Problem Set 9, Part I

Problem 1: Comparing data structures

They want to be able to retrieve course records by specifying the course name, which consists of the departmental abbreviation followed by the course number.
searching and deleting
They want the time required to retrieve a record to relatively efficient – on the order of 15-20 operations per retrieval, given a database of approximately 20,000 records.
efficiency of deleting
They want to be able to efficiently retrieve all courses for a given department.
efficiency of multiple deleting a whole part
They want to be able to add the records for the upcoming semester – adding a large set of new records – without taking the system offline.
inserting a large set of data and its efficiency

Efficiency for **searching** in a **binary search tree** in average is O(log n), and the worst case is O(n).
Efficiency for **inserting** in a **binary search tree** in average is O(log n), and the worst case is O(n).
Efficiency for **deleting** in a **binary search tree** in average is O(log n), and the worst case is O(n).
Efficiency for **searching** in a **2-3 tree** in average is O(log n), and the worst case is O(log n).
Efficiency for **inserting** in a **2-3 tree** in average is O(log n), and the worst case is O(log n).
Efficiency for **deleting** in a **2-3 tree** in average is O(log n), and the worst case is O(log n).
Efficiency for **searching** in a **hash table** in average is O(1), and the worst case is O(n).
Efficiency for **inserting** in a **hash table** in average is O(1), and the worst case is O(n).
Efficiency for **deleting** in a **hash table** in average is O(1), and the worst case is O(n).

Since O(1) < O(log n), a hash table is the best choice in average.

Problem 2: Complete trees and arrays
2-1)
For node at index i in array, has left child present on index 2i + 1, right child on index 2i + 2, and parent present on (i + 1) / 2.

left child: 2(75) + 1 = 151
right child: 2(75) + 2 = 152
parent: (75 - 1) / 2 = 37

2-2)
For tree of height h, will have 2^h - 1 nodes in total.

