DATA STRUCTURES WORKBOOK

by PROFESSOR KATHERINE CHUANG

OPERATIONS

Inserting, deleting, and searching

TRADEOFFS

The best structure for your application

ALGORITHMS

Searching and sorting

ISSUE 2 | FALL 2020

Katherine Chuang

Data Structures Workbook

a collection of problem sets

Brooklyn College of CUNY

Fall 2020

Printed in the United States of America

Version 0.1

Brooklyn College of C.U.N.Y.
Brooklyn, NY 11210 Ordering Information:

`https://store.drkat.dev`

Contents

Frontmatter

Preface

To the student

This workbook was created primarily for the **CISC 3130 Data Structures** course at Brooklyn College. It contains a collection of problems that commonly appear in introductory data structures topics.

General Advice: Problem solving is knowing exactly the problem you're solving and breaking down larger problems to smaller problems. Check carefully the description presented before answering. Don't give up immediately if you can't get figure out the answer. Make notes for yourself on what didn't make sense, your hypotheses, and also what does make sense. Your process to problem solving can include retracing your steps, reading textbooks, or chatting with someone.

The process of problem solving: Approach each problem as an exploration. Read each question carefully, break the problem into smaller parts and analyze the solution. Working problems in this book will give you practice in using the techniques for breaking down problems and analyzing solutions. The problems in this book will also prepare you for similar problems on exams.

Organization of the book

The book is divided into several parts.

Part 1 contains sets of problems on a review of fundamental concepts
Part 2 contains sets of problems using data structures and performing operations on data structures.
Part 3 contains sets of problems on sorting algorithms.
Part 4 contains sets of problems on searching algorithms.
Part 5 is an appendix of helpful guides to quickly look up information.

How to use this workbook

Most of the problems should take you only a few minutes or less. If you cannot do a problem in 10 to 15 minutes, seek help from a tutor.

Acknowledgements

I would like to thank the faculty and staff at Brooklyn College who thoughtfully pointed me into the right direction for producing this book. Particularly to the fine folks of the Department of Computer & Information Science and the university librarians. Special thanks to Professor Yedidyah Langsam, the chair of the CIS department for the inspiration to create this manual, and providing a wonderful example of a workbook to inspire the creation of this one. Many thanks Murray Gross for supplying helpful guidance through the typesetting process. Many thanks to friends and family who provided emotional support and encouragement through the process of putting this together.

Part I

Review of prerequisites

Chapter 1

Growth of Functions

1.1 Logarithms Review

1. $2 * 2 = ?$

2. $2 * 2 * 2 = ?$

3. Find the logarithm of 8. Using the function $log_2 8 = x$, what is x?

4. Write an equivalent exponential statement for $log_2 16 = 4$

1.2 Finding the Dominant Term

Given each expression, find the dominant terms.

5. $100n + 0.01n^2$

6. $10n + 20n + 5n$

1.3 Expressing the Rate of Growth

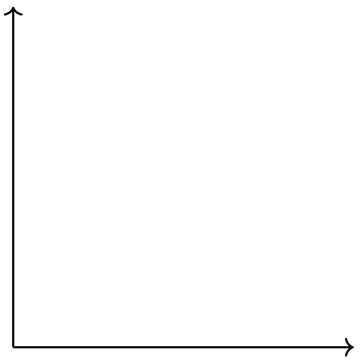
7. What does it mean for a function to be considered growing linearly? Express the answer in a function form, $f(n) = ?$

8. For the following formula $f(n) = 3n + 1$, as n increases from 1 thru 10, what is the resulting output? Write the numbers as a series.

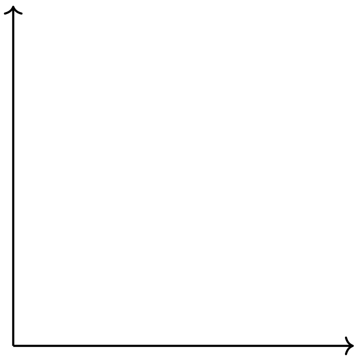
9. For a function to be considered to grow at a constant rate, what does that mean? Express the answer in a function form, $f(n) = ?$

1.4 Plotting the series

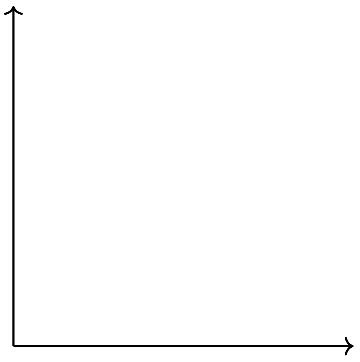
10. n is a number expressed by the pattern $f(n) = n$. As n increases from 1 thru 10, what is the resulting output? Write out the series it generates and plot it on a chart.



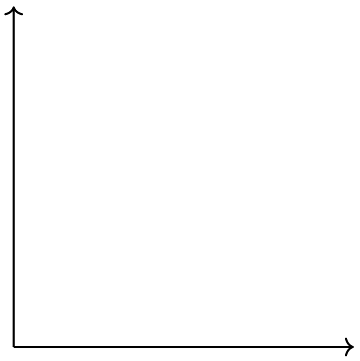
11. A function is defined as $f(n) = n + 1$. What will the resulting chart look like when plotted on a two dimensional chart?



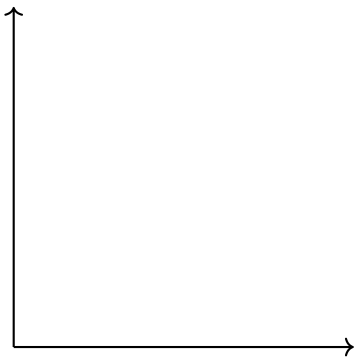
12. A function is defined as $f(n) = 5$. What will the resulting chart look like when plotted on a two dimensional chart?



13. A function is defined as $f(n) = n^2$. What will the resulting chart look like when plotted on a two dimensional chart?



14. For the following formula, as n increases from 1 thru 10, what is the resulting output? $f(n) = n^3$



1.5 Ranking the growth of functions

15. Rank these functions according to their growth, from most expensive to the least expensive:

$$n \quad n^3 \quad 1 \quad \tfrac{3}{2}n \quad n^2 \quad 2n \quad log_2 n$$

16. Rank these functions according to their growth, from least expensive to the most expensive:

$$4n \quad 6n^3 \quad 64 \quad 8n^2 \quad nlog_2 n \quad 82n \quad log_2 n$$

Solutions to the Exercises

Answer 1 4

Answer 2 8

Answer 3 x = 3

Answer 4 $2^4 = 16$

Answer 5 $n + n^2$, or $O(n^2)$

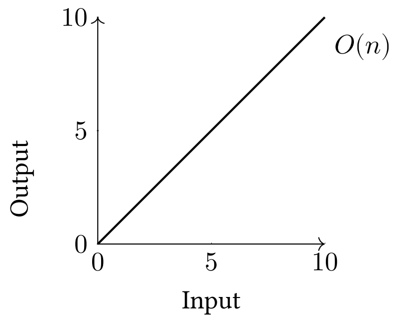
Answer 6 $35n$, or $O(n)$

Answer 7 Linear growth grows by the same amount at each step. $f(n) = n$.

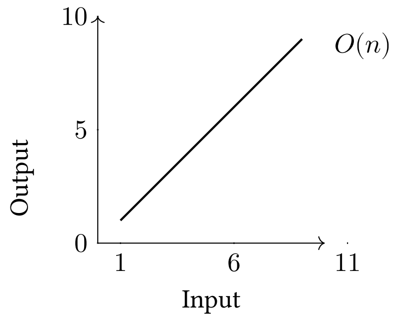
Answer 8 For input = 1, 2, 3, 4, 5, ..., the output is 1, 4, 7, 10, 13, 16, ...

Answer 9 $F(x) = C$, where C is a constant real number.

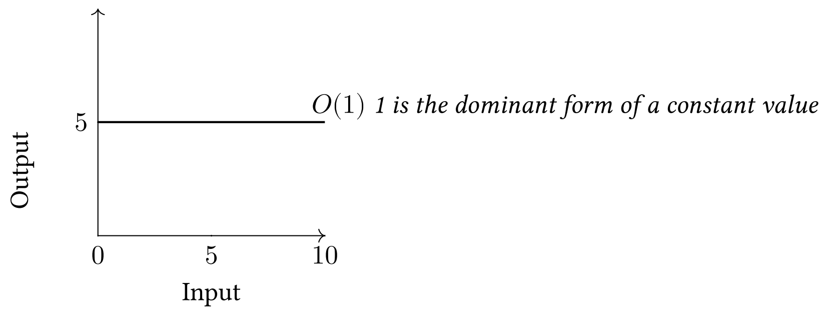
Answer 10 This function is considered a series of linear growth, because for each input $input = 1, 2, 3, 4, 5, ...$ it returns the same value as the output, $output = 1, 2, 3, 4, 5, ...$. This can be represented by $O(n)$.



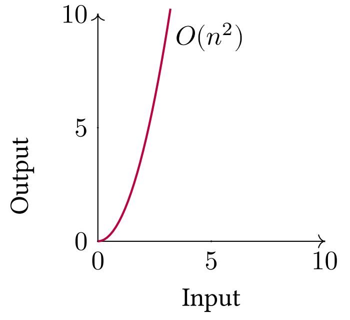
Answer 11 For $input = 1, 2, 3, 4, 5, ...$, the output is $2, 3, 4, 5, 6, ...$



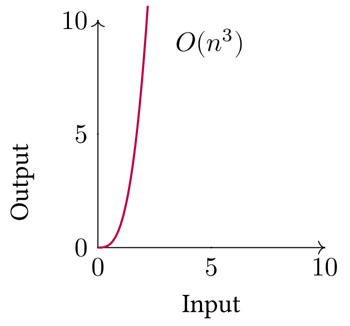
Answer 12



Answer 13



Answer 14



Answer 15 The slowest one has the steepest curve.

Most Expensive n^3 n^2 $2n$ $\frac{3}{2}n$ n $log_2 n$ 1 *Least Expensive*

Answer 16

Least Expensive 64 $log_2 n$ $4n$ $nlog_2 n$ $8n^2$ $6n^3$ $82n$ *Most Expensive*

Chapter 2

Loop Analysis

2.1 Identify the Time Complexity

Identify the Time Complexity for the following descriptions. Write your answer in Big O notation.

1. A *for* loop iterates through an array of n elements, where the iterator i starts from 0 and increments at index $i+1$. How many steps will it take to iterate through that array?

2. A *while* loop iterates through an array of n elements. With each pass the a counter variable i incrementally increases $i = i + 2$. How many steps will it take to iterate through that array?

3. A *while* loop iterates through an array of n elements. With each pass a counter variable i incrementally increases $i = i * 2$. How many steps will it take to iterate through that array?

2.2 Time Complexity (Java Code)

Identify the Time Complexity for the following descriptions. Write your answer in Big O notation. Assume for the following that n represents the number of elements of an arbitrary array.

4.

```java
for(int i=0;i< n-1;i++) {
  ..
  ..
}
```

5.

```java
int i = 0;
while( i < n) {
  ..
  i++;
}
```

6.

```java
for(int i = 1; i < = n; i*=2){
  sum++;
}
```

7.

```java
double product = 1.0;
for (int i = 1; i <= n; i *= 2) {
   product *= i;
}
```

8.

```java
for (int i=1; i<= n; ++i){
  for (int j=1; j<n; j*=2){
    sum++;
  }
}
```

Solutions to the Exercises

Answer 1 $O(n)$

Answer 2 $O(n)$

Answer 3 $O(logn)$

Answer 4 The for loop takes n time to complete and so it is O(n). For example, $(1 + 2 + 3 + 4 + 5 + ...n)$ ”=” n.

Answer 5 $O(n)$

Answer 6 When a for loop takes n time and its index i increases or decreases by a multiple, the upper bound is $O(logn)$

Answer 7 $O(logn)$

Answer 8 $O(nlogn)$ where the outer loop is n, and inner loop is $logn$

Chapter 3

Recursion

1. What value is returned by the method call $sum(5)$?

```java
public int sum(int n)
{
  if (n == 1)
    return 1;
  else
    return n + sum(n - 1);
}
```

2. What value is returned by the method call result(5) ?

```java
public int result(int n) {
 if (n == 1)
   return 2;
 else
   return 2 * result(n - 1);
}
```

3. What value is returned by the method call $f(6, 8)$?

```java
public int f(int k, int n) {
 if (n == k)
   return k;
 else
   if (n > k)
     return f(k, n - k);
   else
     return f(k - n, n);
}
```

4. Write a recursive function to reverse a string. Write a recursive function to reverse the words in a string, i.e.,“cat is running” becomes “running is cat”.

