In this week’s hand-in we consider clustering indexes and non-clustered indexes. In a clustering index we assume that all tuples with the same clustering-values are stored in at most $O(c)$ blocks, where $c$ is the minimum possible number of blocks to store them in. In a non-clustered index only pointers to tuples are stored in the index structure (e.g., in the leaves of the B-tree).

1. We want to do a select operation $\sigma_{a_i=374}(R)$, where $a_i$ is an attribute of $R$. Denote by $t = |\sigma_{a_i=374}(R)|$, the number of tuples in the result.

Consider four types of indexes for $R$ on $a_i$:

(a) Hash index, which is a clustering index.
(b) Hash index, not clustered.
(c) B-tree index, which is a clustering index.
(d) B-tree index, not clustered.

Denote by $u$ the number of tuples of $R$ that hash to the same bucket as 374, and by $n$ the degree of the B-tree.

What is the worst case I/O-complexity (as a function of $|R|$, $t$, $u$, and $n$) of select using each of the four types of indexes?

2. We want to do join, $R(X,Y) \bowtie S(Y,Z)$, where $Y$ is a primary key. There is a hash index on $Y$ for $R$ (clustering index). The number of buckets is much larger than the number of blocks in main memory and one block suffices to store each bucket. There is a B-tree index on $Y$ for $S$ (also a clustering index).

Consider three algorithms for join:

(a) Scan $S$: for each tuple $s \in S$ look up the join-value for $s$ in the index of $R$.
(b) Sort $R$ and merge the two relations (as in the sorting based algorithms) using the sorted order in the B-tree index of $S$.
(c) Scan $R$: for each tuple $r \in R$ look up the join-value for $r$ in the index of $S$.

For which sizes of $R$, $S$, and the main memory $M$ is (a), (b), or (c) the most efficient (I/O-complexity)? (Assume that $R$ and $S$ are both too large to fit in main memory.)

**Other exercises for discussion on March 20**

1. GUW 15.3.4 on page 737.
2. GUW 15.5.4 on page 756.
3. GUW 15.7.2 on page 770.