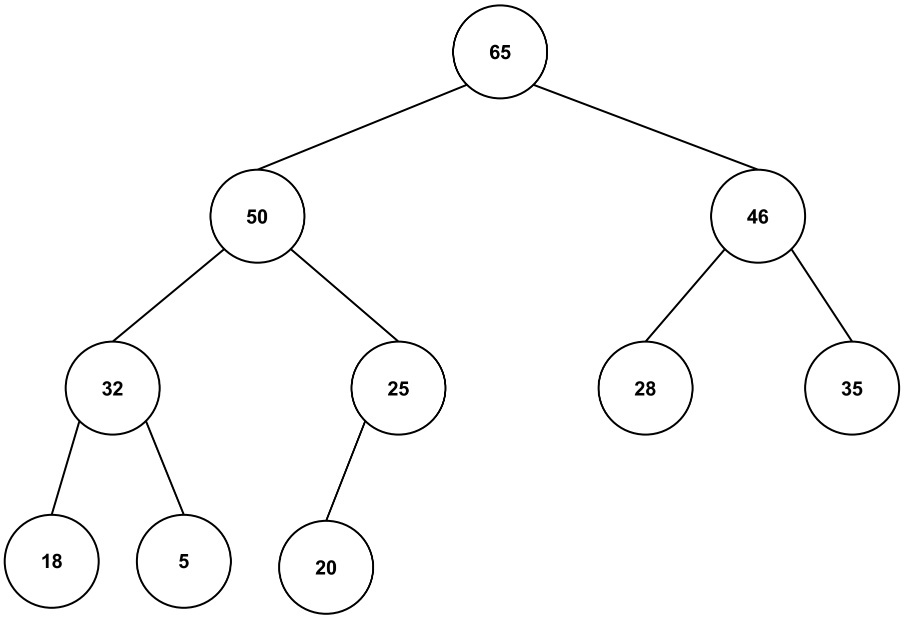
325 = 2^h – 1
2^h = 326
h = 8.35

so the heigh of the tree is 9 because round to next larger whole number

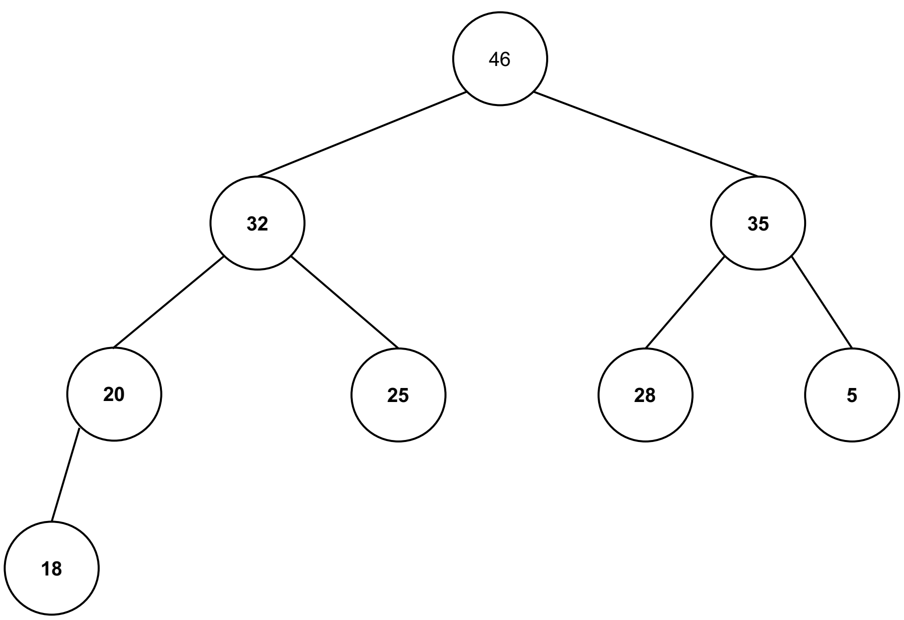
2-3)
The left node of index i is at 2*i + 1, and the right child is at 2*i + 2, so if the index number is odd, it is a left node, or the index number is eve, it is a right node.

As we calculated, the bottom level of the tree is at height 9
so there are 2^8 = 256 nodes above the bottom level
and the bottom level have 326 – 256 = 70 nodes
the last node is the 70th node which is even so it is right child

