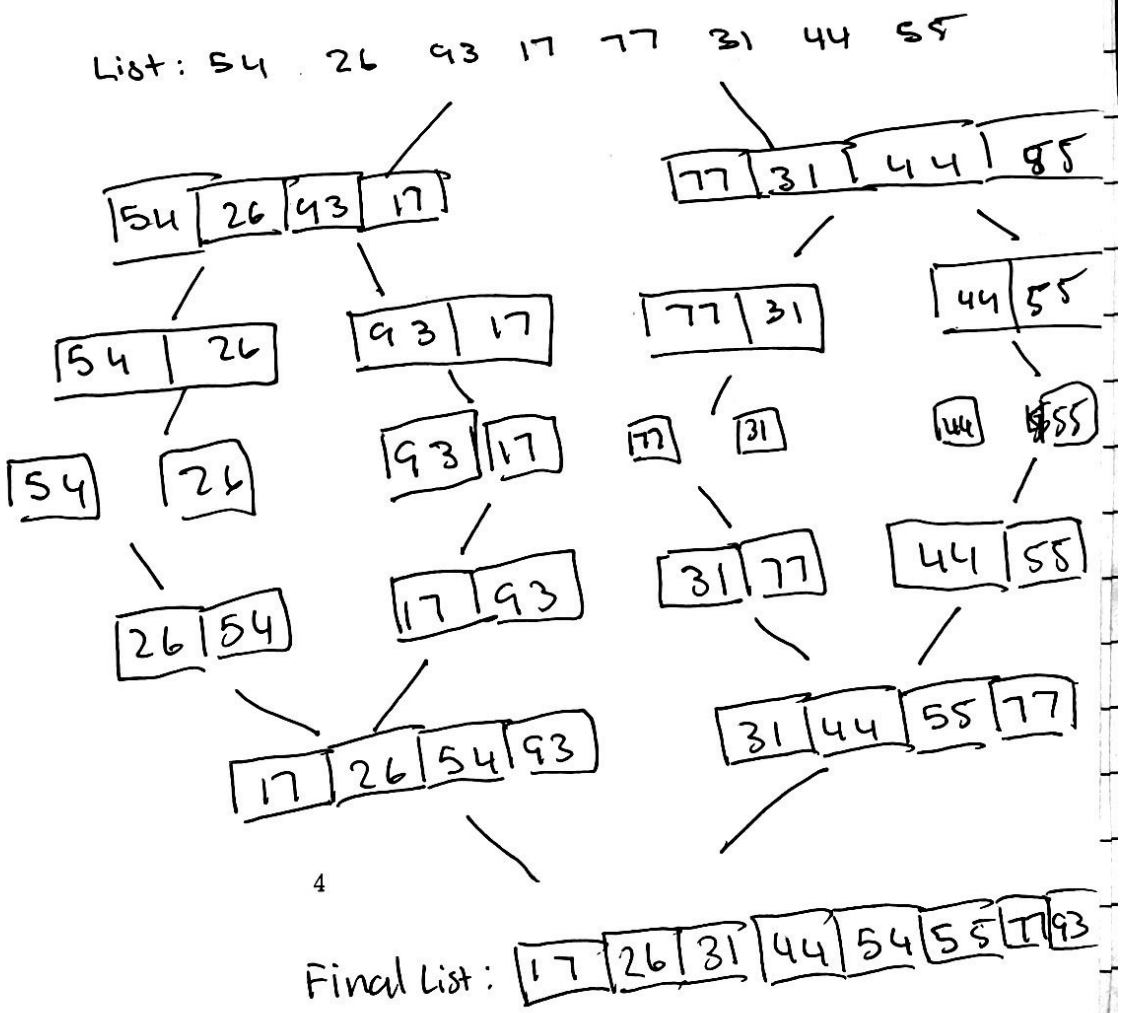
Yes we still able to achieve log n look up when inserting books in sorted order. When the colleague finished the inserting each book in order, the sorts become Linear.

7. Given the following numbers in an array, draw how the merge sort would sort the numbers (Show the divide, conquer, and combine steps). What is the Big-Oh complexity of merge-sort in the worse-case? Is it different then the best case scenario?

54	26	93	17	77	31	44	55

- The worst Big-Oh complexity of merg-sord is $O(n \cdot \log n)$.
- There isn't any different then the best case and worst case scenario.. Both Best and worst case complexity of merge sort is $O(n \cdot \log n)$

Drawing:



8. Given the following numbers in an array, draw how the insertion sort would sort the numbers (show how the array would look after each iteration of the outer for (i.e major) loop). Briefly explain how insertion sort works. What is the worst and best case running time?

