CS 202

Homework 4

Sec 03

Kutay Tire

22001787

13.05.2022

TA: Batuhan Kaynak

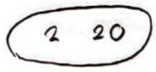
Instructor: Ertuğrul Kartal Tabak

Question 1.)

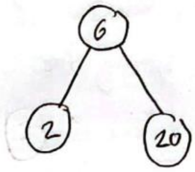
a.)

1- Insert 2

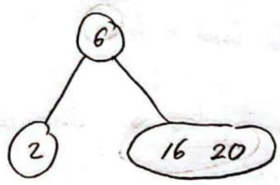
2- Insert 20



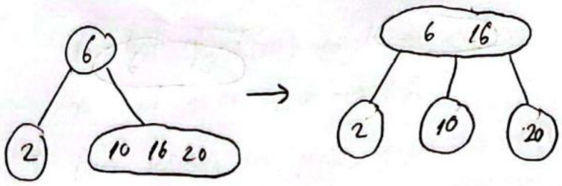
3- Insert 6



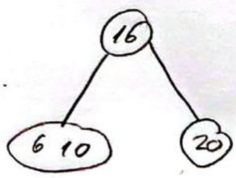
4- Insert 16



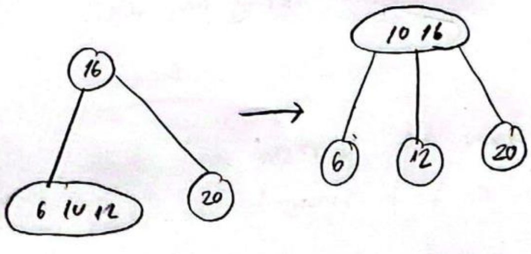
5- Insert 10



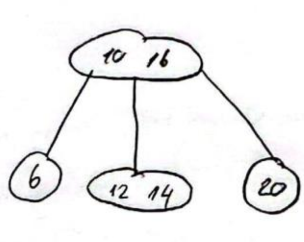
6- Delete 2



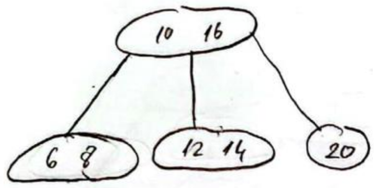
7- Insert 12



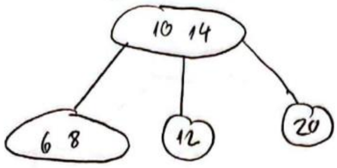
8- Insert 14



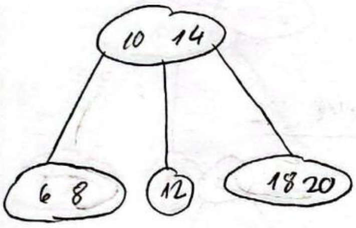
9 - Insert 8



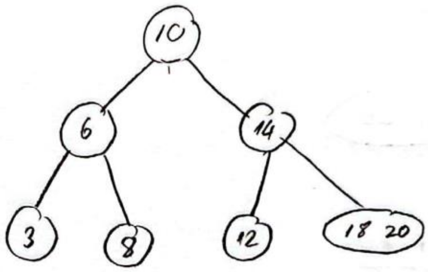
10 - Delete 16



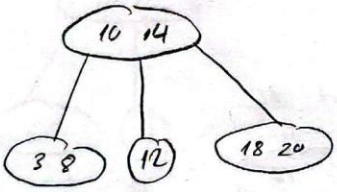
11 - Insert 18



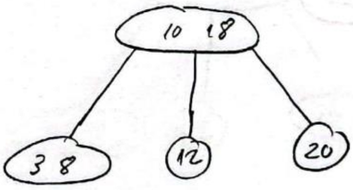
12 - Insert 3



13 - Delete 6



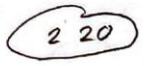
14 - Delete 14



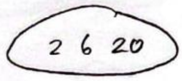
b.)