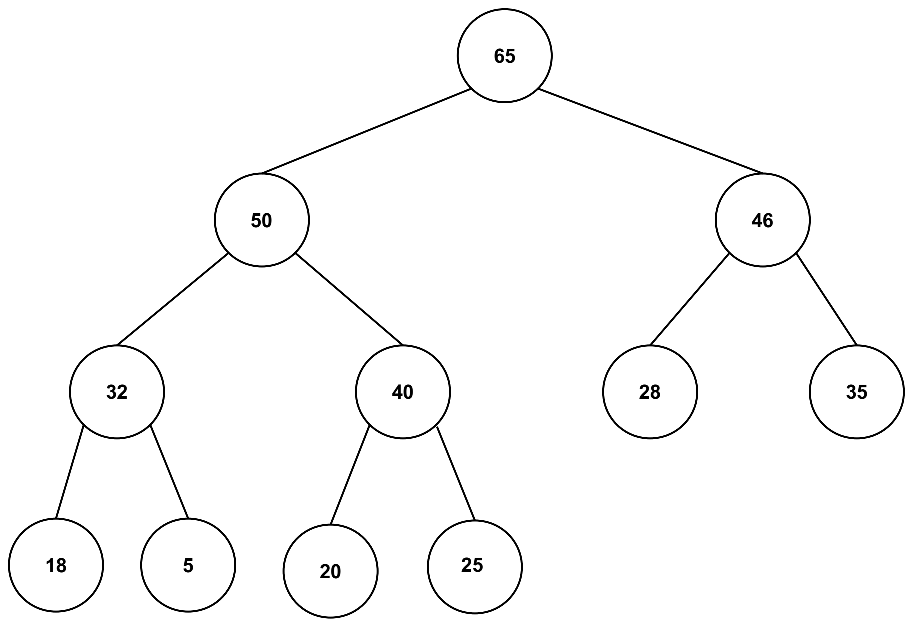
Problem 3: Heaps
3-1)
after one removal

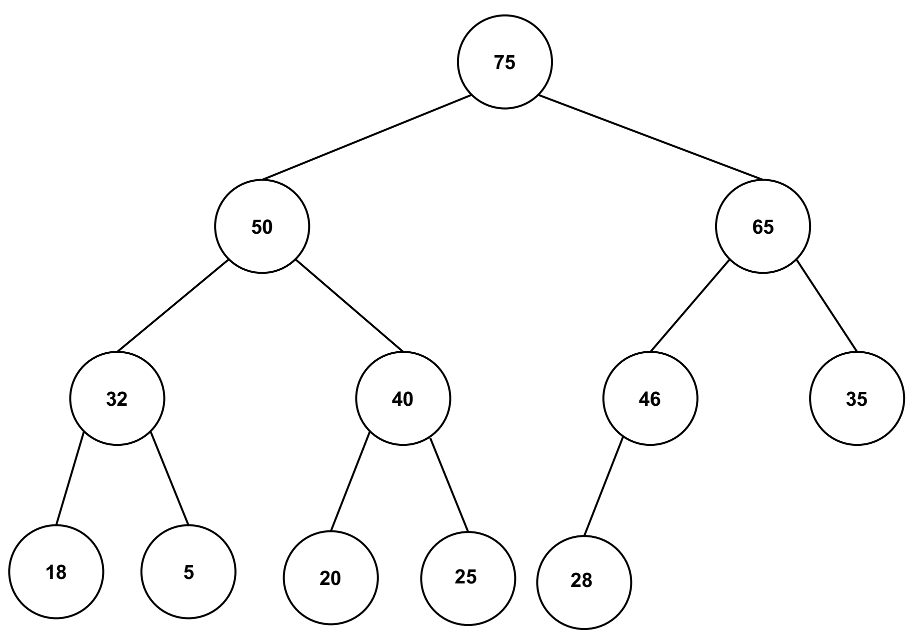


after a second removal
(copy your revised diagram from part 1 here, and edit it to show the result of the second removal)

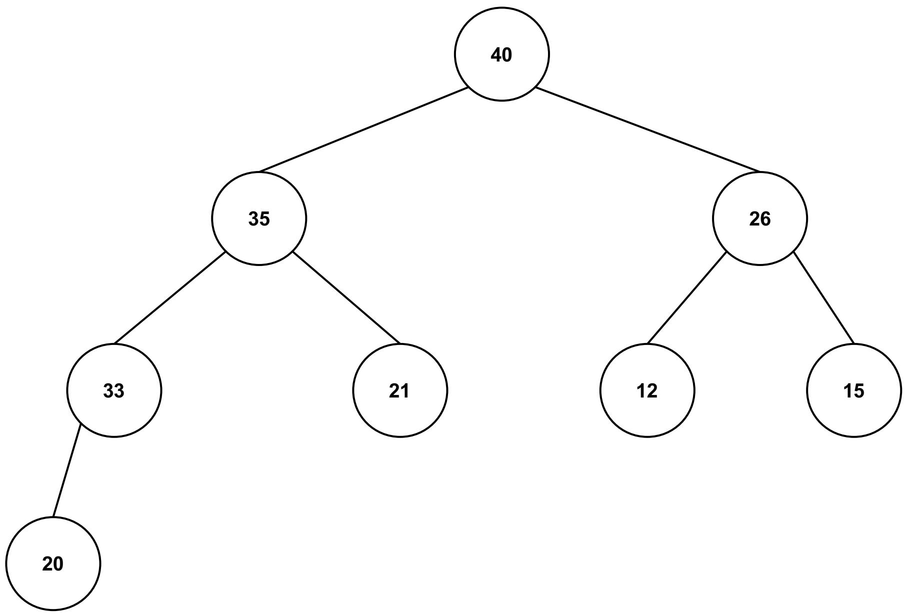


3-2)





Problem 4: Heaps and HeapSort



4-2)
40 35 36 33 21 12 15 20

4-3)
first time: 35 33 26 20 21 12 15 40
second time: 33 21 26 20 15 12 35 40

Problem 5: Hash tables

5-1) linear

0	if
1	to
2	my
3	the
4	an
5	by
6	do
7	we

go overflow

5-2) quadratic

0	
1	
2	my
3	the
4	
5	
6	an
7	

by overflow

5-3) double hashing

0	we
1	
2	my
3	the
4	do
5	an
6	by
7	

if overflow

5-4) probe sequence:
3, 5, 7

5-5) table after the insertion:

0	function
1	
2	
3	our
4	
5	table
6	
7	see