5. Compute the sum of natural numbers until N.

6. Compute the Factorial of a number N given this formula $(N)! = N * (N - 1) * (N - 2) * ... * 1$. Write the code example

Solutions to the Exercises

Answer 1 15

Answer 2 32

Answer 3 2

Answer 4

```
    public String reverse(String s) {
      // When string is empty, recursion stops
      if (s.isEmpty()){
        System.out.println("String is now empty");
        return s;
      }

      //Calling the reverse method recursively
      System.out.println("String to be passed in Recursive Function: " + s.substring(1));

      return reverse(s.substring(1)) + s.charAt(0);
  }
```

Answer 5

```
public int sumOf(int n) {
  if (n == 1){
     return n; // we don't need to recursively call after 1
  }
  //Calling Function Recursively
  return sumOf(n-1) + n;
}
```

Answer 6

```
public int factorial(int n) {
  if (n == 1){
     return n; // we don't need to recursively call after 1
  }
  //Calling Function Recursively
  return n * sumOf(n-1);
}
```

Chapter 4

Arithmetic Notations

4.1 Basics

1. Convert each of the formulas to its prefix and postfix notations.

Infix	**Prefix**	**Postfix**
$1+1$		
$3+9$		
$6+8$		

2. Convert each of the formulas to its prefix and postfix notations.

Infix	**Prefix**	**Postfix**
$1*1$		
$3*9$		
$6*8$		

4.2 Pseudocode

Algorithm for Evaluating Postfix

```
1. Create an empty stack and scan postfix expression from left to right
2. If element is an operand, push it into the stack
3. If the element is an operator, pop twice and get A and B (calculate B and A) and push result back to the stack
4. When expression is ended, the value in stack is the final answer
```

Algorithm for converting infix to postfix

```
1. Create an empty stack. Read infix expression from left to right
2. For each character c in the input stream:
3. case c where :
    a. operand ⟶ print operand
    b. '(' ⟶ push c on the stack
    c. ')' ⟶ pop operators from stack to the output until '(' is popped, do not output either of parentheses. Stop before lower precedence operators or a '('.
    d. operator ⟶ Push operator on the stack
4. Pop from stack and output tokens until the stack is empty
```

4.3 Infix to Postfix

For the following problems, find the postfix notation form of the given infix expression.

Three Operands

3. $9 - 6 - 2$

Char	Postfix (Output)	Stack	Rule

4. $3 + 1 * 9$

5. $(1 + 2) + 3$

Char	Postfix (Output)	Stack	Rule

6. $(3 + 1) * 9$

Char	Postfix (Output)	Stack	Rule

7. $(5 - 2)/9$

8. $4\ /2 - 7$

9. $A * (B + C)$

10. $(A + B) * C$

Fours Fives

11. $1 + 2 - 3 + 4$

12. $5 - 9 + 1 * (4 + 6)$

13. $A + (B * C)$

14. $(A + (B * C))/(D - E)$

15. $(A + B) * (C + E)$

16. $(2 * 1)/5 - (4 * 3)$

17. $(8 - 1) * (3 - 4)$

18. $5 - 9 + 1 * (4 + 6)$

19. $(A - B) * (C - E)$

4.4 Postfix to Infix Practice

For the following, write the given expression in infix notation.

20. $1\ 2\ +$

21. $7\ 1\ +\ 2\ +$

22. $A\ B\ C\ D\ *\ -\ E\ /\ +\ F\ +\ G\ H\ /\ -$

23. $A\ B\ *\ C\ /\ D\ E\ +\ F\ G\ H\ /\ *\ -\ +$

4.5 Postfix Evaluation

Evaluate the postfix notation:

24. $31 + 9*$

25. $3\ 1\ +$

26. $3\ 1\ *$

27. $2\ 4\ 9\ *\ -$

28. $5\,4\,6\,1 - + *$

29. `4 3 2 - +`

30. $8\,7\,2\,/\,-$

31. $3\,8\,5\,1 +\; + -$

32. $54\,6 * 7\,4\; -\; - 9\,35\,5\; +\; * +$

Solutions to the Exercises

Answer 1

Infix	**Prefix**	**Postfix**
$1+1$	$+1\ 1$	$1\ 1+$
$3+9$	$+3\ 9$	$3\ 9+$
$6+8$	$+6\ 8$	$6\ 8+$

Answer 2

Infix	**Prefix**	**Postfix**
$1*1$	$*1\ 1$	$1\ 1*$
$3*9$	$*3\ 9$	$3\ 9*$
$6*8$	$*6\ 8$	$6\ 8*$

Answer 3 $9\ 6-2-$

Char	Postfix (Output)	Stack	Rule
9	9		3a. print operand
-	9	-	3d. operator pushed to stack
6	9 6	-	3a. print operand
-	9 6	- -	3d. operator pushed to stack
2	9 6 2	- -	3a. print operand
	9 6 2 - -		

Answer 4 $3\ 1\ 9*+$

Answer 5 $1\ 2+3+$

Char	Postfix (Output)	Stack	Rule
(		(	3b
1	1		3a
+		(+	3d
2	1 2	(+	3a
)	1 2 +		3c
+		+	3d
3	1 2 + 3	+	3a
	1 2 + 3 +		

Answer 6 $3\ 1+9*$

Char	Postfix (Output)	Stack	Rule
(		(	3b
3	3	(	3a
+	3	(+	3d
1	3 1	(+	3a
)	3 1 +		3c
*	3 1 +	*	3d
9	3 1 + 9	*	3a
	3 1 + 9 *		

Answer 7 $5\ 2 - 9/$

Answer 8 $4\ 2/7-$

Answer 9 $A\ B\ C + *$

Answer 10 $A\ B + C*$

Answer 11 $1\ 2 + 3 - 4+$

Answer 12 $5\ 9 - 1\ 4\ 6 + *+$

Answer 13 $A\ B\ C * +$

Answer 14 $A\ B\ C * +D\ E - /$

Answer 15 $A\ B + C\ E + *$

Answer 16 $2\ 1 * 5/4\ 3 * -$

Answer 17 $8\ 1 - 3\ 4 - *$

Answer 18 $5\ 9 - 1\ 4\ 6 + *+$

Answer 19 $AB - CE - *$

Answer 20 $1 + 2$

Answer 21 $7 + 1 + 2$

Answer 22 $A + (B - C * D)/E + F - G/H$

Answer 23 $A * B/C + D + E - F * G/H$

Answer 24 36

Answer 25 4

Answer 26 3

Answer 27 -34

Answer 28 45

Answer 29 5

Answer 30 4.5

Answer 31 -11

Answer 32 673

Part II

Problem Sets on Data Structures

Chapter 5

Working with Arrays

5.1 True or False?

1. *True or false?* An array automatically expands in size to accommodate the number of items stored in it.

2. *True or false?* If data is sorted in ascending order, it means it is ordered from lowest value to highest value.

3. *True or false?* Adding more elements to an array that is already full can extend the size of that array.

4. *True or false?* An array A is created to hold 10 elements. 15 values from list L are added one by one into array A using a for loop. The 11th through 15th elements of L are added starting from the first position in A. At the end of the loop, you can access values of all 15 elements from array A.

5.2 Accessing Elements from Arrays

5. In the following array, what would be the the element printed at position 5?
Assume the array index starts at 0.

E	X	A	M	P	L	E

6. Using pseudocode, write the steps for iterating through an array. Print the last three elements.

7. Copy every other element to a new array.

8. Write pseudocode that finds the smallest value from this array. $A = \{3, 4, 5, 7, 9, 1, 6\}$

3	4	5	7	9	1	6

5.3 Inserting Elements to Arrays

9. Write pseudocode that inserts the value ‘2’ into array A at the correct position, resulting in an ordered array. Use the linear search strategy. $A = \{1, 3, 5, 7, 9, 11\}$

10. Write pseudocode that inserts the value ‘84’ into array A at the correct position, resulting in an ordered array. Use the linear search strategy. $A = \{3, 15, 16, 28, 49, 70, 111\}$

5.4 Deleting Elements from Arrays

11. Write pseudocode that removes the value ‘2’ from array A. Use the linear search strategy to move through the array $A = \{1, 2, 3, 5, 7, 9, 11\}$ You may assume that there exists a magical function insertBefore() that takes as inputs the value to insert, as well as the position to insert at while shifting subsequent values over to the right and resizing the array.

12. Write pseudocode that removes the value ‘17’ from array A. Use the linear search strategy to move through the array. $A = \{1, 2, 3, 5, 7, 9, 11\}$ You may assume that there exists a magical function insertBefore() that takes as inputs the value to insert, as well as the position to insert at while shifting subsequent values over to the right and resizing the array.

13. Write pseudocode that removes the value '9' from array A. Use the linear search strategy to move through the array.
$A = \{1, 2, 3, 5, 7, 9, 11\}$

14. Write pseudocode that removes the value '39' from array A. Use the linear search strategy to move through the array.
$A = \{4, 7, 9, 11, 22, 28, 39, 70, 105\}$

5.5 More on Arrays

15. *True or false?* If data is sorted in descending order, it means it is ordered from lowest value to highest value.

16. Rearrange an array such as $A = 3, 1, 5, 5, 4$ to *ascending* order.

17. Write pseudocode that merges the sorted array A and sorted array B into a resulting sorted array C.
$A = \{1, 3, 5, 7, 9\}$, $B = \{2, 4, 6, 8\}$, $C = \{?\}$

Solutions to the Exercises

Answer 1 False. An array is declared with fixed size. Linked lists expands in size to accommodate new items

Answer 2 True. Ascending order means sorted order from lowest value to highest value.

Answer 3 False. An array size cannot change dynamically. Array sizes are fixed, and the only time you can define the size is when it is created.

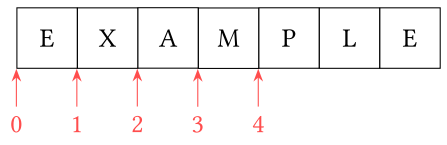
Answer 4 False. The 11th to 15th values from L will override the original elements 1 through 5, so you cannot access them. The figure below shows how the resulting array would look like.

11	12	13	14	15	6	7	8	9	10

Sample code might be:

```
  int[] A = new int[10];
  for (int i = 0; i < L.size; i++)
  {
    if(i < 10)
    {
      A[i] = L[i];
    }
    else
    {
      i = i - 10; // change indices
      A[i] = L[i];
    }
}
```

Answer 5 The 5th element has index 4.



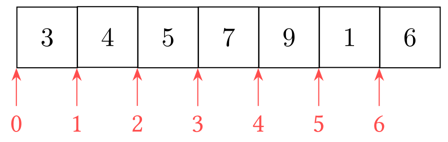
Answer 6 To iterate through the array, we can use a for loop.

```
printTrigger = A.length-3;
for i->0 to A.length:
  if i > printTrigger
    return A[i]
```

Answer 7

```
  public int[] copyArray(int[] n)
  {
    int[] newArray = new int[n.length];
    for(int i=0; i < n.length - 1; i++)
    {
      newArray[i] = n[i];
    }
    return newArray;
}
```

Answer 8 One solution would need to have a reference to the interim smallest value while looping through each element of the array.



Using this sample pseudocode and the sample array given,

```
  min = A[0]
  for i->1 to A.length:
    if A[i] < min
      min = A[i]
  return min
```

We would start with $min = 3$ because the index 0 of Array A has value 3. Then the next step would be to search through the array until a value smaller than the current min is found.
The resulting output from this search should return 1.

Answer 9

```
key = 2
for item in A:
  if key > A[i]
    A.insertBefore(key, i)
```

Answer 10

```
key = 84
for item in A:
  if key > A[i]
    A.insertBefore(key, i)
```

Answer 11

```
key = 2
for item in A:
  if key == A[i]
    A.remove(key)
```

Answer 12

```
key = 2
for item in A:
  if key == A[i]
    A.remove(key)
```

Answer 13 No answer provided.

