CS 440 HW 2

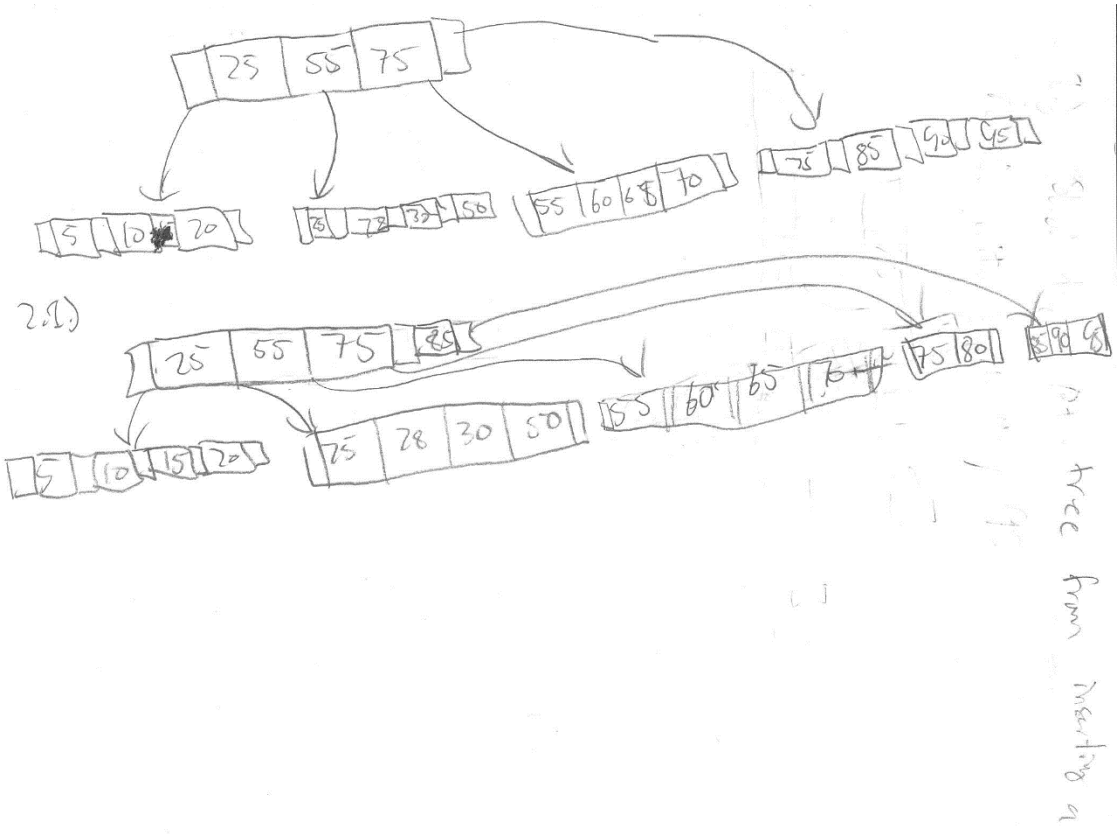
Thomas Noelcke noelcket

1/25/2018 CS 440 HW 2

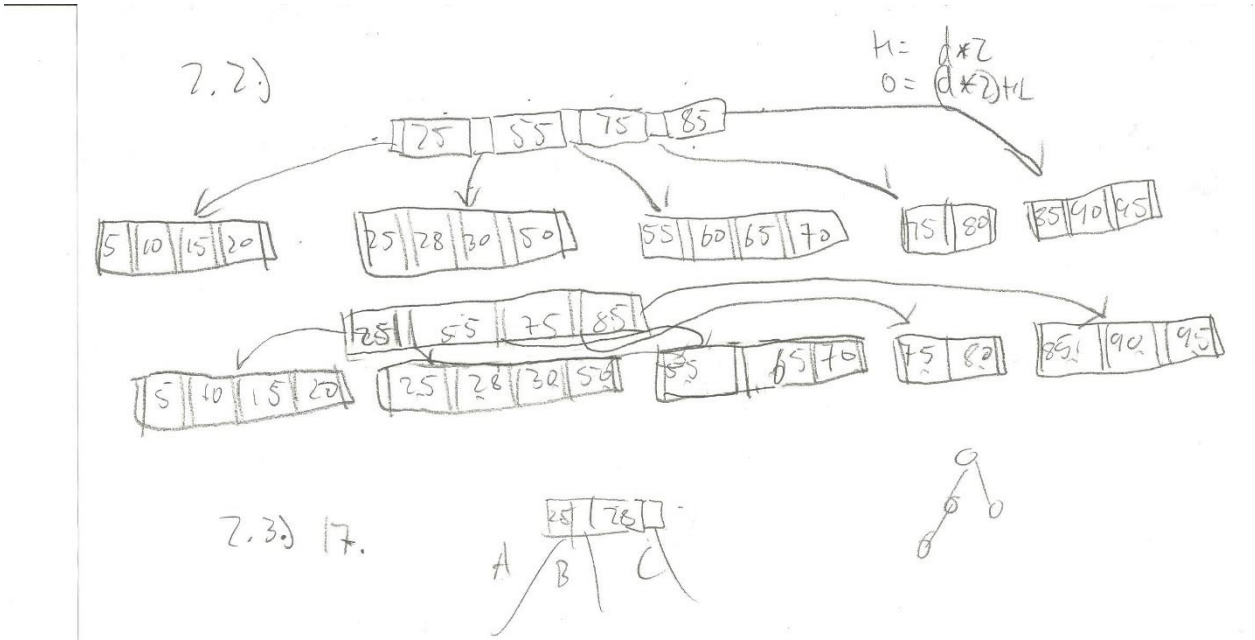
1.1. For this example, the best file structure would be a sorted file structure sorted based on the ID of the Customer record. I would choose this file structure because a sorted file structure offers better run time efficiency for search type operations. This file structure does have some disadvantages when it comes to writing to the file structure however as indicated in the question we don't expect this operation to happen often.

1.2. In this instance I would probably use a heap type files structure. As indicated by the question there will be a large number of records added to this file. A heap type file structure will offer the best performance when it comes to writing to the file. However, this file structure will be slower at querying using a group by statement as mentioned in the question prompt. However, if you are using a group by statement you are probably reading a decent amount of the records in the file any ways and likely won't suffer enough to justify giving up the write performance of a heap file structure.