1- Insert 2

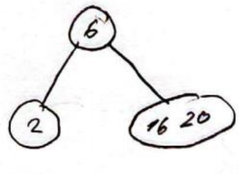
2- Insert 20



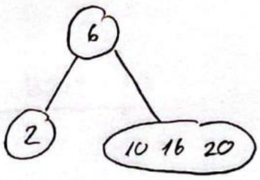
3 - Insert 6



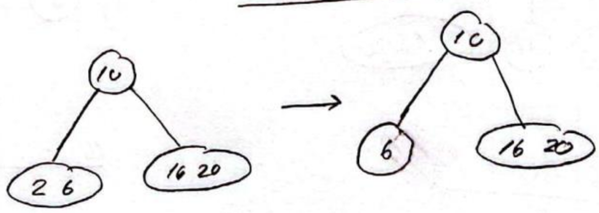
4- Insert 16



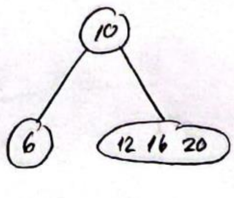
5 - Insert 10



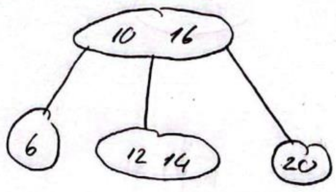
6- Delete 2



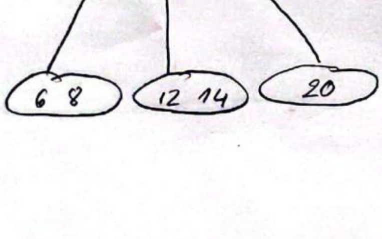
7- Insert 12



8- Insert 14

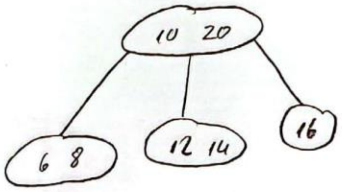


9- Insert 8

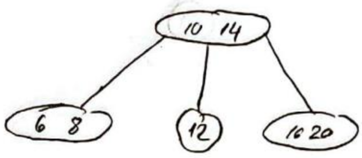


10 - Delete 16

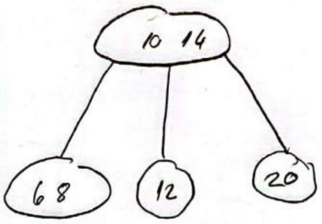
1. Find the successor and swap



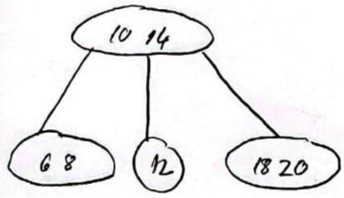
2. Transfer an item as 16 is 2-node



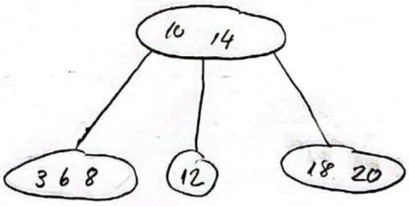
3. Delete 16



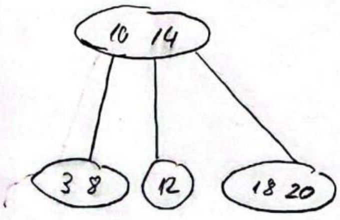
11 - Insert 18



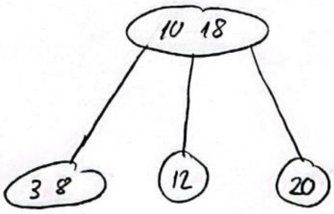
12 - Insert 3



13 - Delete 6



14 - Delete 14



Question 2

a.)