Answer 14

```
key = 39
for item in A:
  if item == 39
    remove item from A
```

Answer 15 False

Answer 16 Using selection sort strategy:

- Find the smallest value in tha array

- Move smallest value into the first slot.
- Repeat for other elements.

```
temp = 0;
for i->0 to A.length:
  temp = A[i]
  for j->0 to A.length:
    if A[j] < A[i]
     temp = A[j]
     A[i] = temp

```

As it turns out, this is the *selection sort* strategy.

Answer 17 While keeping track of the first element of each array A and array B, compare to see which is smaller or larger. The smaller of the two should be saved into C at the end. This way we can maintain the sorted order.

Chapter 6

Linked List

6.1 Short Answer

1. A simple linked list does not support direct ____ access.

a) Indexed b) Implemented c) Inversed d) Invented

2. Which of the following statements are true about the linked list data structure when compared with arrays? Select all that apply.

a) Arrays are better when there's an odd number of elements.

b) It is easier to insert and delete elements in the middle of a linked list.

c) Random access is not allowed in a typical implementation of a simple linked list.

d) The size of an array has to be pre-decided, linked lists can change their size at any time.

3. *True or false?* A linked list automatically expands in size to accommodate the number of items stored in it.

4. *True or false?* You can reference nodes in a linked list using index numbers.

5. *True or false?* If data is sorted in ascending order, it means it is ordered from lowest value to highest value.

6. *True or false?* The size of a linked list is stored as a property of a linked list.

6.2 Accessing elements in a linked list

7. Describe the steps to print out elements in a simple linked list.

8. Write out the pseudo code to find the size of a linked list.

9. Write a recursive method for calculating the size of a linked list.

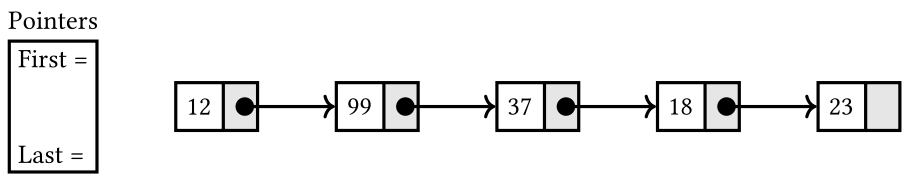
10. Write a method that converts a lower case string to uppercase.

11. Describe at least three differences between arrays and linked lists.

12. With the design of a double ended linked list such as the following code block, you can insert to the end of the list. Describe the features of the linked list makes this possible.

```java
class LinkedList {
  private Node first;
  private Node last;

  public void LinkedList(){
    first = null;
    last = null;
  }

  public void insertLast(int i){
    Node newNode = new Node(i);
    if( isEmpty() )
      first = newNode;
    else
      last.next = newNode;

    last = newNode;
  }
}
```

13. Draw lines from the first and last pointers to the correct nodes of the double ended linked list pictured below.



6.3 Doubly Linked Lists

14. With the design of a doubly linked list, you can traverse forwards and backwards through the list of nodes. Describe what makes this possible.

15. What is something that is possible with a doubly linked list but not with a simple linked list?

Solutions to the Exercises

Answer 1 a. Index

Answer 2 B,D

Answer 3 True - linked lists expands in size to accommodate new items

Answer 4 False

Answer 5 True

Answer 6 False

Answer 7

```
    for node in linkedList
      display node
      move to next node

```

Answer 8 The number of elements in a linked list can be determined by initializing a counter to 0. Then incrementing the counter at each step of traversal of the linked list.

Answer 9 Lets give the size of the linked list a number n. The number of elements in a linked list can be determined by making a recursive call to one size smaller in each preceeding step so that it becomes $1 + (n - 1)$ and so on until the chain of recursive calls becomes $1 + 1 + 1 + ... + 1$ until the series of the sum equals n.

Answer 10

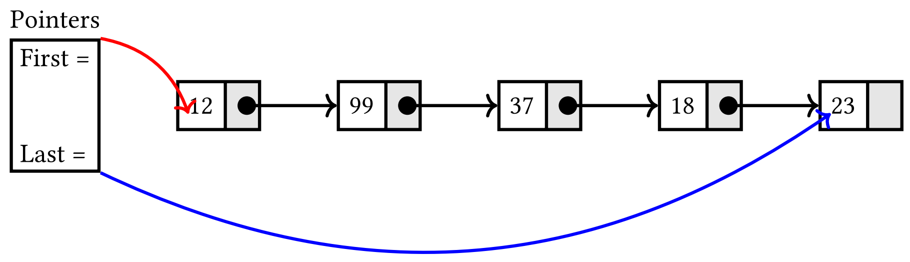
```
public static void toUpperCase(StringNode str) {
    StringNode trav = str;
    while (trav != null) {
        trav.ch = Character.toUpperCase(trav.ch);
        trav = trav.next;
    }
}
```

Answer 11 Three differences between arrays and linked lists.

1. Arrays are fixed size
2. Linked lists are dynamically allocated size
3. Linked lists are great when you don't know the size of data you've got to work with.

Answer 12 The double ended linked list knows where the end of the list is, and would be able to allow you to start traversing from the end.

Answer 13 Pointers



Answer 14 Each node a doubly linked list has a pointer to the next node and the previous node. This allows for both forwards and backwards traversal. Additionally, the doubly linked list recognizes where the end of the list is, and would be able to allow you to start traversing from the end.

Answer 15 Traversing backwards is possible with a doubly linked list. Traversing backwards is not possible with a simple linked list.

A doubly linked lists has a connection to the previous node in addition to the next node. The class for a doubly linked list node might look like the following:

```java
public class Node {
  int value;  // this is data that the node can store
  Node next;  // this the reference to the next node
  Node prev;  // this is the reference to the previous node. not in simple linked list node.
}

```

Chapter 7

Stacks & Queues

7.1 Multiple Choice

1. Stack is a linear data structure where the order is:

a) Last in, finally out (LIFO)

b) First in, first out (FIFO)

c) Last in, first out (LIFO)

d) Later instance, for output (LIFO)

2. Basic operations of a stack are:

a) Push, reveal, peek

b) Pop, push, peek

c) Push, append, display

d) Pop, print, reverse

3. What are the elements in queues ordered?

a) First in, first out (FIFO)

b) First in, filled out (FIFO)

c) Last in, first out (LIFO)

d) First in, fouled out (FIFO)

4. Can a stack be implemented by a linked list? Describe the linked list.

a) Yes   b) No

5. *True or false?* Pop() works only on empty stacks.

6. *True or false?* A queue can be implemented by a linked list. Explain why or why not.

7. How is a queue different from stack in how is it implemented in the number of pointers needed?

7.2 Inserting and Removing from Stacks

8. Insert these values into a stack. Then print the output. What is the output? $S = \{22, 44, 66, 88\}$

9. Insert these values $33, 55$ into a stack. Then pop one then push in $55, 66, 99$. What is the output when you print elements from the stack?

7.3 Inserting and Removing from Queues

10. Insert these values into a queue. Then print the output. What is the output? $S = 22, 44, 66, 88$

11. Insert these values $33, 55$ into a queue. Then remove one then push in $55, 66, 99$. What is the output when you print elements from the queue?

Solutions to the Exercises

Answer 1 Last in, finally out (LIFO)

Answer 2 Pop, push, peek

Answer 3 First in, first out (FIFO)

Answer 4 Yes

Answer 5 False - can't pop an item off the stack when it's empty

Answer 6 Yes. Double ended linked list.

Answer 7 Stacks insert and delete from one end, needs only 1 pointer. Queues work with both ends so it needs two pointers.

Answer 8 $88, 66, 44, 22$

Answer 9 $99, 66, 55, 33$

Answer 10 $22, 44, 66, 88$

Answer 11 $33, 55, 66, 99$

Chapter 8

Trees

8.1 Parents and Children

1. How many children do parent nodes in a binary tree have?

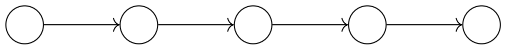
2. What is the height of a full, complete binary tree with 3 nodes?

3. Can a node in a binary tree have zero children?

4. What is the height of a full, complete tree with 7 nodes?

8.2 Comparing to other structures

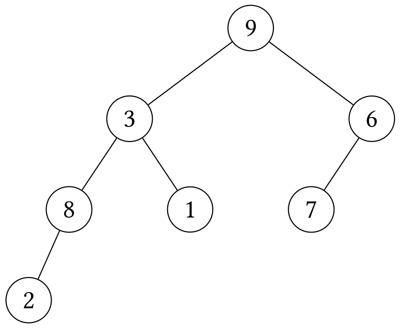
5. Can the following structure that resembles a linked list be considered a legal binary tree? Why or why not?



6. Find the height of a binary tree with 10 nodes for (a) a complete binary tree, and (b) where each parent has at most 1 child.

8.3 Analyze the structure

Use the tree pictured to answer the questions.



7. Is the pictured tree a legal binary tree?

8. In the tree pictured, which node is the root?

9. Which nodes are level 2 nodes?

10. Which nodes are leaves?

Solutions to the Exercises

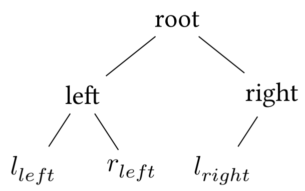
Answer 1 At most 2.

Answer 2 Height of the binary tree is 1.

Answer 3 Yes

Answer 4 Each node can have up to 2 children, which means the height for a binary tree is between $log2(N)+1 <= H <= N/2$.

A binary tree is full if every node has 0 or 2 children. A complete tree means reaching maximum children nodes for each subsequent level, which works out to $height = 2$ because height starts from leaves starting with value 0.



Answer 5 Yes

Answer 6 Complete tree would have height = 3. A tree where each parent has at most 1 child would have height = 9.

Answer 7 Yes.

Answer 8 9

Answer 9 Root is level 0, so level 2 nodes are 8, 1, 7.

Answer 10 2, 1, 7

Chapter 9

Binary Search Trees

9.1 Multiple Choice

1. *True or false?* In a binary search tree, the left child has a larger value than the parent node.

2. *True or false?* The order of nodes in a binary tree matters.

3. *True or false?* In a binary search tree, the parent node is bigger than both children.

9.2 Identifying Select Values

4. The following questions use the tree on the right.

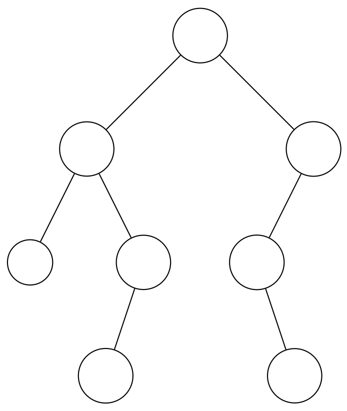
a) Which node of the binary search tree pictured contains the minimum value? Why?

b) Which node of the binary search tree pictured contains the maximum value? Why?

c) Which nodes are leaves?

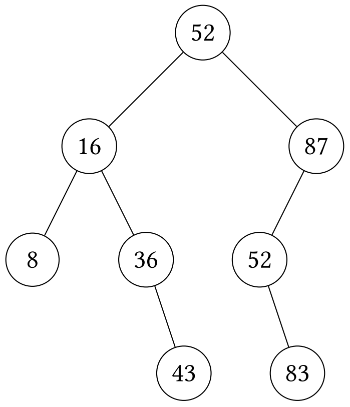
d) What is the height of '52'?

e) How many levels does the tree have?

