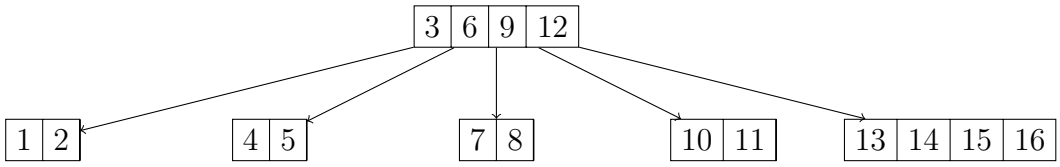
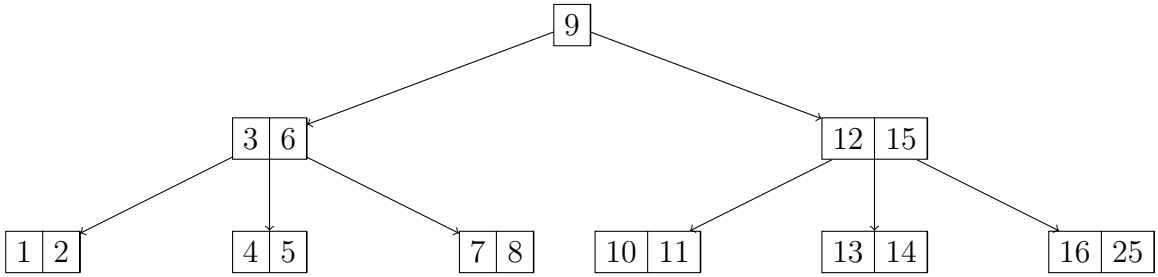
Databases (Sheet #3)

Marius Gavrilescu

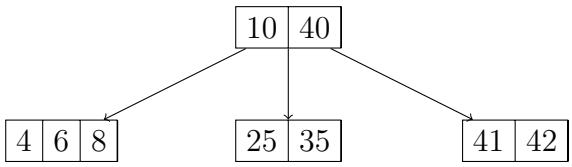
1. (a)



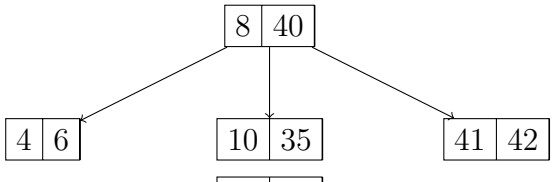
After insertion of 25, it becomes:



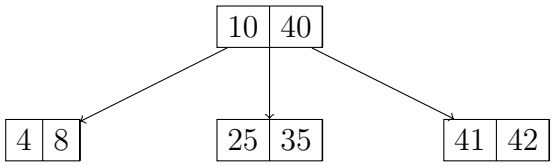
(b)



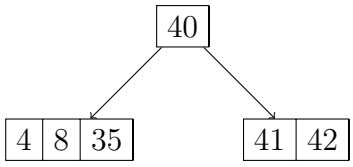
After deletion of 25, it becomes:



(c)



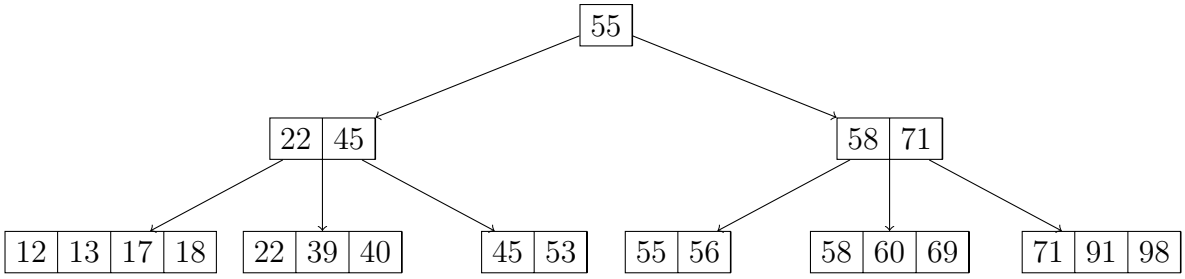
After deletion of 25, it becomes:



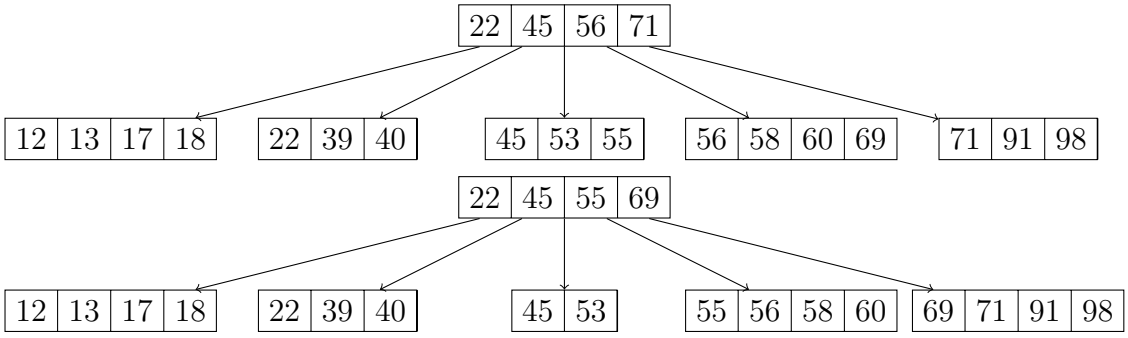
2. Assuming *sid*s are uniformly distributed between 1 and 100000, the first query will return half of all sailors, who will reside in the first half of all pages. Hence we need to read 1 + 4 + 250 = 255 pages.

   For the second query there's a single matching page, so we need to read about 1 + 4 + 1 = 6 pages.

3. According to the Insert Algorithm at the end of lecture 10, there is a deterministic algorithm for inserting an element into a B+-tree. Its result follows.



Alternatively, we can redistribute to the left or to the right, producing the following two trees:



4. We do a block nested loops join with Employees in the outer loop and WorkedOn in the inner loop.

   As we have 5 buffer pages, we will have a 3-page block for Employee and a 2-page block for WorkedOn.
   Page I/Os needed are $\frac{1000}{3} + \frac{1000}{3}\frac{10000}{2} \approx 1.66M$.

   A sort-merge join will sort both tables and merge them.
   Page I/Os needed are $1000 * \log(1000) + 10000 * \log(10000) + 1000 + 10000 \approx 0.15M$.

   Clearly the sort-merge strategy uses significantly less memory than block nested loops.

   In either case we'd filter by salary after the join. The cost of filtering is not dependent on the join algorithm used.