1 Representation of relations

In this problem we consider a relation \( R(a, b) \), where \( a \) and \( b \) are integers (of type INT). We let \( B > 1 \) denote the number of integers that fits in a disk block. Suppose that \( R \) consists of \( N \) tuples, \( \{(a_1, b_1), \ldots, (a_N, b_N)\} \), sorted such that \( a_1 < a_2 < a_3 < \cdots < a_N \).

There are two natural ways of representing the relation on disk, ordered according to \( a \):

**Horizontal:** \( a_1, b_1, a_2, b_2, \ldots, a_N, b_N \) (this is the standard order).

**Vertical:** \( a_1, a_2, \ldots, a_N, b_1, b_2, \ldots, b_N \).

Some DBMSs allow the user to specify that vertical order should be used (this is an example of *vertical partitioning*). We assume that there are no updates to the data, and it is thus stored as a sequential file. The size \( N \) of the relation is known.

**a)** How many I/Os are needed to read the \( K \) smallest values of \( a \), i.e., \( a_1, \ldots, a_K \), in each of the two representations? State your answers as functions of \( K \) and \( B \) (exact numbers, no asymptotic notation).

**b)** How many I/Os are needed to read the \( K \) smallest values of \( b \) in each of the two representations? State your answers as functions of \( N, K, \) and \( B \) (exact numbers, no asymptotic notation).

**c)** Assume that there in no index on \( R \). How many I/Os are needed to find the tuple with a particular value of \( a \) in each of the two representations? State the worst case number of I/Os for the best algorithms you can think of (exact numbers, no asymptotic notation).

We now consider a third alternative representation, the *multi-sorted* representation. Assume that the number of tuples in \( R \) is a perfect square, i.e., that \( \sqrt{N} \) is an integer. The idea is to change the horizontal representation by splitting it into \( \sqrt{N} \) intervals of \( \sqrt{N} \) tuples, and sorting each interval according to the value of \( b \). An example instance with \( N = 9 \) is the following (we mark tuples by parentheses and intervals by square brackets for readability):
d) Show that in the multi-sorted representation, it is possible to search for a particular value of $a$, as well as a particular value of $b$, in $O(\sqrt{N} \log N)$ I/Os (without any index). You should describe search algorithms achieving this I/O bound (or better). Can you improve the representation, in terms of search time for particular values?

2 \hspace{1em} \textbf{B$^+$-trees}

Consider the following setting: We have a disk with block size 2404 bytes, and want to construct a B$^+$-tree index on an integer attribute of a relation $R$. An integer occupies 8 bytes of space, and a pointer uses 4 bytes of space. The size of a tuple in $R$ is 100 bytes. The leaves of the B$^+$-tree contain pointers to the tuples of the relation, i.e., the index is dense. Each node in the B$^+$-tree is contained in 1 disk block.

a) What is the largest possible degree of an internal node in the B$^+$-tree?

b) In the above setting, what is the size of the largest relation that can be indexed by a B$^+$-tree with two levels of internal nodes?

In GUW it is described how keys can be deleted from a B$^+$-tree. A deletion may require several I/Os in addition to those needed for locating the key. An alternative strategy would be to use tombstones to mark keys as deleted. This could always be done using one I/O.

c) Discuss possible disadvantages of the tombstone approach. Consider:

- The space occupied by deleted keys.
- The time complexity of searching for a key in the B$^+$-tree.
- The time complexity of range queries.