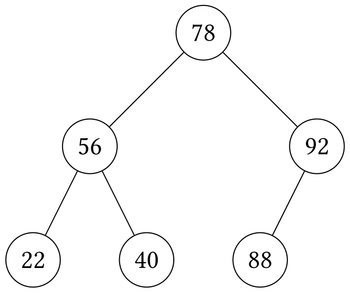


9.3 Removing elements from Binary Search Trees

5. Remove the value ‘43’ from the binary tree pictured. What is the new tree?



6. Remove the value ‘22’ from the binary from the binary tree pictured. What is the new tree?

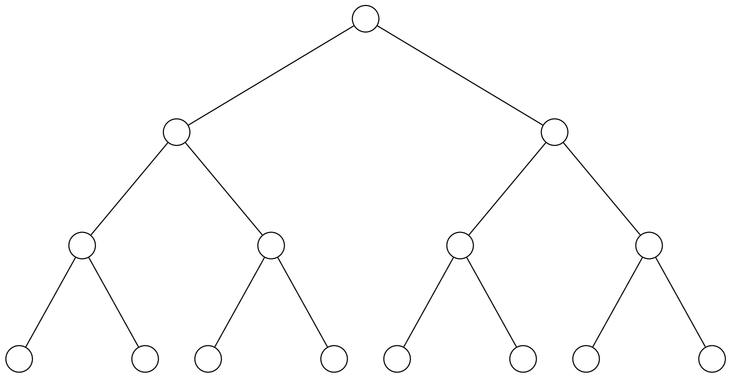


9.4 Inserting elements to Binary Search Trees

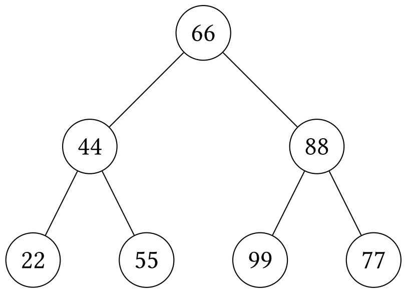
7. Insert the following values into the binary search tree. Redraw the tree for each of the values.
$A = \{51, 36, 1, 43, 87, 52, 83\}$

9.5 Identifying Parts of a Binary Search Tree

8. Find the minimum and maximum value in the Binary Search Tree structure pictured. Mark the node that is the minimum value and the node that is the maximum value.



9. Is the tree pictured a valid binary search tree? If not, rearrange the nodes, keeping the same root element.



Solutions to the Exercises

Answer 1 The right child is larger than the parent node. The left child is smaller than the parent node.

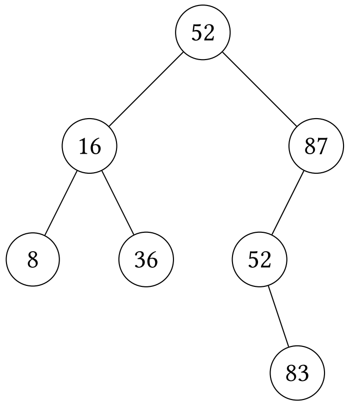
Answer 2 True. The order matters in a binary search tree. The left child is smaller than parent, and parent is smaller than the right child.

Answer 3 False. In a binary search tree, the parent is only bigger than the left child.

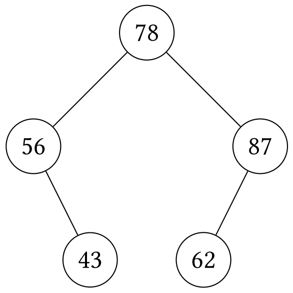
Answer 4 The minimum value of a binary search tree is the left most child. In the figure, it would be value 1.

The maximum value of a binary search tree is the right most child. In the figure, it would be value 87.

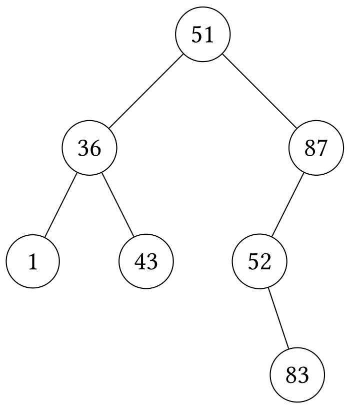
Answer 5



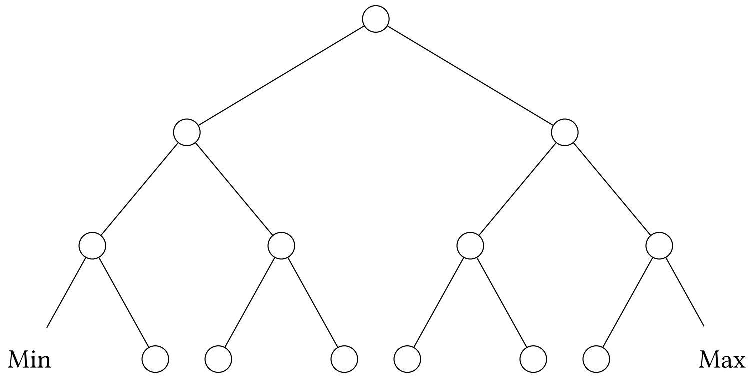
Answer 6



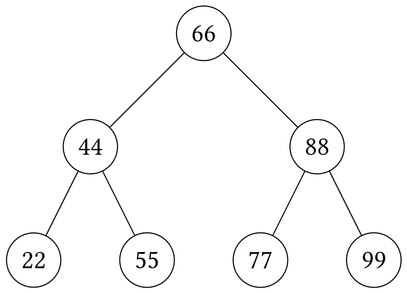
Answer 7



Answer 8



Answer 9



Chapter 10

Max Heaps

10.1 Basic Concepts of Max Heaps

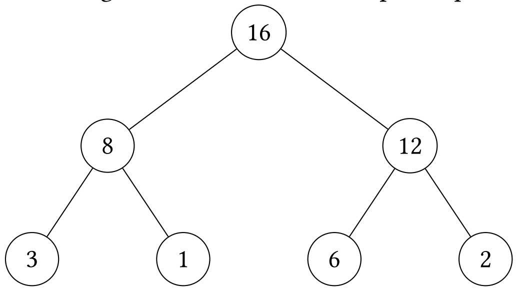
1. *True or false?* In a max heap, the left child is always smaller than the right child.

2. *True or false?* In a max heap, the parent is smaller than both children.

3. *True or false?* In a binary max heap, the order of nodes is completely arbitrary. You can have the values of nodes in any order.

4. *True or false?* In a binary max heap, the parent node is bigger than both children.

Use this image to answer the next couple of questions



5. Which node of the max heap pictured contains the maximum value?

6. Draw the max heap as an array.

10.2 Inserting to Max Heaps

7. Convert this input to a max heap, then show how the array looks like: $A = \{7, 3, 10, 2, 4, 1\}$

8. Insert the array elements $A = \{4, 2, 7, 5\}$ into a max heap. Redraw the array given input value *x* to the max heap.

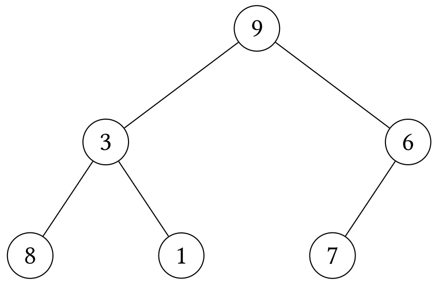
9. Insert the following values $A = 20, 10, 22, 8, 18$ into the binary max heap. Redraw the tree each time you insert one of the values.

10. Insert the following elements from array $A = \{4, 5, 1, 10, 32\}$ into a binary max heap in the same order as reading the elements one by one from the left to the right. Draw the final max heap.

11. Is the tree pictured to the right a legal binary max heap?

12. Which nodes are level 2 nodes?

13. Which nodes are leaves?



14. What is the height of a heap with 3 nodes?

15. What is the height of a heap with 6 nodes?

10.3 Searching Max Heaps

16. In a binary max heap, which of the two children has a larger value than the parent node?

17. The following questions use the tree on the right.

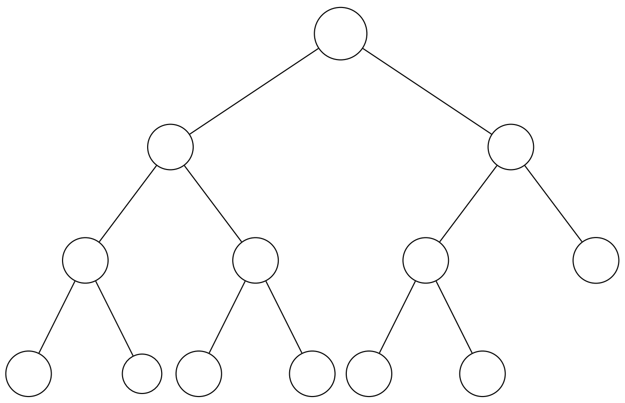
a) Which node of the max heap pictured contains the maximum value? Why?

b) If you remove the root node, which of the two children should become the new root? Why?

c) Which nodes are leaves?

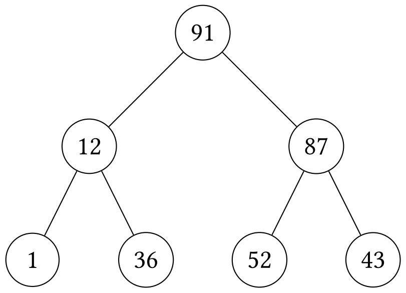
d) Is it a legal max heap?

e) Where would the next value go?

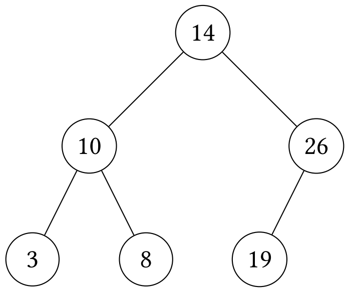


10.4 Removing from Max Heaps

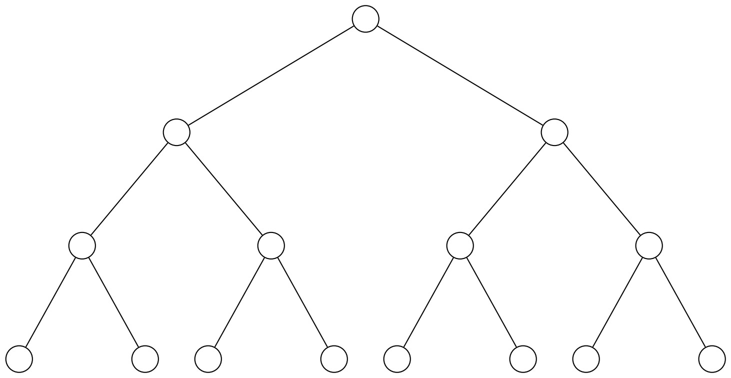
18. Remove the value '43' from the binary tree pictured. What is the new tree?



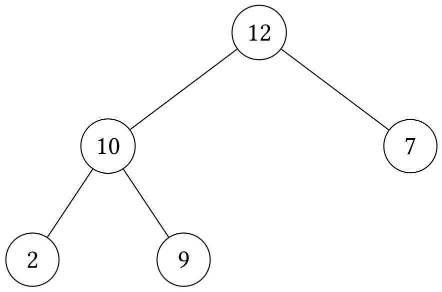
19. Remove the value ‘19’ from the binary from the binary tree pictured. What is the new tree?



20. Identify the node with the maximum value from the binary max heap pictured.



21. Is this binary tree a valid max heap? Why or why not?



Solutions to the Exercises

Answer 1 False

Answer 2 False

Answer 3 False

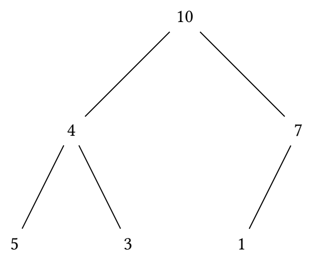
Answer 4 True. In a binary max heap the parent has a larger value than children.

Answer 5 Root

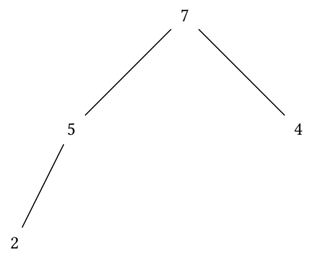
Answer 6

16	8	12	3	1	6	3

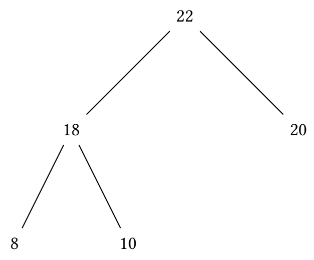
Answer 7



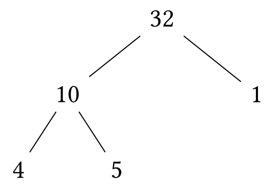
Answer 8



Answer 9



Answer 10



Answer 11 Yes.

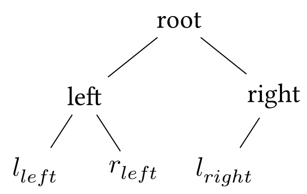
Answer 12 Root is level 0, so level 2 nodes are 8, 1, 7.

Answer 13 2, 1, 7

Answer 14 Height of the heap is 1.

Answer 15 Height for a heap is $log_2(N) + 1 <= H <= \frac{N}{2}$.