54	26	93	17	77	31	44	55

Explain: It's a sorting algo. where the sorted arrays is built from having one item at a time. The array element are compare with each other sequentially & then arranged simultaneously in some particular order.

Best: $O(n)$

worst: $O(n^2)$

* diagram in scrap paper.

9. Given the following numbers in an array, draw how the bubble sort would sort the numbers. Briefly expalin how bubble sort works. What is the worst and best case running time?

54	26	93	17	77	31	44	55

explain: It's a sorting algorithms that repeatly steps throught the list, compare adjacent element and swaps them if they are in the wrong order. The pass through the list is repeated until is sorted.

Best: $O(n)$

Worst: $O(n^2)$

* diagram in the next page.(scrap).

8. Literation: List: 54 26 93 17 77 31 44 55

1 ⇒ 26 54 93 17 77 31 44 55

2 ⇒ 26 54 93 17 77 31 44 55

3 ⇒ 17 26 54 93 77 31 44 55

4 ⇒ 17 26 54 77 93 31 44 55

5 ⇒ 17 26 31 54 77 93 44 55

6 ⇒ 17 26 31 44 54 77 93 55

7 ⇒ 17 26 31 44 54 55 77 93

Final List: 17 26 31 44 54 55 77 93

9. Literation: List: 54 26 93 17 77 31 44 55

1 ⇒ 26 54 17 77 31 44 55 93

⋮

* in each iteration, the sorting pass throug the list repeatly and compare the element and swap them. the process continue until the list is sorted.

n^{th}

Final: 17 26 31 44 54 55 77 93

code example:

```
void sort (int arr[ ] , int bob){
int i,j;
for ( i = 0; i < bob -1 ;i ++){
    For (j=0 , j < bob -1; j++){
        if ( Arr[j] > Arr[j+1]){
            swap ( & Arr[j] , & Arr[j+1]);
} } } }
```

i: 0

j: 0 to n^{th}

10. Given the following algorithm, what is its Big-Oh complexity? Which algorithm does this code resemble?

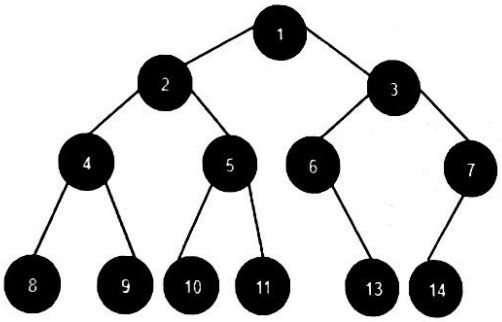
```
void mystery(int* a, int n){
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n - 1; j++) {
            if (a[j] > a[j + 1]) {
                swap(&a[j], &a[j + 1]);
            }
        }
    }
}
```

The given code is a bubble sort algorithms.
The Big-Oh complexity is $O(n^2)$

11. Why do computer scientists typically rely on 'worse-case' analysis?

The wort case scenario gives an "upper bound" for how long some of the operation will take to run

12. Given the following tree, write a recursive pseudo-code algorithm that would print the tree in the follwing order: 8 9 4 10 11 5 2 etc.



12 => back page.

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct node* left;
    struct node* right;
} node_t;

struct
```

12.

```
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node* leftC;
    struct node* rightC;
};

struct node* newNode(int data){
    struct node* temp_node = malloc(sizeof(struct node));
    temp_node -> data = data;
    temp_node -> leftC = NULL;
    temp_node -> rightC = NULL;

    return temp_node;
}

void sort(struct node* node) {
    if (node == NULL){
        return -1;
    }
    sor(node -> leftC);
    sort(node -> rightC);
    Printf("%d", node -> data);
}
```

13. Some engineers at Google want you to implement a graph structure of the subway system in Boston showing the connections between each station. They are wondering what data structure to use to represent the graph, given that most stations do not connect to many others. Which graph data structure do you use and why?

We would use an adjacency list graph b/c the set of adjacent vertices to a given vertex quicker than an adjacent matrix. The adjacent matrix is also too sparse for the data structure.

14. (10 pts)Given a directed graph describe what algorithm you would use to detect if two nodes are connected. Give your algorithm a name, and describe how it is used to traverse the graph and detect if the two nodes are reachable. A picture to go along with your explanation would be helpful. What would be the running time of your algorithm?

For this question, I decided to choose DFS because it's search can pull / present all possibities of the graph. It also check for the best / count the number of all possible way.

The running time of DFS is O(V + E), where V is vertices and E is Edges.

★ codes on scrap paper.

↳★ The code example, I use OH video example. It's a mixure of OH codes and code I found online. (a univerty from HK / UK)

