Mehmet Bora Kurucu

21703404

CS202 HW4

Question 1)

a)

Answer: $2(\log_2(n + 1))$

Explanation: A 2-3-4 can't be taller than $(\log_2(n + 1))$. Number of black nodes in the path from root to deepest node is equal to the height of the 2-3-4 tree. In that path, number of red nodes can be equal to number of black nodes at max. Hence, for the tallest 2-3-4 tree, height is $(\log_2(n + 1))$. And the mentioned paths contains $(\log_2 n + 1)$ black nodes, and can contain $(\log_2(n + 1))$ red nodes. Hence, answer is = $(\log_2 (n + 1)) + (\log_2(n + 1)) = 2(\log_2 (n + 1))$.

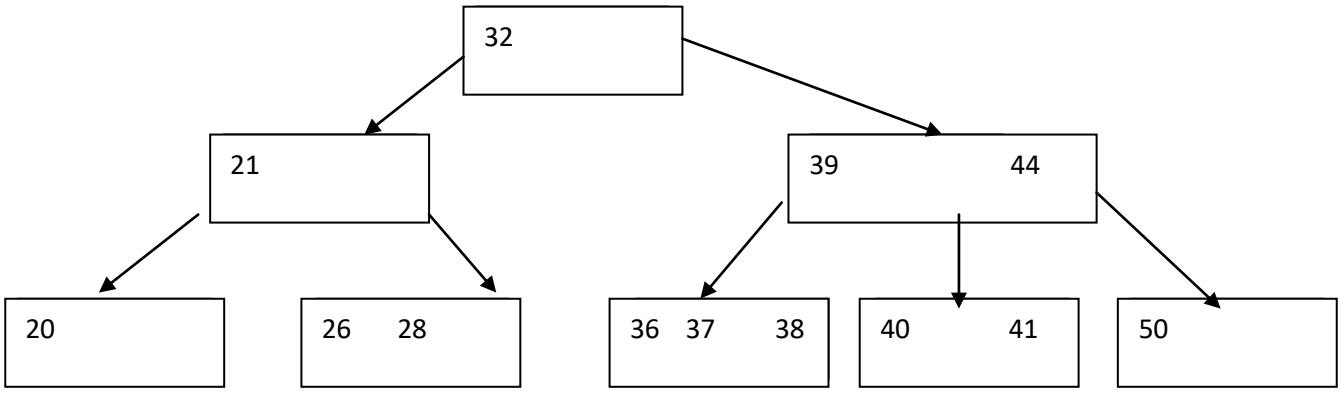
b)AVL tree requires more space .In addition to Binary Search Tree, AVL tree keeps balance factor and height for each node while Red Black trees only stores color as extra.

c) It stops when there is no unvisited vertice left for each vertex in the call stack.

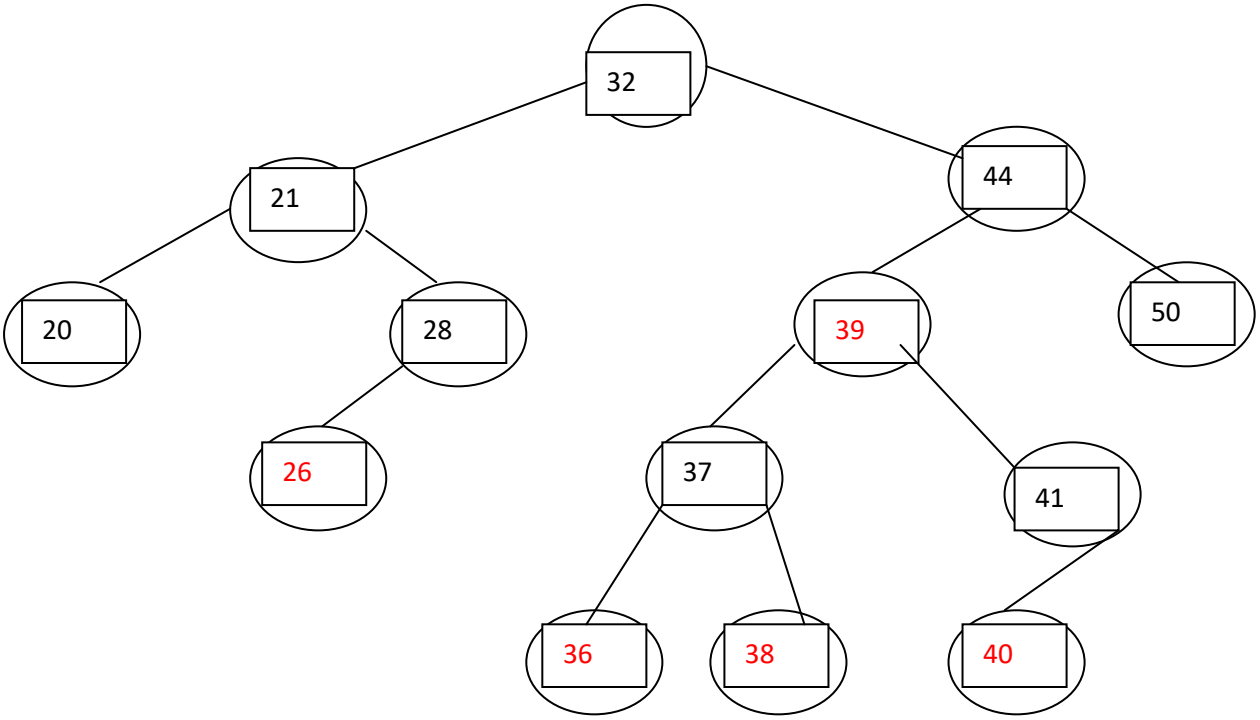
Question 2)