A heap is a full, complete tree, which means reaching maximum children nodes for each subsequent level, which works out to $height = 2$ because height starts from leaves starting with value 0.



Answer 16 Neither. The parent node is larger value than the children. The children can be in any order as long as it is smaller in value than the parent node.

Answer 17

a) Root. 132

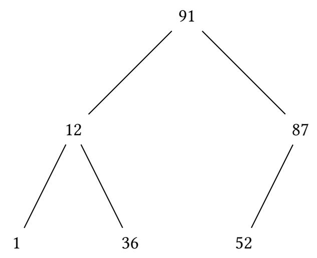
b) 82. Larger of the 2

c) 26,3,36,33,49,53,79

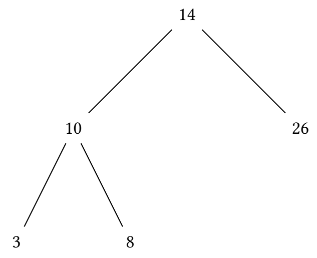
d) Yes

e) Left child of 79

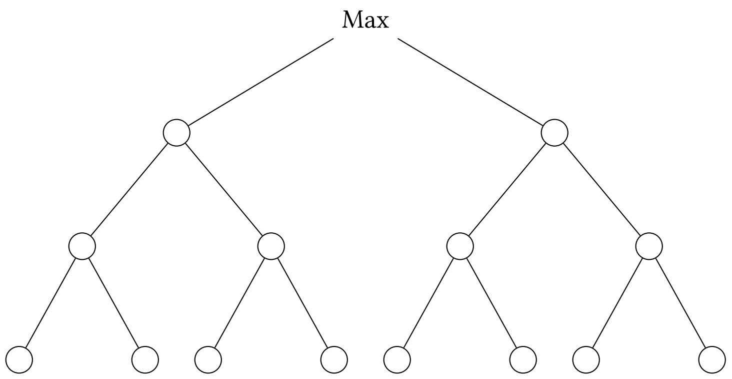
Answer 18



Answer 19



Answer 20



Answer 21 Yes

Chapter 11

Graphs

11.1 Multiple Choice

1. If G is an directed graph with 20 vertices, how many boolean values will be needed to represent G using an adjacency matrix?

a) 10

b) 20

c) 40

d) another number

2. In a graph, an ______ connects two ______ .

3. A directed graph is one in which

a) you must follow the minimum spanning tree.

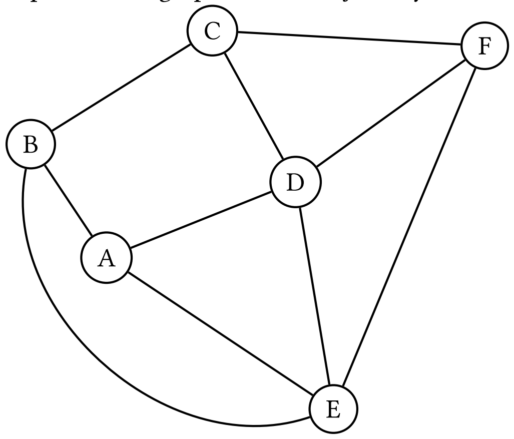
b) you must go from vertex A to vertex B to vertex C and so on so that values are in ascending order.

c) you can go in only one direction from one given vertex to another.

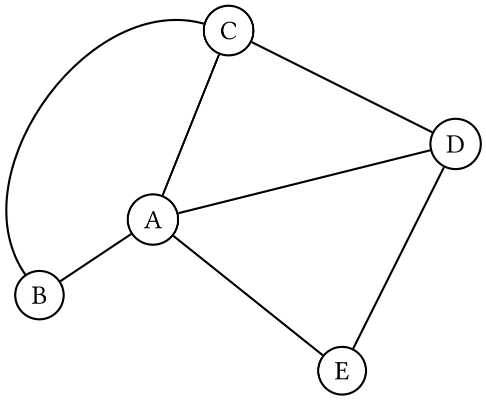
d) you can go in only one direction on any given path.

11.2 Undirected Graphs

4. Represent the graph with an adjacency matrix.

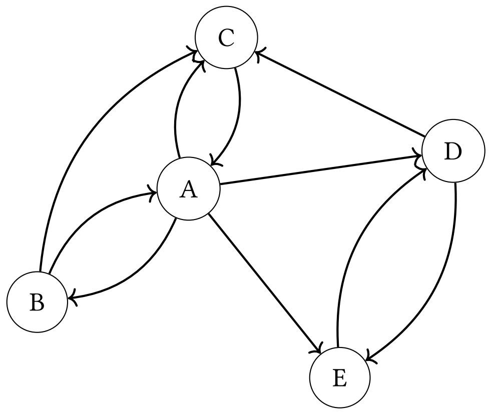


5. Represent the undirected graph with an adjacency list.



11.3 Directed Graphs

6. Represent the directed graph with an adjacency list.



11.4 Draw the Graphs

7. Draw the undirected graph for this adjacency matrix:

		To			
		A	B	C	D
From	A	0	0	0	0
	B	0	0	1	0
	C	0	1	0	1
	D	0	1	0	0

8. Convert the adjacency matrix to an undirected graph

		To							
		A	B	C	D	E	F	G	H
From	A	0	1	1	0	1	0	0	0
	B	1	0	1	1	0	0	0	0
	C	1	1	0	0	0	0	0	0
	D	0	1	0	0	1	0	0	0
	E	0	0	0	1	0	1	0	0
	F	0	0	0	1	1	0	0	0
	G	0	0	0	0	0	0	0	1
	H	0	0	0	0	0	0	1	0

Solutions to the Exercises

Answer 1 20

Answer 2 edge; vertices (or nodes)

Answer 3 d) you can go in only one direction on any given path.

Answer 4
6 vertices, 9 edges

		To					
		A	B	C	D	E	F
From	A	0	1	0	1	1	0
	B	1	0	1	0	1	0
	C	0	1	0	1	0	1
	D	1	0	1	0	1	1
	E	1	1	0	1	0	1
	F	0	0	1	1	1	0

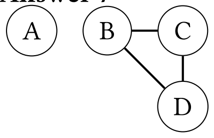
Answer 5 This graph has 5 vertices, 7 edges

		To				
		A	B	C	D	E
From	A	0	1	1	1	1
	B	1	0	1	0	0
	C	1	1	0	1	0
	D	1	0	1	0	1
	E	1	0	1	1	0

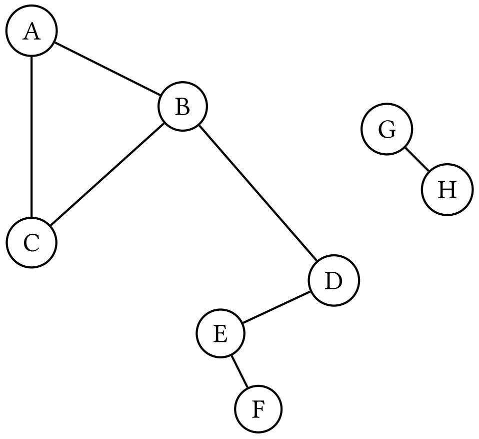
Answer 6 This graph has 5 vertices, 10 edges

		To				
		A	B	C	D	E
From	A	0	1	1	1	1
	B	1	0	1	0	0
	C	1	0	0	0	0
	D	0	0	0	0	1
	E	0	0	0	1	0

Answer 7



Answer 8


A
B
C
D
E
F
G
H

Chapter 12

Associative Arrays

1. Is searching a hashtable faster than searching a binary tree?

2. Insert the following three key/value pairs into a hashmap. What is the resulting data structure?

```
("potato", 0.65)
("celery", 0.30)
("carrot", 1.99)
```

3. Insert the following three key/value pairs into a hashmap. This time 2 keys are the same. What is the resulting data structure?

```
("apple", 1.27)
("pear", 1.50)
("kiwi", 1.99)
("kiwi", 2.50)
```

4. Using the table pictured below to complete this query. What value would be retrieved with get("Papaya")?

Key	**Value**
Coconut	1.99
Banana	0.49
Papaya	2.49
Mango	0.89

Solutions to the Exercises

Answer 1 Yes

Answer 2

Key	**Value**
`potato`	`0.65`
`celery`	`0.30`
`carrot`	`1.99`

Answer 3

Key	**Value**
apple	1.27
pear	1.99
kiwi	2.50

Answer 4 2.49

Chapter 13

Selecting Data Structures

Given the scenario described in the questions of the following pages, choose from the available data structures that should be used for implementation. It may be helpful to sketch out the data flow to understand the information that needs to be stored, and how.

a) Unsorted Array - a fixed sized container of elements

b) Sorted Array -a fixed sized container of sorted elements

c) Stack - last in, first out order

d) Queue - first in, first out order

e) Linked List - dynamically sized, linear container

f) HashMap - key value pair storage

g) Binary Max Heap - a binary tree where the root holds the maximum value

h) Binary Search Tree - starting from root, each node can have at most two children. Left child is smaller than parent, and right child larger.

i) Adjacency Matrix - a display of row and columns denoting graph node connections

j) Adjacency List - an array of nodes, each with a linked list denoting connected nodes

1. *Warehouse Items*

A warehouse manages their inventory in a database that is essentially a text file. Every week new items (with new product keys) are inserted and deleted from their inventory, and thus updates the database. These operations happen overnight while closed for 10 consecutive hours.

Quantities of items are change frequently: incremented as they are stocked, and decremented as they are sold. Stocking and selling items requires the item to be retrieved from the system using its product key.

It is also important that the system be robust, well-tested, and have predictable behaviour. Delays in retrieving an item are not acceptable, since it could cause problems for the sales staff. The system will potentially be used for a long time, though largely it is only the front end that is likely to be modified.

2. *Music Artists*

A record label executive received text files that contain the top streamed music artists during certain weeks. Each file represents one song by an artist. An artist's name might appear multiple times. In order for their in- house IT to be able to process the information, they need someone to help process the raw data by converting it to a list of artists with how many times they appear.

3. *Song List*

Your VIP client wants to listen to the music tracks in this week's latest song list to review them and get a sense for what they sound like. This time your client wants you to process all the full weeks of this quarter that have already passed so you're given multiple CSV files of the same format.

Your client also says they want to listen to songs. What data structure would you use to build a playlist? How about if the list were sorted in song/track titles in ascending order?

4. *Song Lengths*

Read in an input file of 100 lines one line at a time. Each line represents the length of a song.

At any point after reading the first 25 lines, if some line is blank (i.e., a string of length 0), then output the line that occurred 25 lines prior to that one. This program should be implemented so that it never stores more than 25 lines of the input at any given time.

5. *Guest List*

Read in a text input of a well known awards' show guest list, one line at a time. Each line represents a person's name.

When the end of the input is reached, write the lines to output in alphabetical order.

6. *Streaming Plays*

Read in a list of songs one line at a time. Each line contains the song title and the number of streaming plays that week for the song. Beginning with the 11th line, output the longest song length for the song read in so far (that hasn't previously been written) and remove that from the data structure.

Longest line would mean the number of characters to a name. This program should be implemented so that it never stores more than 10 lines of the input at any given time.

7. *Music Tour*

Read in the input that stores a locations a music artist travels on a tour. Each line represents a path between a start and an end point.

At the end of the reading in the input, the artist's manager want to quickly know how many miles has been traveled.

8. *Radio Plays*

Read in an input from a list of most played radio songs across the whole country, one line at a time and write each line to the output if it is not a duplicate of some previous input line.

Note that a file with a lot of duplicate lines should not require more memory than what is required for the number of unique lines.

Solutions to the Exercises

Answer 1 *Warehouse Items* F. HashMap

Answer 2 *Music Artists* F. HashMap

Answer 3 *Song List* B. Sorted array

Answer 4 *Song Lengths* D. Queue

Answer 5 *Guest List* B. Sorted array

Answer 6 *Streaming Plays* G. Binary Max Heap

Answer 7 *Music Tour* I. Adjacency Matrix

Answer 8 *Radio Plays* F. HashMap

Part III

Problem Sets on Sorting Algorithms

Chapter 14

Bubblesort

1. What is the bubble sort algorithm and how does it work? Write the pseudo code.

2. Given the input array below, sort using bubble sort algorithm.

7	2	3	8	9	1

3. Given the input array below, sort using bubble sort.

8	4	9	7	2	5

4. Given the input array below, sort using bubble sort.

8	4	9	7	2	5	10	1	3	6

5. Sort the following array with bubble sort: $A = [B, R, O, O, K, L, Y, N, C, O, L, L, E, G, E]$

Solutions to the Exercises

Answer 1 The logic for bubble sort algorithm:

- Start at position $i = 0$
- Compare two elements at position i and $i + 1$
- If $A[i] > A[i + 1]$, swap elements
- Move pointer one position right.

or in code,

```
BubbleSort( int a[])
    int out, in
    for out = a.length-1 to 1
    //outer loop backward
        for in = 0 to out
        //inner loop forward
            if a[in] > a[in+1]
                temp = a[in]
                a[in] = a[in+1]
                a[in+1] = temp
        end for
    end for

```

This will take an upper bound of $O(n^2)$ steps.

