İsmet Alp Eren // 21703786 // Cs-202 / Section-1 // Hw4 – Hashing

**1-)**

a-)

60

47

50 | 51

37 | 40

33

49 | 55

4 | 26

35

30 | 45

28

8 | 11 | 19

3

b-)

**2-)**

a-)

max number of nodes can be: (3h-1)/2

max number of keys can be: (3h-1)

b-)

when “i” inserted tree will be expand. I assume pre-splitting occures

g | h

b | d | f

e

c

a

a

c

d

b

f

e

g | h | i

c-)

It is same running time with binary search tree sorting which is O(n logn) because they have same order. Smaller value in the left child, grater value in the right child.

d-)

Since root have to be black and if root has a red child then we can easily say, not each subtree is a red-black tree. Eg. 26 and 4 are in same node(first tree in that document).

**3-)**

1. First create a hash table with size n. Then loop over the given array. While looping check the hash table whether “target – arr[i]” (arr[i] is current item of the loop) is already in the hash table. If hash table contains “target – arr[i]” (target – arr[i] + arr[i] = target) then array contains pair which sum is target. (Hash Table insertion O(n) time, for n element O(n) time and search in hash table is O(1) time, for n element O(n) time. Total is O(2n) which is equal to O(n))

Linear probing

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Slot | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Content | 30 | 15 | 22 | 11 | 14 | 18 |  |

Quadratic probing

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Slot | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Content | 30 | 15 | 22 | 11 | 14 | 18 |  |