```
DFS-reachable(g) {
    for all v ∈ g.vertices {
        setLable(v, unvisited);
    }
    for all e ∈ g.Edges {
        setLable(e, undiscoverd);
    }

    for a v ∈ g.vertices
        if (v.lable == unvisited) {
            visit_helper(v, G);
        }
    }
}
```

```
visit-helper(v, g) {
    setLable(v, visited);
    for all e ∈ v.incidentEdges {
        if (e.Lable == undiscovered) {
            w = adjacent-vertex(v, e);
            if (w.Lable == unvisited) {
                setLable(e, Discovered);
                visit-helper(w, g);
            }
            if (w.Lable == visited) {
                setLable(e, Back);
            }
        }
    }
}
```

15. Suppose you are trying to determine if two nodes are reachable in a graph. Give an example and a short explanation of when BFS would perform better and an example of when DFS would perform better when trying to determine if the two nodes are reachable. You can draw a graph accompanied with a short explanation for each case.

Explanation:

BFS is general best when we try to approach the depth of the tree can be vary, which only required to search the part of the tree.

DFS is generally best when we try to exhaust all the choices b/c of the graph nature and going in depth, discovering the longest depth path between two nodes.

example:

↳ in the book.

16. A major airline decides it wants to board passengers passed on their priority status. The higher the priority passenger rating, the earlier they should board (assume there are no ties). What data structure should this airline use? How long will it take to board N passengers in Big-Oh?

The data structure that the airline use is priority queque data structure. It will takes $O(n \cdot \log n)$ time complexity to board N passenger.

Example:

BFS:

GPS navigation syster is a perfect examples of the usage BFS implement. it take the user location to the source node and destination to the destination node on the graph.

* Dijkstra algorithms can be also interpreted as BFS.

DFS

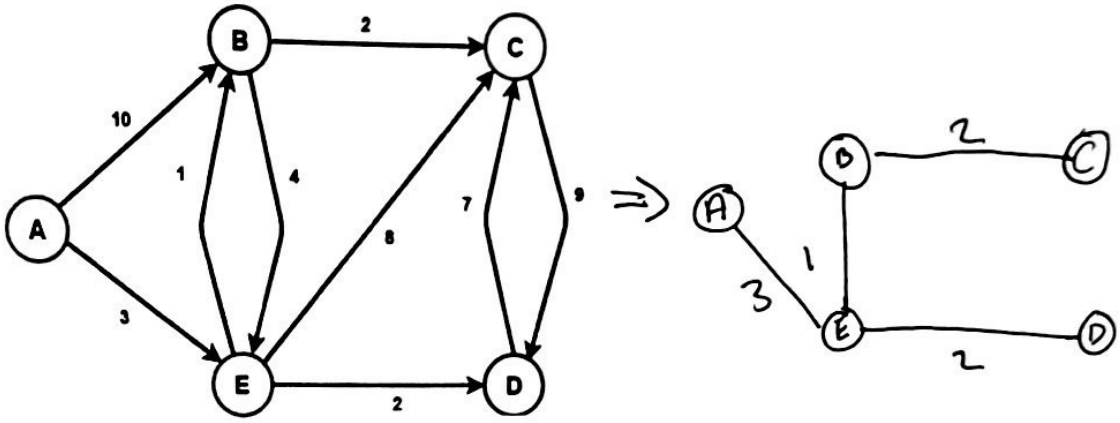
* Back tracking algorithms can be also interpret as DFS

* Assignment 2 - stack is a perfect example of DFS.

Between BFS and DFS, I would choose DFS because it can pull all the possibilities of the graph.

By Comparing BFS and DFS, the biggest advantage DFS is that is used less memory then BFS. Whereas BFS advantage is the faster that a DFS. Therefore, BFS is more efficient when we trying to determine if two node are reachable in a graph.

17. Compute the shortest path from node A to each of the other nodes. You can use the table to help compute the final answers.



Node	Distance	Last Edge
A	0	A
B	4	E
C	6	B
D	5	D
E	3	C

Dikjskra Algo.

Dis(A) = distance
Pre(A) = previous.

Iterator	visited	Dis(A)Pre(A)	D(B)P(B)	D(C)P(C)	D(D)P(D)
1	A	0	10, A	∞	∞
2	AE	0	4, E	11, E	5, E
3	AEB	0	4, E	6, B	5, E
4	AEBD	0	4, E	6, B	5, E
5	AEBDC	0	4, E	6, B	5, E

D(E)P(E)
3, A
3, A
3, A
3, A
3, A

18. (Bonus). Write a joke.

Original:"

me: Hi, I'm Alex.!

↳ Text from classmate: hey Alex!...

also me: Who's is Alex?"

Online joke:

"This grave looks overcrowded. People must be dying to get in."

(Scratch Paper–indicate clearly if I should grade something here as needed)