Answer 2 Bubble Sort on $A = 7, 2, 3, 8, 9, 1$ answer:

7	2	3	8	9	1
2	7	3	8	9	1
2	3	7	8	9	1
2	3	7	8	1	9
2	3	7	1	8	9

Answer 3 Bubble Sort on $A = 8, 4, 9, 7, 2, 5$:

8	4	9	7	2	5
4	8	9	7	2	5
4	8	7	9	2	5
4	8	7	2	9	5
4	8	7	2	5	9
4	7	8	2	5	9
4	7	2	8	5	9
4	7	2	5	8	9
4	7	2	5	8	9
4	2	7	5	8	9
4	2	5	7	8	9
2	4	5	7	8	9

Answer 4 Bubble Sort on $A = 8, 4, 9, 7, 2, 5, 10, 1, 3, 6$:

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

Answer 5 Bubble Sort on $A = [B, R, O, O, K, L, Y, N, C, O, L, L, E, G, E]$:

```
B R O O K L Y N C O L L E G E
B O R O K L Y N C O L L E G E
B O O R K L Y N C O L L E G E
B O O K R L Y N C O L L E G E
B O O K L R Y N C O L L E G E
B O O K L R N Y C O L L E G E
B O O K L R N C Y O L L E G E
B O O K L R N C O Y L L E G E
B O O K L R N C O L Y L E G E
B O O K L R N C O L L Y E G E
B O O K L R N C O L L E Y G E
B O O K L R N C O L L E G Y E
B O O K L R N C O L L E G E Y
B O K O L R N C O L L E G E Y
B O K L O R N C O L L E G E Y
B O K L O N R C O L L E G E Y
B O K L O N C R O L L E G E Y
B O K L O N C O R L L E G E Y
B O K L O N C O L R L E G E Y
B O K L O N C O L L R E G E Y
B O K L O N C O L L E R G E Y
B O K L O N C O L L E G R E Y
B O K L O N C O L L E G E R Y
B K O L O N C O L L E G E R Y
B K L O O N C O L L E G E R Y
B K L O O C N O L L E G E R Y
B K L O O C N L O L E G E R Y
B K L O O C N L L O E G E R Y
B K L O O C N L L E O G E R Y
B K L O O C N L L E G O E R Y
B K L O O C N L L E G E O R Y
B K L O C O N L L E G E O R Y
B K L O C N O L L E G E O R Y
...
B C E E G K L L L N O O O R Y
```

Chapter 15

Selection Sort

1. What is the selection sort algorithm and how does it work? Write the pseudo code.

2. Sort the given array using the selection sort algorithm. Write out the elements at each step.

5	7	2	8	9	1

3. Given the input array below, sort using selection sort.

8	4	9	7	2	5

4. Given the input array below, sort using selection sort.

8	4	9	7	2	5	10	1	3	6

5. Sort the following array with selection sort: $A = [B, R, O, O, K, L, Y, N, C, O, L, L, E, G, E]$

Solutions to the Exercises

Answer 1 Selection sort strategy selects the smallest value at each pass:

- Find the smallest value in tha array
- Move smallest value into the first slot.
- Repeat for other elements.

```
temp = 0;
for i->0 to A.length:
  temp = A[i]
  for j->0 to A.length:
    if A[j] < A[i]
      temp = A[j]
      A[i] = temp

```

Answer 2
Color code: Red: Element to sort (current index), Purple: Swapped element with smallest value found, Blue: Sorted into place, Sorted element in the array

Selection sort picks the smallest value at each step and sorts it into the right place. This is what the array would look like at each step.

5	7	2	8	9	$\boxed{1}$
1	7	2	8	9	5
1	7	$\boxed{2}$	8	9	5
1	2	7	8	9	$\boxed{5}$
1	2	5	8	9	7
1	2	5	8	9	$\boxed{7}$
1	2	5	7	9	8
1	2	5	7	9	$\boxed{8}$
1	2	5	7	8	9
1	2	5	7	8	9
1	2	5	7	8	9

Answer 3
Selection sort on $A = 8, 4, 9, 7, 2, 5$:

8	4	9	7	2	5
2	4	9	7	8	5
2	4	5	7	8	9

Answer 4
Selection sort on $A = 8, 4, 9, 7, 2, 5, 10, 1, 3, 6$:

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

Answer 5
Selection sort on $A = [B, R, O, O, K, L, Y, N, C, O, L, L, E, G, E]$:

```
B  R  O  O  K  L  Y  N  C  O  L  L  E  G  E
B  C  O  O  K  L  Y  N  R  O  L  L  E  G  E
B  C  E  O  K  L  Y  N  R  O  L  L  O  G  E
B  C  E  E  K  L  Y  N  R  O  L  L  O  G  O
B  C  E  E  G  L  Y  N  R  O  L  L  O  K  O
B  C  E  E  G  K  Y  N  R  O  L  L  O  L  O
B  C  E  E  G  K  L  N  R  O  Y  L  O  L  O
B  C  E  E  G  K  L  L  R  O  Y  N  O  L  O
B  C  E  E  G  K  L  L  L  O  Y  N  O  R  O
B  C  E  E  G  K  L  L  L  N  Y  O  O  R  O
B  C  E  E  G  K  L  L  L  N  O  Y  O  R  O
B  C  E  E  G  K  L  L  L  N  O  O  Y  R  O
B  C  E  E  G  K  L  L  L  N  O  O  O  R  Y
```

Chapter 16

Insertion Sort

1. What is the insertion sort algorithm and how does it work? Write the pseudo code.

2. Given the input array, sort using insertion sort algorithm.

7	2	4	6	3	1

3. Given the input array below, sort using insertion sort.

8	4	9	7	2	5

4. Given the input array below, sort using insertion sort.

8	4	9	7	2	5	10	1	3	6

5. Sort the following array with insertion sort: $A = [B, R, O, O, K, L, Y, N, C, O, L, L, E, G, E]$

Solutions to the Exercises

Answer 1 Think of insertion sort as sorting a hand of cards.

- Marked item as first unsorted
- Move this element to temporary variable
- Shift sorted elements until space opens for temporary variable
- Continue to place the next unsorted item

```
InsertionSort(int a[])
    int out, in
    for out = 1 to a.length
        temp = a[out]
        in = out
        while in>0 and a[in-1]>=temp
            a[in] = a[in - 1]
            in--
        end while
        a[in] = temp
    end for

```

Answer 2 See below for the answer:

7	2	4	6	3	1
2	7	4	6	3	1
2	4	7	6	3	1
2	4	6	7	3	1
2	3	4	6	7	1
1	2	3	4	6	7

Answer 3 Insertion Sort on $A = 8, 4, 9, 7, 2, 5$:

8	4	9	7	2	2	5
4	8	9	7	2	2	5
4	7	8	9	2	2	5
2	4	7	8	9	2	5
2	2	4	7	8	9	5
2	2	4	5	7	8	9

Answer 4 Insertion Sort on $A = 8, 4, 9, 7, 2, 5, 10, 1, 3, 6$:

8	4	9	7	2	5	10	1	3	6
4	8	9	7	2	5	10	1	3	6
4	7	8	9	2	5	10	1	3	6
2	4	7	8	9	5	10	1	3	6
2	4	5	7	8	9	10	1	3	6
sort1	2	4	5	7	8	9	10	3	6
1	2	3	4	5	7	8	9	10	6
1	2	3	4	5	7	8	9	10	6
1	2	3	4	5	6	7	8	9	10

Answer 5 Insertion Sort on $A = [B, R, O, O, K, L, Y, N, C, O, L, L, E, G, E]$:

B	R	O	O	K	L	Y	N	C	O	L	L	E	G	E
B	R	O	O	K	L	Y	N	C	O	L	L	E	G	E
B	R	O	O	K	L	Y	N	C	O	L	L	E	G	E
B	O	R	O	K	L	Y	N	C	O	L	L	E	G	E
B	O	O	R	K	L	Y	N	C	O	L	L	E	G	E
B	K	O	O	R	L	Y	N	C	O	L	L	E	G	E
B	K	L	O	O	R	Y	N	C	O	L	L	E	G	E
B	K	L	O	O	R	Y	N	C	O	L	L	E	G	E
B	K	L	N	O	O	R	Y	C	O	L	L	E	G	E
B	C	K	L	N	O	O	R	Y	O	L	L	E	G	E
B	C	K	L	N	O	O	O	R	Y	L	L	E	G	E
B	C	K	L	L	N	O	O	O	R	Y	L	E	G	E
B	C	K	L	L	L	N	O	O	O	R	Y	E	G	E
B	C	E	K	L	L	L	N	O	O	O	R	Y	G	E
B	C	E	E	K	L	L	L	N	O	O	O	R	Y	G
B	C	E	E	G	K	L	L	L	N	O	O	O	R	Y

Chapter 17

Mergesort

17.1 Pseudocode

1. What is mergesort, how does it work? Write pseudocode to describe the steps.

2. Fill in the blanks of the merge function. The mergesort function is provided.

```
# Sort an array of size n
mergesort(A, n):
    mergesort(_____________)
    mergesort(_____________)
    merge(first half of A, second half of A)
```

```
merge (A):
  while A not empty or B not empty:
    # Fill in this comment:
    if first element of A < first element of B:
      remove first element from A
      insert element into C
    end if

    # Fill in this comment:
    else:
      remove first element from B
      insert element into C
  end # while
```

17.2 Sorting Practice

3. Use merge sort on the array $A = 3, 1, 7, 7, 4$

Solutions to the Exercises

Answer 1 A divide and conquer approach to sorting.

1. Divide to find midpoint q. Split input array to $[0..q]$ and $[q+1..n]$
2. Conquer by recursively sorting subarrays from first step.
3. Combine by merging the two sorted sub arrays.

Answer 2

```
mergesort(A, n):   # Sort an array of size n
   mergesort(first half of A, n/2)
   mergesort(second half of A, n/2)
   merge(first half of A, second half of A)

merge (A):
while A not empty or B not empty:
   if first element of A < first element of B:
      remove first element from A
      insert element into C
   end if
   else:
      remove first element from B
      insert element into C
      end while

```

Answer 3 Split the array into two. Merge the first half, recursively. Merge the 2nd half, recursively.

$B = \{3, 1\}$ and $C = \{7, 7, 4\}$

Left:
$B_1 = \{3\}$ and $B_2 = \{3\}$
$B' = \{1, 3\}$

Right:
$C_1 = \{7\}$ and $C_2 = \{7, 4\}$
$C_1 = \{7\}$ and $C_{21} = \{7\}$ and $C_{22} = \{4\}$
$C_1 = \{7\}$ and $C_{2'} = \{4, 7\}$
$C' = \{4, 7, 7\}$

Merging B and C
$A' = \{1, 3, 4, 7, 7\}$

Chapter 18
Quicksort

18.1 True or false?

1. True or false? In quicksort, the pivot can be an arbitrary element of the array.

18.2 Pseudocode

2. What is quicksort, how does it work? Write pseudocode to describe the steps.

3. Fill in the blanks.

```
QuickSort(int a[])
    if start index < end index
        partition list around a pivot
        QuickSort(________)
        QuickSort(________)
```

18.3 Sorting Practice

4. Partition this array $A = \{25, 61, 14, 1, 8, 33, 7, 3, 20\}$ using pivot value 15

5. Sort the following array $A = \{5, 3, 9, 8, 7, 2, 4, 1, 6, 5\}$ using quick sort.

6. Sort the following array $A = \{9, 8, 5, 11, 12, 1, 15, 2, 10, 6\}$ using quick sort,

Solutions to the Exercises

Answer 1 True.