SECOND 36 IS IGNORED.

a)



b) I know normally red nodes indicated with half lines. But half lines were not available in word,so I indicated red nodes by coloring them in red.



c)

Inf = Infinite

Step	v	vertexSet	v1	v2	v3	v4	v5
1	-	v5	3	Inf	Inf	Inf	0
2	v1	v5 ,v1	3	4	Inf	4	0
3	v4	v5,v1,v4	3	4	6	4	0
4	v2	v5,v1,v4,v2	3	4	6	4	0
5	v3	v5,v1,v4,v3	3	4	6	4	0

The shortest path from v5 to v3 is the path v5,v1,v4,v3 with the weight 6.

d)

Node Visited	Queue(front to back)
Arad	Arad
Enqueqe neighbours	
Sibiu	Sibui Zerind Timisora
After Dequeue Sibui and enqueuing all neigbours	
Fagaras	Zerind Timisora Rimnicu Fagaras
After processing Zerind Timisora Rimnicu the front is Fagaras,dequeue Fagaras	
Bucharest	“Others before target”…… Bucharest

Arad,Sibiu,Fagaras,Bucharist is the path.Hence the result is 140 + 99 + 211 = 350

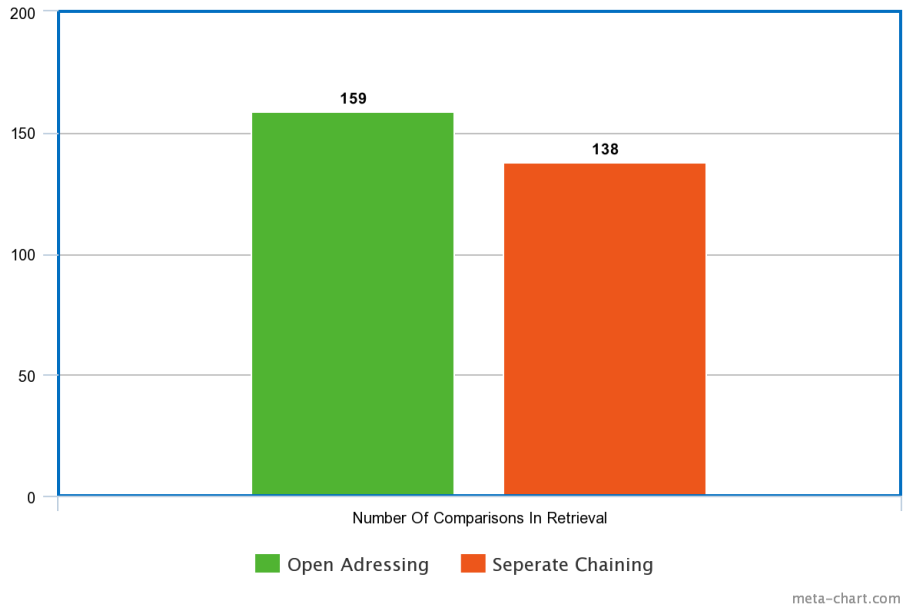
Question 3

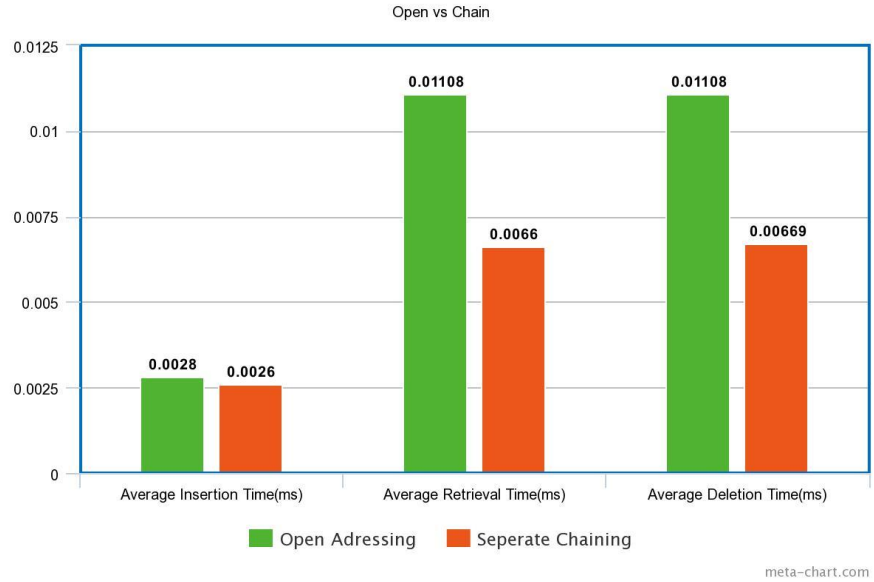
Table size should be prime, and approximately 1.5 times larger in than the input size. It should be prime to make mod operations to give the same value less frequently, so to lower the chance of collisions. It should be approximately 1.5 times larger to avoid problems created by clustering. If table size is too close to the input size, items will be loaded into a specific part of an array, which is clustering. Hence, for operations, mostly a big part of the cluster should be scanned to find the target. If the table size is large enough, clusters will not be that powerful. In open addressing table size is chosen as 199, since it is a prime number, and can handle 150 inputs well enough. If the input size is 50 but not 150, a prime size between 70-80 may be chosen. I chose the function

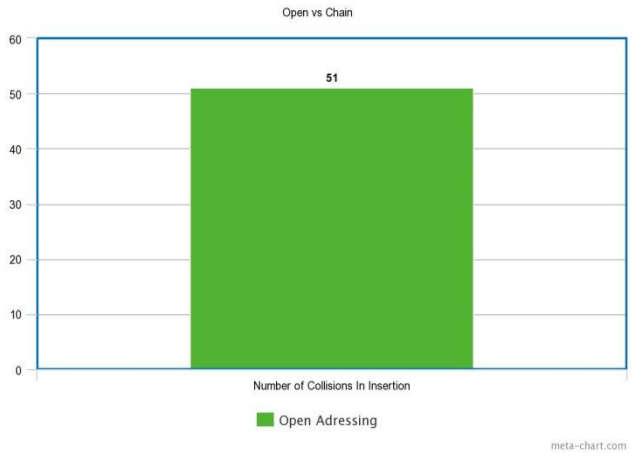