2.1


25
55
75
5
10
20
55
60
70
2.1)
25
55
75
5
10
15
20
25
28
30
50
55
60
65
75
80
90

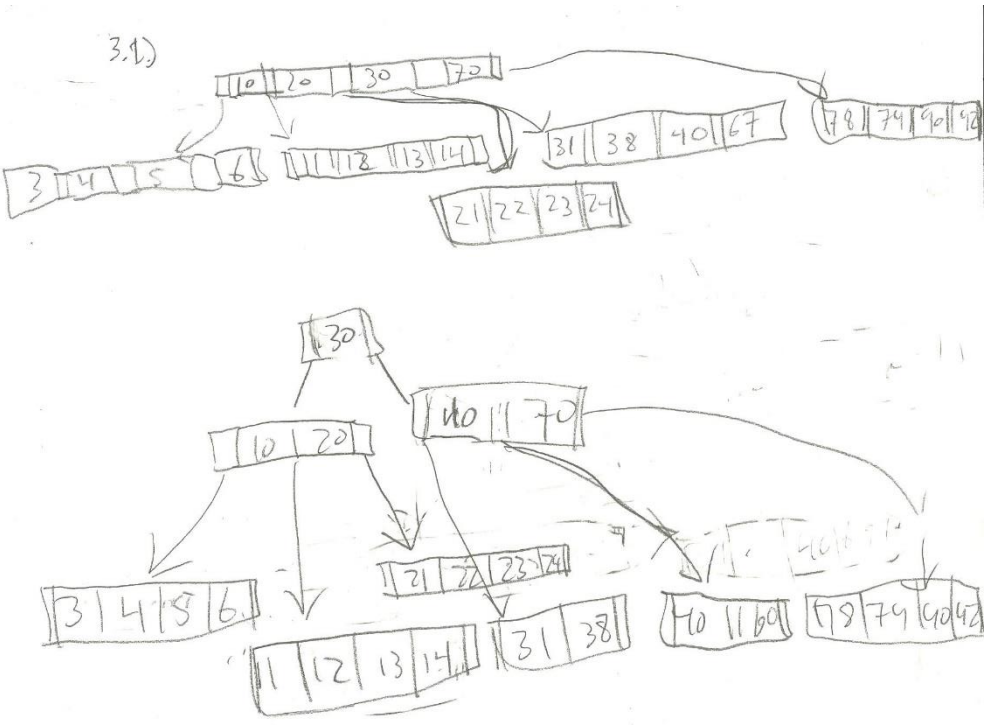
2.2



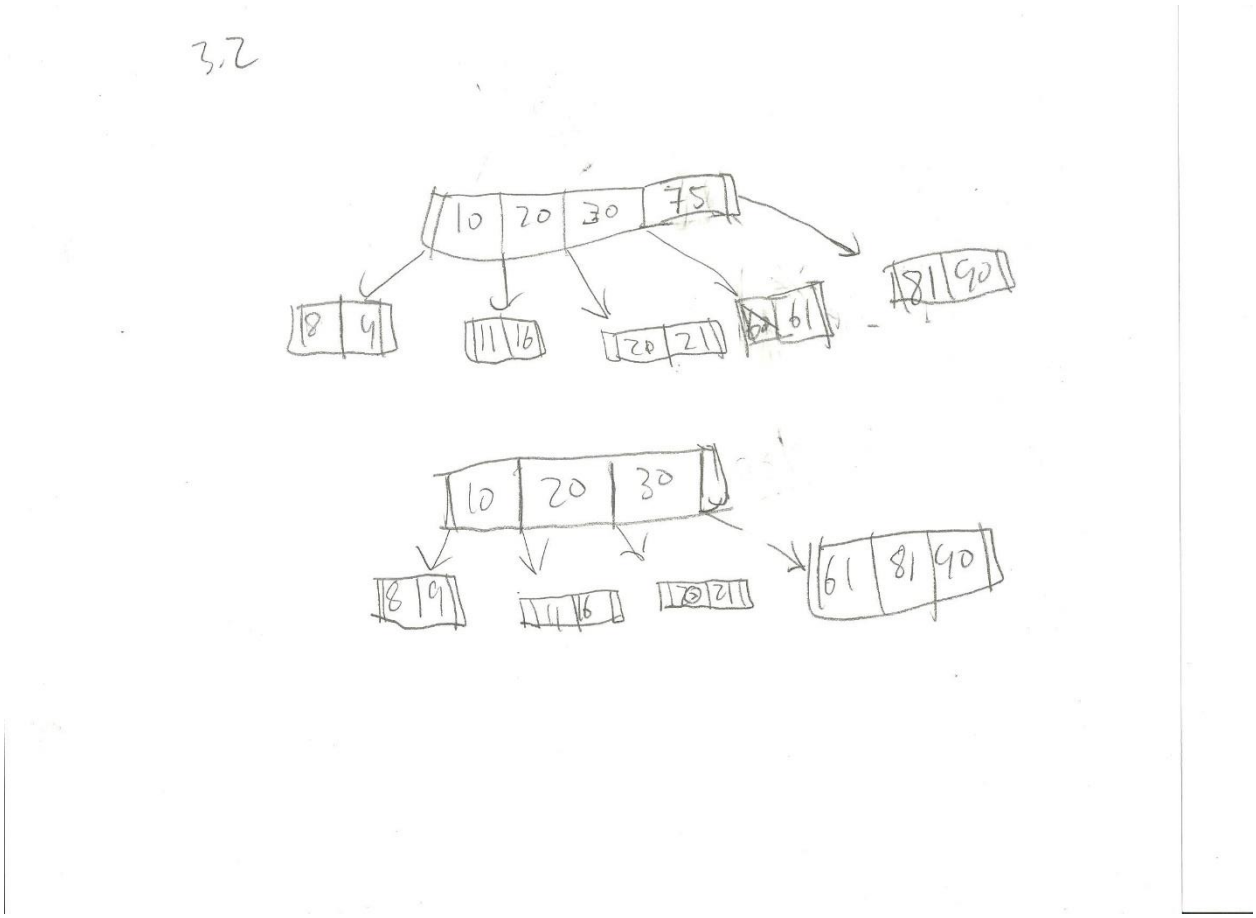
2.3 If you insert a value in the left most or second left most node such that the node won't be able to redistribute the extra values to it's neighbor this would cause the tree to increase in height. In other words a value between 5 and 30.

2.4 We know that the A will contain values less than 10. We also know that b will contain key values between 10 and 19 and C will contain values between 20 and 29. We can also say that A, B and C must be leaf's because the tree must be balanced. Finally, we can say that each node must have between 2 and 4 keys and can at most have 5 pointers.