Answer 2 A divide and conquer approach to sorting.

1. Choose the pivot, store in the rightmost element
2. Partition the array into left (smaller keys) and right (larger keys) groups
3. Call self to sort the left group
4. Call self to sort the right group
5. Merge groups

Answer 3 Answer: See section on quicksort in this document, and also:

```
QuickSort(int a[])
  if start index < end index
      partition list around a pivot
      QuickSort (aLeft[])
      QuickSort (aRight[])

```

Left partition should come before right partition.

Input types should match the function input parameter

Answer 4 Left subarray elements should be smaller than or equal to pivot value.
Left subarray:
$A_L = \{14, 1, 8, 7, 3\}$

Right subarray:
$A_L = \{25, 61, 33, 20\}$

Answer 5 First split the array with the pivot value as the right most.
Left subarray:
$A_L = \{5, 3, 2, 4, 1, 5\}$
$A_L = \{5, 3, 2, 4, 1, 5\}$

Right subarray:
$A_R = \{9, 8, 7\}$

Answer 6 First split the array with the pivot value as the right most.
Left subarray:
$A_L = \{5, 1, 2\}$ and sorted = 6
$A_{LL} = \{1\}$; $A_{LS} = \{2\}$; $A_{LR} = \{5\}$ using 2 as next pivot value
$Left = \{1, 2, 5, 6\}$

Right subarray:
$A_R = \{12, 8, 15, 9, 10, 11\}$
$A_R = \{12, 8, 15, 9, 10, 11\}$ using 11 as next pivot value
$A_{RL} = \{7, 8, 10\}$; $A_{RS} = \{11\}$; $A_{RR} = \{12, 15\}$ $Right = \{7, 8, 10, 11, 12, 15\}$

Chapter 19

Heapsort

19.1 True or False?

1. True or false? The first node in a heap is the root.

2. True or false? The last node in a heap is the minimum value.

3. True or false? Heapsort is faster than shellsort

4. True or false? Heapsort is a stable algorithm.

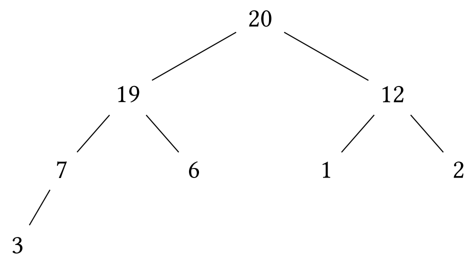
19.2 Data Representation

5. The root node of a Max Heap contains the _________ value.

6. What is the typical running time of a heapsort algorithm?

a) $O(n)$
b) $O(nlogn)$
c) $O(logn)$
d) $O(n^2)$

7. Represent the following heap as an array



19.3 Pseudocode

8. What are the steps to the heapsort algorithm?

19.4 Sorting Practice

9. Sort this array using heapsort. $A = [a, n, d, r, o, i, d]$

10. Sort this array $A = [i, p, h, o, n, e]$ using heapsort algorithm

11. Sort this array using heapsort $A = [n, e, w, y, o, r, k]$

Solutions to the Exercises

Answer 1 True

Answer 2 False

Answer 3 False

Answer 4 True. Heapsort uses fewer comparisons than other sorting algorithms and hence it is an extremely stable algorithm.

Answer 5 The root node of a Max Heap contains the maximum value.

Answer 6 b. $O(nlogn)$

Answer 7 20, 19, 12, 7, 6, 1, 2, 3

Answer 8 The heapsort algorithm is described in a few different ways:
1) Heapsort algorithm:

1. Build max heap from input array
2. Replace root with last item of heap. Reduce size of heap by 1. Largest value item becomes ‘sorted‘
3. Repeat above steps while size of heap is greater than 1

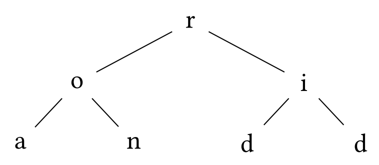
2) Pseudocode:

1. Start with a max heap (max element at the root)
2. Remove root and put into the last/vacant place
3. Reduce heap size by 1, heapify
4. Repeat

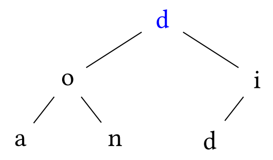
3) Pseudocode:

```
procedure heapsort(a, count) is
    input: an unordered array a of length count

    // (Build the heap in array a so that largest value is at the root)
    heapify(a, count)

    // (The following loop maintains the invariants that a[0:end]
    // is a heap and every element
    // beyond end is greater than everything before it (so a[end:count]
    // is in sorted order))
    end = count - 1
    while end > 0 do
        (a[0] is the root and largest value. The swap moves it in front of the sorted elements.)
        swap(a[end], a[0])-
        (the heap size is reduced by one)
        end = end - 1
        (the swap ruined the heap property, so restore it)
        siftDown(a, 0, end)
```

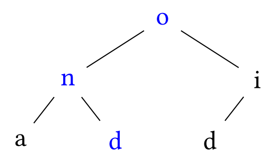
Answer 9 Steps
1. Build a max heap from the input array $A = [a, n, d, r, o, i, d]$



2. Replace root with last node, reduce heap by size of 1. Trickle new root down until you have a legal max heap.

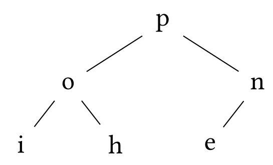


becomes

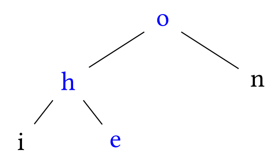


repeat steps 1 and 2 until you're left with $sorted = d, d, i, n, o, r$ and the max heap size is 1 (a). Largest values accumulate from the right.

Answer 10 Steps
1. Build a max heap from the input array $A = [i, p, h, o, n, e]$

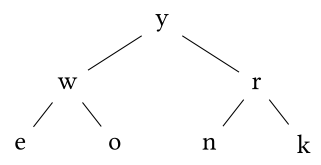


2. Remove root, new max heap becomes

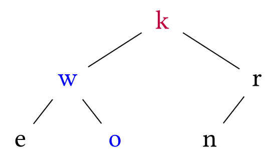


repeat steps 1 and 2 until you're left with $sorted = h, i, n, o, p$ and the max heap size is 1 (e).

Answer 11 Steps
1. Build a max heap from the input array "newyork"



2. Remove root, new max heap becomes



repeat steps 1 and 2 until you're left with $sorted = k, n, o, r, w, y$ and the max heap size is 1 (e).

Chapter 20

Shellsort

1. Which of the following sorting algorithms is closely related to shell sort?

a) John Von Neumann

b) Donald Shell

c) Tony Hoare

d) Alan Shell

2. Which of the following sorting algorithms is closely related to shell sort?

a) Selection sort

b) Merge Sort

c) Insertion sort

d) Bucket sort

3. The Shellsort works by:

a) partitioning the array

b) swapping adjacent elements.

c) dealing with widely separated elements

d) starting with the normal insertion sort

Solutions to the Exercises

Answer 1 b. Donald Shell

Answer 2 c. Insertion sort.

Answer 3 (c)

a) partitioning the array (Quicksort)

b) swapping adjacent elements. (bubble sort)

c) dealing with widely separated elements.

d) starting with the normal insertion sort. (statement doesn't make sense)

Part IV

Problem Sets on Searching Algorithms

Chapter 21

Searching Arrays

1. Describe the two search algorithm strategies, linear search and binary earch.

2. Write pseudocode that looks for value 33 from an array of values 0-100 using linear search.

3. Write pseudocode that searches the following array for value 7 using linear search. $A = \{1, 2, 5, 7, 9\}$

4. Write pseudocode that searches the following array for value 8. $B = \{3, 4, 6, 8, 10\}$

5. Write pseudocode that uses binary search to search through the array for value 7. $A = \{1, 3, 5, 7, 9, 11\}$

6. Given this array, how can we find the value "55" ? $A = \{15, 20, 25, 35, 40, 55, 85\}$

Solutions to the Exercises

Answer 1 Linear Search $O(n)$: Start from the left (index=0) and if the value is not found, keep going with increment of the position by 1.

```
for item in array
  if item == key
    return true
return false
```

Binary Search $O(logn)$: Before binary search we should ensure it's a sorted array. Take the array size, divide in half to find the midpoint value. Compare the value you're searching for with the midpoint value. If search value is smaller than midpoint, search the left half (smaller values) which is between index(0,midpoint). If search value is larger than midpoint, search the right half (larger values) which is between index(midpoint,size). Continue this strategy of finding a new midpoint value and searching the smaller array size until search value is found.

```
if no items
    Return false
if middle item == key
    Return true
else-if key < middle item
    Search left half
else-if key > middle item
    Search right half
```

Answer 2 Linear search is O(n) time and this would take 33 steps.

```
  key = 33
  for item in A:
    if key == A[i]
      return;
```

Answer 3

```
key = 7
for item in A:
  if key == A[i]
    return 'Found'
```

Answer 4

```
key = 8
for item in A:
  if key == A[i]
    return 'Found'
```

Answer 5

```
if no items
    Return false
if middle item == 7
    Return true
else-if 7 < middle item
    Search left half
else-if 7 > middle item
    Search right half
```

Answer 6

```
key = 55
if no items
    Return false
if middle item == key
    Return true
else-if key < middle item
    Search left half
else-if key > middle item
    Search right half
```

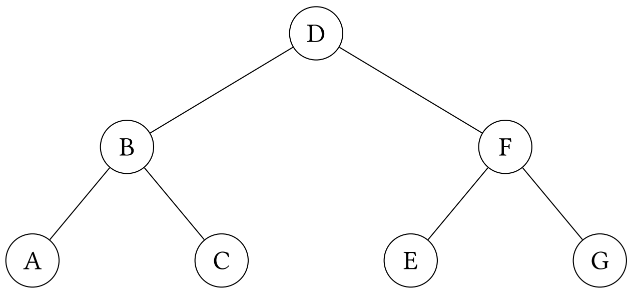
Chapter 22

Depth First Search (DFS) Tree Traversals

22.1 Inorder Traversal

1. Describe the steps for depth first search using inorder traversal algorithm.

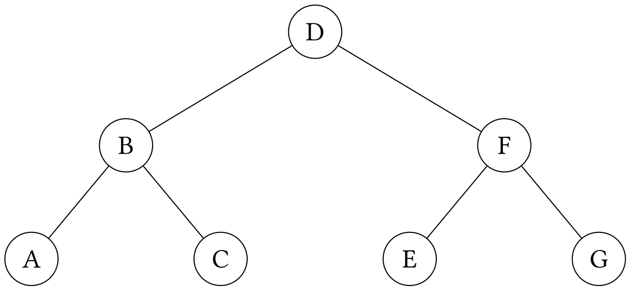
2. Using the tree pictured below, list out the nodes using inorder traversal.



22.2 Pre order Traversal

3. Describe the steps for depth first search using pre-order traversal algorithm.

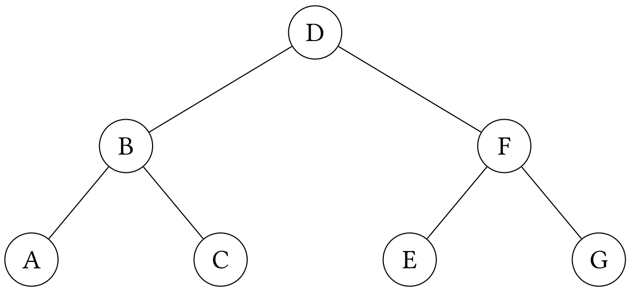
4. Using the tree pictured below, list out the nodes using pre-order traversal.



22.3 Post order Traversal

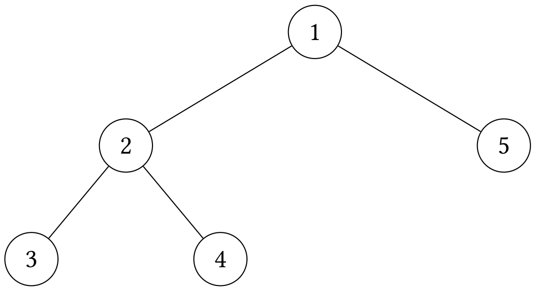
5. Describe the steps for depth first search using post-order traversal algorithm.

6. Using the tree pictured below, list out the nodes using post-order traversal.



22.4 All in

Using the tree pictured below, please list the final output from the following traversal algorithms.