```
int hashVal = 0;

for (int i = 0; i < name.length(); i++)

 {

  if(name[i] != '\0' && name[i] != ' ')

            hashVal = 37 * hashVal + name[i];

 }hashVal %=tableSize;

 if (hashVal < 0)

            hashVal += tableSize;

 return hashVal;
```

Since this function is not likely to match 2 different strings to the same place. This functions sums the ascii char’s in the string, and takes mod with table size to ensure that the returned index is not larger than the tableSize, and adds the tableSize to it if it is negative to make sure that the returned value is not negative.It skips space chars to able to search(“banana leaf”) and (“ banana leaf “) both return positive. So it covers possible user mistakes. It multiplies sum with 37 every time to even decrease

the chance of collisions. So even if the total sums of two strings (“ cat”), (“tac”) are same, they will not be indexed same. I choose the key as string since search, deletion and more operation takes string as the parameter,as the key.







Inser6tion time is shorter in separate chaining, since you do not need to probe in collisions. You just insert to the head of the linked list. Retrieval time shorter too, since you traverse only the nodes with same hashIndex, in open addressing probing may result in checking different hashed items as well. Deletion itself is constant amount of work after search, so the time difference in deletions accused by the time difference in retrievals. Comparisons in retrieval is more in open addressing, since again by separate chaining you only compare the true candidates, the elements with the matched hashing. With open addressing you may even need to compare all the elements. The downside is, separate chaining requires more memory.