3.1


3.1)
10
20
30
70
3
4
5
6
11
12
13
14
31
38
40
67
78
79
90
98
21
22
23
24
30
40
70
10
20
3
4
5
6
21
22
23
24
11
12
13
14
31
38
40
69
78
79
90
98

3.2



4.1 We know the following equation shown below:

$$k_v = Key\ value\ size$$

$$r_p = record\ pointer\ size$$

$$b_s = block\ size$$

$$2d\ k_v + (2d + 1)r_p \leq b_s$$

Given this equation and the values in the problem statement we can say:

$$2d * 16 + (2d + 1) * 32 \leq 28 * 8$$

$$32d + 32(2d + 1) \leq 224$$

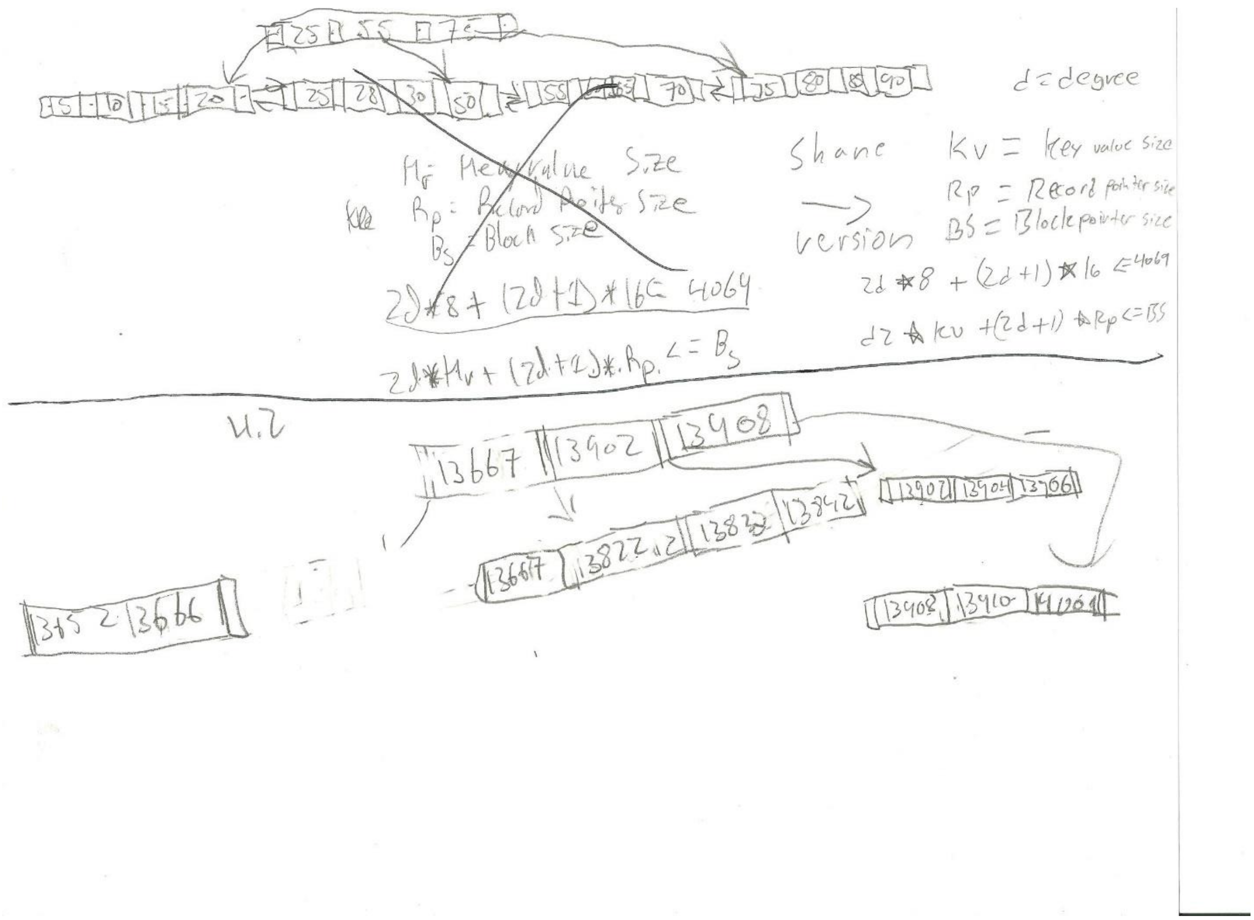
$$d + (2d + 1) \leq 7$$

$$3d \leq 6$$

$$d \leq 2$$

So we can Say that the maximum degree of the tree is 2.

4.2



d = degree
Shane
version
Kv = Key value size
Rp = Record pointer size
BS = Block pointer size
2d * 8 + (2d+1) * 16 <= 4069
2d * Kv + (2d+1) * Rp <= BS
Bs = Block size
2d * 8 + (2d+1) * 16 <= 4064
13667
13902
13408
13667
13822
13842
13902
13904
13906
13408
13410
3666