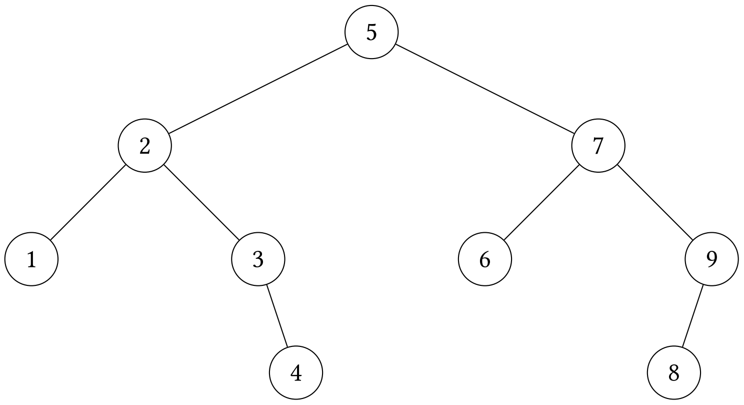


7. Inorder Traversal

8. Preorder Traversal

9. Postorder Traversal

Using the tree pictured below, please list the final output from the following traversal algorithms.



10. Inorder Traversal

11. Preorder Traversal

12. Postorder Traversal

Solutions to the Exercises

Answer 1

```
inorder(tree)
 begin
     if tree is null, return;
     inorder(tree.left_subtree);
     print(tree.root);
     inorder(tree.right_subtree);
 end
```

Answer 2 A, B, C, D, E, F, G

Answer 3

```
preorder(tree)
 begin
     if tree is null, return;

     print(tree.root);
     preorder(tree.left_subtree);
     preorder(tree.right_subtree);
 end
```

Answer 4 D,B,A,C,F,E,G

Answer 5

```
postorder(tree)
 begin
     if tree is null, return;

     postorder(tree.left_subtree);
     postorder(tree.right_subtree);
     print(tree.root);
 end
```

Answer 6 A,C,B,E,G,F,D

Answer 7 3,2,4,1,5

Answer 8 1,2,3,4,5

Answer 9 3,4,2,5,1

Answer 10 1,2,3,4,5,6,7,8,9

Answer 11 5,2,1,3,4,7,6,9,8

Answer 12 1,4,3,2,6,8,9,7,5

Appendices

Growth of Functions

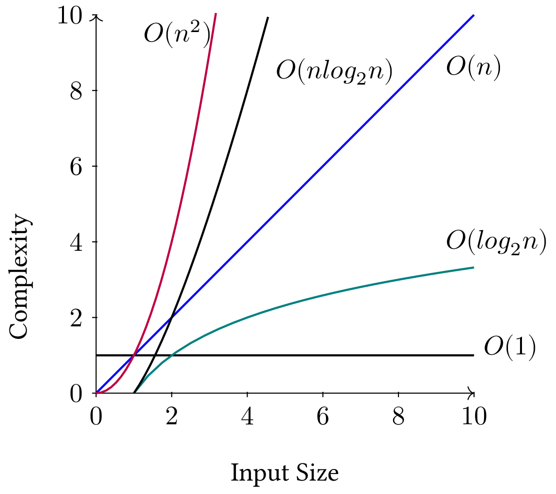
Introduction

A common technique used to compare the performance of data structure operations is to estimate the number of steps as a unit of time to complete an operation. In order to make an estimation we can look at the asymptotic behavior (a function or expression with definite limit) based on collection size.

Asymptotic notation is the notation used to present the estimated complexity. While many times folks refer to Big O notation with dominant coefficients, this can be misleading when your analysis needs to be more precise than only the worst case (upper bound). To be precise, you should be looking at use of big theta, big omega, or big O. Each of these represent slightly different measures for lower bound to upper bound range. Regardless, this is a form of notation that gives us the possibility to compare between different algorithms in terms of efficiency.

Plotting the orders of growth

There are several common order of growth functions that will frequently appear as you study algorithms that correspond to Big O notation. They are graphed on the chart below



When n for number of elements is small, the functions are similar, it's hard to tell which is more efficient. However, as input size n grows along the x axis, the trajectories branch apart. In the plot, the higher the y value means the more steps an operation requires. Steeper y means it grows faster, or is more expensive.

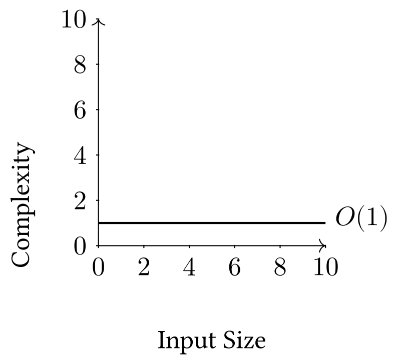
Orders of growth

Constant order of growth $O(1)$

This means that the algorithm is performed in constant time regardless of the amount of data. The different size of input doesn't affect the operation's performance.

This might be some operation such as popping an item off a stack. Or adding two numbers.

```
stack.pop();  // remove item from stack
a = b + c;    // assign expression to name
```



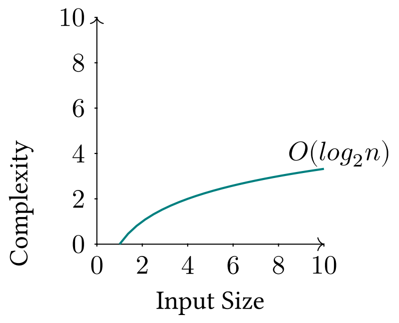
Logarithmic order of growth $O(logn)$

Logarithmic oder of growth follows the number series that grows exponentially such as $1, 2, 4, 8, 16, ..., 2^n$ except flipped. For each input number n using a series $1, 2, 3...n$ into a function $O(logn)$, the output equals $log_2 n$.

Commonly this is expressed as a for loop where the index starts from i and increased by order of multiples.

```
for (int i = 0; i < A.length - 1; i*=2){
  // do something
}
```

Binary search is another example where at each step, the amount of items to iterate on is half of before.



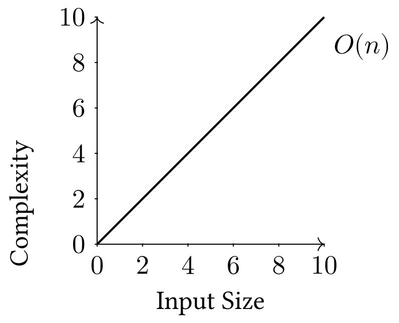
Linear order of growth $O(n)$

Linear is pretty much synonymous with the number series $1, 2, 3, 4, 5, ...$ For each input number n into $O(n)$, the output equals n.

Commonly this is expressed as a for loop where the index starts from i and increased by order of 1 increment.

Finding the maximum value in an array is an example.

```
for (int i = 0; i < A.length - 1; i++){
  // do something
}
```



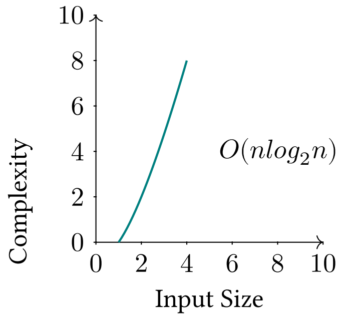
Linearithmic order of growth $O(nlogn)$

Linearithmic is a little more complicated. It is a combination of the number series $1, 2, 3, 4, 5, ...$ combined with the log of that input number.. For each input number n the output is $O(n * logn)$, the output equals $n * logn$.

Commonly this is expressed as a combination where one part of the algorithm divides and other part merges.

This might be used for merge sort for instance.

```
for(int i = 0; i < A.length - 1; i++){
  for(int j = 0; j < A.length - 1; j*=2)  {
    // do something
  }
}
```



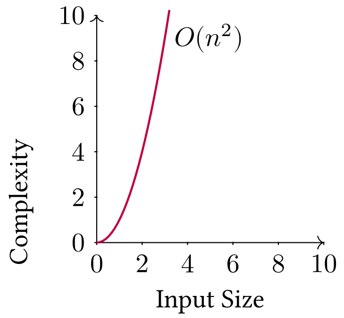
Quadratic order of growth $O(n^2)$

Quadratic is the sequence for when the input is sequred. In the number series $1, 2, 3, 4, 5, ...$ For each input number n is multiplied by itself to become squared $O(n^2)$, the output equals n^2.

Commonly this is expressed as a nested for loop where for both the index starts from i and increased by order of 1 increment.

This might be used for checking all pairs of an array.

```
for (int i = 0; i < A.length - 1; i++){
  for (int j = 0; j < i; j++){
    // do something
  }
}

```



Summary of Complexities

Data Structures: Advantages and Disadvantages

Data Structure	**Advantages**	**Disadvantages**
Arrays (Unsorted)	Quick insertion, very fast access if the index is known	Slow search, slow deletion, fixed size
Arrays (Sorted)	Slow Insertion	Slow search with linear search (faster with binary search)
Stack	Provides last-in, first-out access (LIFO)	Slow access to other items that's not first
Queue	Provides first-in, first-out access (FIFO)	Slow access to other items in the middle.
Linked List	Quick insertion, quick deletion	Slow search
Binary Search Tree	Quick search, insertion, deletion	Detection algorithm is complex
Max Heap	Easy access to maximum value	Slow access to other values.
Graph	Models real-world situations	Some algorithms are slow and complex
Associative arrays	Very fast access if key known. Fast insertion	Slow deletion, access slow if key not known. Inefficient memory usage

Data Structures: Operations

Basic Data Structures
These are the basic data structures for storing and retrieving data using key values, which works for general-purpose programs. They do not restrict access the way the logical data structures do in the next section.

Data Structure	**Insert**	**Delete**	**Search**	**Traversal**
Arrays (Unsorted)	$O(1)$	$O(n)$	$O(n)$	
Arrays (Sorted)	$O(n)$	$O(n)$	$O(logn)$	
Linked List	$O(1)$	$O(n)$	$O(n)$	
Binary Search Tree	$O(logn)$	$O(logn)$	$O(logn)$	$O(n)$
Binary Heap	$O(n)$	$O(n)$	$O(n)$	
HashMap	$O(1)$	$O(1)$	$O(1)$	$O(1)$

Special Purpose Data Structures
These are logical data structures used with algorithms that could be implemented by any of the above basic data structures.

Data Structure	**Insert**	**Delete**	**Access**
Stacks	$O(1)$	$O(1)$	$O(1)$ last item inserted
Queues	$O(1)$	$O(1)$	$O(1)$ first item inserted

Sorting Algorithms

	Algorithmic complexity			
Sort	**Worst case**	**Best case**	Inplace?	Stable?
Bubble	$O(n^2)$	$O(n)$	yes	yes
Insertion	$O(n^2)$	$O(n)$	yes	yes
Selection	$O(n^2)$	$O(n^2)$	yes	no
Mergesort	$O(nlogn)$	$O(nlogn)$	no	yes
Quicksort	$O(n^2)$	$O(nlogn)$	yes	no
Heapsort	$O(nlogn)$	$O(nlogn)$	yes	no

Backcover

Description

This *Data Structures Workbook* is suitable for students in introductory data structures courses. Most chapters that appear in this text do not depend on any particular programming language. The exercises are designed to help with reviewing main concepts for each data structure or algorithm.

The book is divided into several parts. **Part 1** contains sets of problems on a review of fundamental concepts. **Part 2** contains sets of problems using data structures and performing operations on data structures.. **Part 3** contains sets of problems on sorting algorithms. **Part 4** contains sets of problems on searching algorithms. **Part 5** is an appendix of helpful guides to quickly look up information.

Biography

The author *Dr. Katherine Chuang* is a Senior Doctoral Lecturer in the Department of Computer and Information Science at Brooklyn College of the City University of New York. She teaches several of the core undergraduate courses in the department with developing curricula alongside with including a practical perspective of technical skill assessment to the classroom.

Prior to joining she earned her doctorate specializing in developing analytical and empirical research techniques of studying social media design and spent many years of practice in the tech industry. Her primary focus these days contributes to the symbiotic relationship between academia and the tech industry.

Dr. Chuang was inspired to create this workbook after teaching the CISC 3130 Data Structures course at Brooklyn College. In the process of publishing course material online under the Open Educational Resource (OER) Initiative with the Brooklyn College librarians, it became clear that there was also additional opportunity for drill-based learning.