Spring 2023 CS 1332 StudyPalooza Worksheet Answer Key

Link to doc: https://bit.ly/40EQULs

Big O

Data Structures Table

We are assuming **worst-case time complexity with amortized analysis** (denoted with an asterisk). Feel free to copy this table and fill it out for average-case analysis.

	add			remove			search		resize
	First index	Last index	Any index	First index	Last index	Any index	At index	For Specific Value	
ArrayLists	O(n)	O(1)*	O(n)	O(n)	O(1)	O(n)	O(1)	O(n)	O(n)
SLL, no tail	O(1)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)	N/A
SLL, with tail	O(1)	O(1)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)	N/A
CSLL	O(1)	O(1)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)	N/A
DLL with tail	O(1)	O(1)	O(n)	O(1)	O(1)	O(n)	O(n)	O(n)	N/A
Stack (array-backed) Add/Remove from “Back”	N/A	O(1)*	N/A	N/A	O(1)	N/A	N/A	N/A	O(n)
Queue (array-backed) Add to “Back” and Remove from “Front”	N/A	O(1)*	N/A	O(1)	N/A	N/A	N/A	N/A	O(n)
Deque (array-backed)	O(1)*	O(1)*	N/A	O(1)	O(1)	N/A	N/A	N/A	O(n)

Data Structures Table (continued) - Assume **Worst Case Amortized** but feel free to make a table for average case

	add	search	remove
BST	O(n)	O(n)	O(n)
Heap	O(log(n))*	N/A	O(log(n))
HashMap with External Chaining	O(n)	O(n)	O(n)
HashMap with Linear/Quadratic Probing	$O(n^2)$	O(n)	O(n)
AVL	O(log(n))	O(log(n))	O(log(n))
SkipList	O(n)	O(n)	O(n)
2-4 Tree	O(log(n))	O(log(n))	O(log(n))

Sorting Algorithms Table

	Big-O Best Case	Big-O Worst Case	stable?	in-place?	adaptive?
Bubble Sort	O(n)	$O(n^2)$	Yes	Yes	Yes
Insertion Sort	O(n)	$O(n^2)$	Yes	Yes	Yes
Selection Sort	$O(n^2)$	$O(n^2)$	No	Yes	No
Cocktail Shaker Sort	O(n)	$O(n^2)$	Yes	Yes	Yes
Merge Sort	O(nlog(n))	O(nlog(n))	Yes	No	No
Quick Sort	O(nlog(n))	$O(n^2)$	No	Yes	No
LSD Radix Sort	O(kn)	O(kn)	Yes	Yes	No
HeapSort	O(nlog(n))	O(nlog(n))	No	No	No

Pattern Matching Table

	Best Case		Worst Case	
	Single occurrence	All occurrences	Single occurrence	All occurrences
Brute Force	O(m)	O(n-m)*	O(nm)	O(nm)
Boyer-Moore (no Galil Rule)	O(m)	O(n/m + m)	O(nm + m)	O(nm + m)
Boyer-Moore with Galil Rule (no good suffix rule)	O(m)	O(n/m + m)	O(nm + m)	O(nm + m)

KMP	O(m)	O(n + m)	O(n + m)	O(n + m)
Rabin Karp	O(m)	O(n + m)	O(nm + m)	O(nm + m)

*For brute force best case all occurrence, you can also say O(n)
For the “nm + m” entries, the +m can technically be dropped.

Graphs Table

Data Structure or Algorithm	Worst Case Time or Space Complexity
Adjacency Matrix (space complexity)	$O(\|V\|^2)$
Adjacency List (space complexity)	$O(\|V\|^2)$
Edge List (space complexity)	O(|E|)
Depth First Search	O(|V| + |E|)
Breadth-First Search	O(|V| + |E|)
Dijkstra’s Algorithm	O(|E|log|E|)
Prim’s Algorithm	O(|E|log|E|)
Kruskal’s Algorithm	O(|E|log|E|)

LCS Runtime: O(nm)

T/F:

1. T
2. T
3. T
4. F
5. F
6. F
7. F

MC:

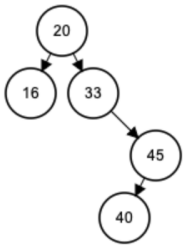
1. A, B
2. A
3. C
4. C
5. B, D, E
6. D
7. C
8. B
9. A

Diagramming:

1.

null	100	46	84	33	25	50	77	11

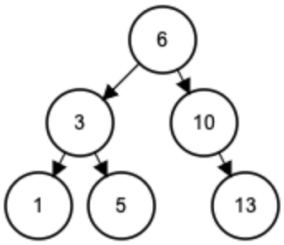
2.



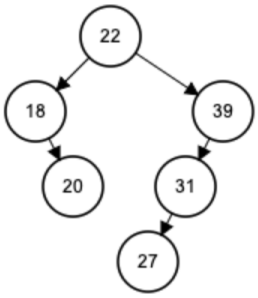
a.

b. Any BST with given data

c.