0	26
1	
2	54
3	
4	17
5	69
6	45
7	58
8	32
9	60
10	
11	
12	64

45 mod 13 = 6
64 mod 13 = 12
54 mod 13 = 2
17 mod 13 = 4
69 mod 13 = 4 + 1 = 5
58 mod 13 = 6 + 1 = 7
32 mod 13 = 6 + 1 + 1 = 8
60 mod 13 = 8 + 1
26 mod 13 = 0

Load Factor = 9/13

- Average number of probes for successful search:

$$\frac{1}{2}\left[1 + \frac{1}{1-\frac{9}{13}}\right] = 2.125$$

- Average number of probes for unsuccessful search:

$$\frac{1}{2}\left[1 + \frac{1}{\left(1-\frac{9}{13}\right)^2}\right] \approx 5.78125$$

b.)

0	26
1	
2	54
3	
4	17
5	69
6	45
7	58
8	60
9	
10	32
11	
12	64

45 mod 13 = 6
64 mod 13 = 12
54 mod 13 = 2
17 mod 13 = 4
69 mod 13 = $4 + 1^2 = 5$
58 mod 13 = $6 + 1^2 = 7$
32 mod 13 = $6 + 2^2 = 10$
60 mod 13 = 8
26 mod 13 = 0

Load Factor = 9/13

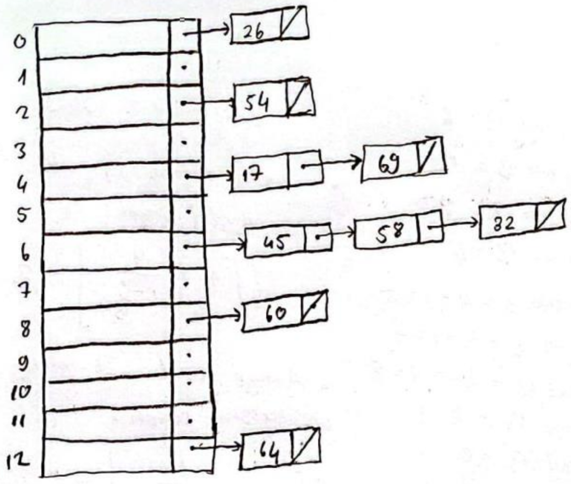
- Average number of probes for successful search:

$$\frac{-\log_e(1 - 9/13)}{9/13} \simeq 1.7025$$

- Average number of probes for unsuccessful search:

$$\frac{1}{1 - 9/13} = \frac{13}{4} = 3.25$$

c.)



Load Factor = 9/13

- Average number of probes for successful search:

$$1 + \frac{9/13}{2} = 1.346$$

- Average number of probes for unsuccessful search:

$$9/13 = 0.69$$

Question 3

For the implementation, there is an list of pointers held as an property of
FlightGraph class. This list holds the heads of the vertices (airports) following the
adjacency list implementation. Below are the analysis of the implemented methods.

Insert Operation

Insert operation always works on O(1) constant time. This is because the
new flights are inserted after the head pointer requiring no traversals.
Although it was a possibility to insert at the end of the list, that would make
the run-time O(n) which is less profitable. So, the worst-time is O(1).

List Operation

List operation works by traversing the specific part of the adjacency list
starting from the given airport and listing every airport directly connected
to the start. So, it works in O(n) where n denotes the number of airports
directly connected to the start. In this method, average-case, best case and
worst case all work in O(n).

Shortest Path Operation

Shortest path algorithm works recursively trying to find the shortest path
between two airports. Alternative to my implementation, a 2D array could
also have been used, but both would give the same results. To find the
shortest path, the algorithm visits every node starting from the given vertex.
Therefore, it needs the traverse the most of the nodes in the worst case.
As there are n nodes and each n node can visit n-1 nodes in the
worst case, the algorithm runs in O(n²) in worst case.

Minimize Cost operation

For this operation, two for loops are used with one inside of the other. The outer loop chooses an unvisited vertex and the inner one checks the reachable nodes from the selected vertex. So, the worst-case for the algorithm is $O(n^2)$. Alternatively, a min-heap could have been used and that would reduce the average run-time to $O(n\log n)$. However, it would require implementing the heap class, so I decided it would be easier to use the for loops.