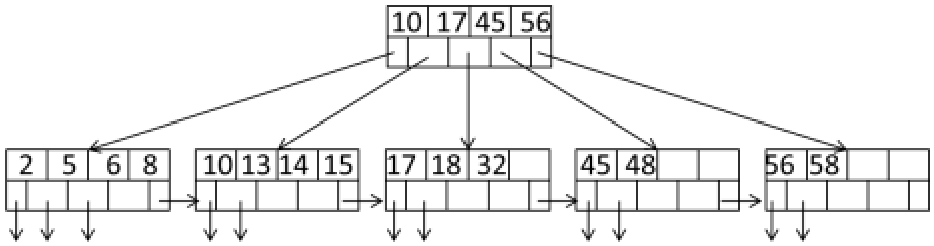
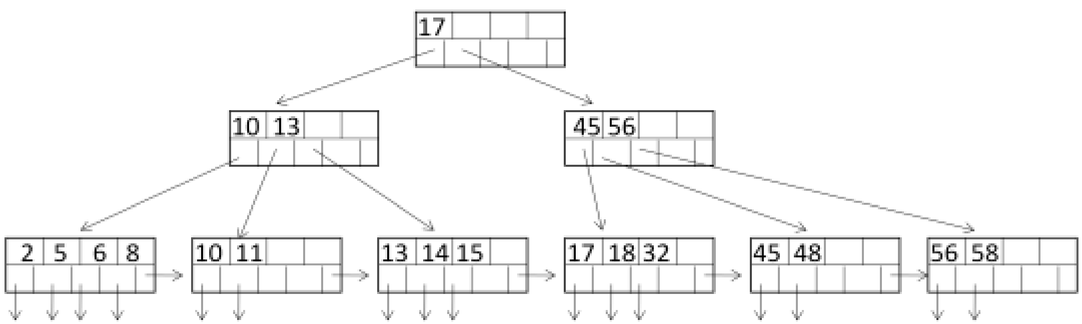
1. [40 points] Consider the following B+tree for the search key "age. Suppose the degree d of the tree = 2, that is, each node (except for root) must have at least two keys and at most 4 keys.



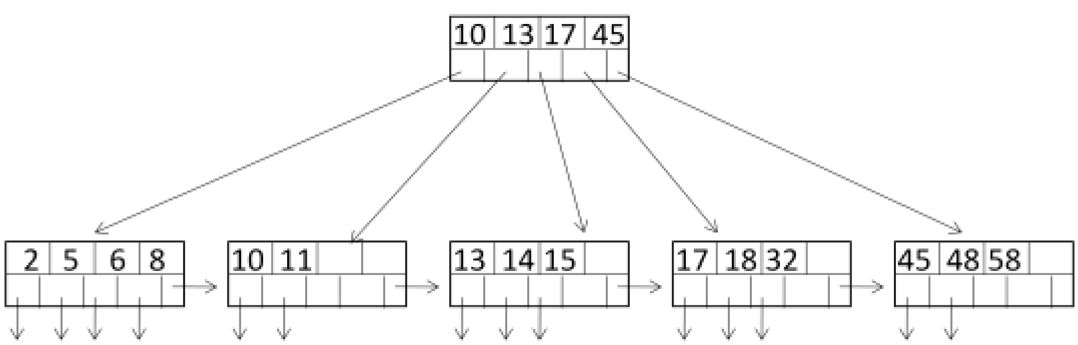
a. Describe the process of finding keys for the query condition "age > 7 and age < 20". How many blocks I/O's are needed for the process?

Ans: First we will solve the portion age > 7. To find age > 7, we first go to the root node. The first value is 10 and we want to find 7, so we will follow the first pointer as 7 < 10. This brings us to the first node in 2nd level. We see that 7 > 6 and 7 < 8, so we will start traversing from this node towards right until we find a value which is greater than equals to 20. We will stop at the 3rd node at Level 2. So in total, we will **read 4 blocks** (1 in Level 1 and 3 in level 2)

b. Draw the updated tree after inserting 11 into the above tree.



c. Draw the updated B+ tree after deleting 56 from the tree obtained in part b.



2. [60 points] Consider natural-joining tables R(a, b) and S(b,c). Suppose we have the following scenario.
   i. R is a clustered relation with 500 blocks and 1,000 tuples
   ii. S is a clustered relation with 1,000 blocks and 20,000 tuples
   iii. S has a clustered index on the join attribute b
   iv. V(S, b) = 2 (recall that V(S,b) is the number of distinct values of b in S)
   v. 100 pages available in main memory for the join
   vi. Assume the output of join is given to the next operator in the query execution plan (instead of writing to the disk) and hence the cost of writing the output is ignored.

   Describe the steps (including input, output, and their size) in each of the following join algorithms. What is the total number of block I/O's needed for each algorithm?

In the given question, we have

B(R) = 500, T(R) = 1000

B(S) = 1000, T(R) = 20000, V(S,b) = 2

M = 100

a. <u>Nested-loop join with R as the outer relation</u>

Here we will use the formula:

B(R) + ceil(B(R)/(M-2)) * B(S) Plugging the values from the above formulas, we get

500 + (ceil(500/98) * 1000)) = **6500 block I/O**

b. <u>Nested-loop join with S as the outer relation</u>

Here we will use the formula:

B(S) + ceil(B(S)/(M-2)) * B(R) Plugging the values from the above formulas, we get

1000 + (ceil(1000/98) * 500)) = **6500 block I/O**

c. <u>Sort-merge join</u>

First we need to check the condition that

B(R) + B(S) <= M^2 Putting the values, we get
500 + 1000 <= 10000 which is true

If the above is true, then the total I/O cost of this operation is:

3 * B(R) + 3 * B(S) = (3 * 500) + (3 * 1000) = 1500 + 3000 = **4500 block I/O**

d. Simple sort-based join

Here we have to make check few assumptions. This algorithm is used when we have large number of tuples with the same value on the common attributes. This seems to be true here because V(S,b) = 2 in our case. Secondly, we have to check:

$B(R) <= (M-2)^2$ ======> 500 <= 9604 True and
$B(S) <= (M-2)^2$ ======> 1000 <= 9604 True then

4* (B(R) + B(S)) is the cost of sorting completely both the relations and B(R) + B(S) is the cost of merge.

Hence Total cost = 5 * (B(R) + B(S)) = 5 * (500 + 1000) = 5 * (1500) = **7500 block I/O**

e. Partitioned-hash join

First we need to check the assumption that

$Min(B(R), B(S)) <= M^2$ ======> Min(500,1000) = 500 <= 10000 which is true. Then total cost is

Total cost = 3 * (B(R) + B(S)) = 3 * (500 + 1000) = 3 * 1500 = **4500 block I/O**

f. Index join (ignore the cost of index lookup)

Here we are given that S has clustered index, so we will use the following formula:

B(R) + T(R) * B(S)/V(S,b)

Plugging the values from given information, we have:

500 + ((1000 * 1000)/2) = **500500 block I/O**