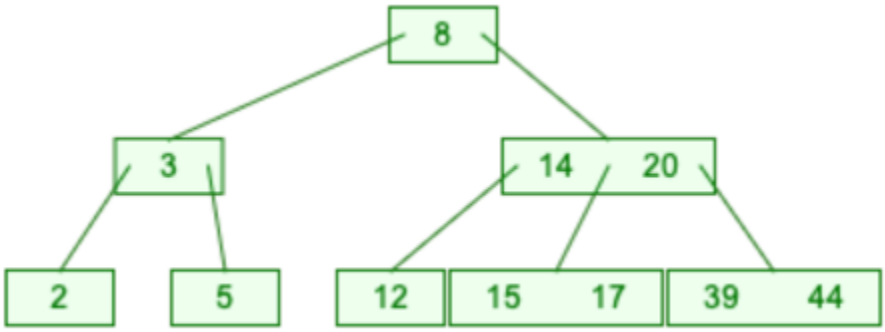


d.

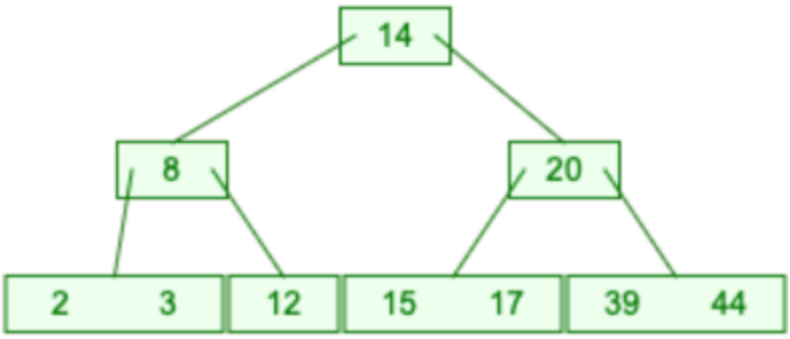


3.

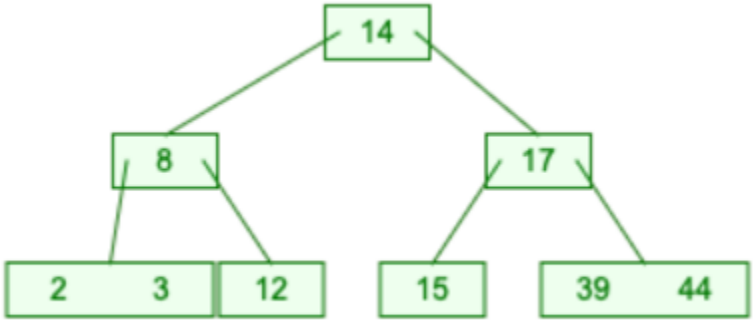
a.



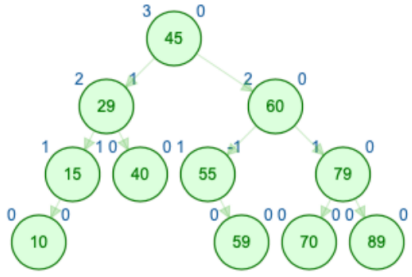
b.


14
8
20
2
3
12
15
17
39
44

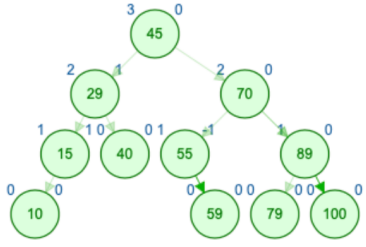
c.


14
8
17
2
3
12
15
39
44

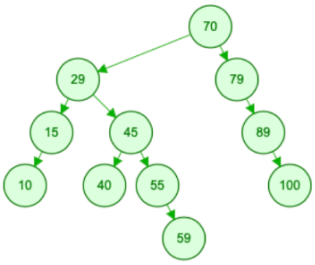
4.


45
29
60
15
40
55
79
10
59
70
89

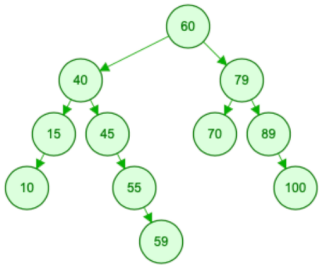
5.


45
29
70
15
40
55
89
10
59
79
100

6.



7.



8. DFS: B A E G H D F C
   BFS: B A C D F G E H

9.

Visited Set:
G B C A E F H D

Current vertex:

Priority Queue: (D, 14)

Distance Map:

A	7
B	6
C	6
D	11
E	7
F	8
G	0
H	9

10. BD, CF, CG, BC, AD, CE, DH
    The edges in above make up the MST

11.

			f	r	o	z	o	n	e
		∅	0	1	2	3	4	5	6
	∅	0	0	0	0	0	0	0	0
b	0	0	0	0	0	0	0	0	0
r	1	0	0	1	1	1	1	1	1
o	2	0	0	1	2	2	2	2	2
n	3	0	0	1	2	2	2	3	3
z	4	0	0	1	2	3	3	3	3
e	5	0	0	1	2	3	3	3	4

LCS: rone or roze

12.

			B	R	O	R	A	N	C	H
		∅	0	1	2	3	4	5	6	7
	∅	0	0	0	0	0	0	0	0	0
B	0	0	1	1	1	1	1	1	1	1
R	1	0	1	2	2	2	2	2	2	2
U	2	0	1	2	2	2	2	2	2	2
N	3	0	1	2	2	2	2	3	3	3
C	4	0	1	2	2	2	2	3	4	4
H	5	0	1	2	2	2	2	3	4	5
I	6	0	1	2	2	2	2	3	4	5
N	7	0	1	2	2	2	2	3	4	5
G	8	0	1	2	2	2	2	3	4	5

LCS: BRNCH