TU Munich, Department of Informatics
Chair XXV: Data Science and Engineering
Prof. Dr. Thomas Neumann

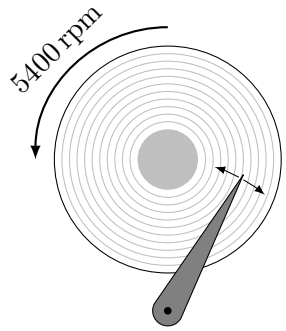
Exercise for Database Systems on Modern CPU Architectures
Summer Term 2019

Philipp Fent, André Kohn ({fent,kohna}@in.tum.de)
https://db.in.tum.de/teaching/ss19/moderndbs/

Sheet Nr. 01

Exercise 1

The following illustration shows a schematic illustration of a hard disk. The hard disk consists of a rotating platter that stores the data and a moving actuator arm that can read and write on this platter. Before the actuator can actually access any data, it first needs to position itself over the corresponding track and wait for the right sector to rotate under it.



Assume the drive rotates with 5400 rpm and the actuator arm can accelerate (and decelerate) with a constant rate of $50\,g$. We now want to access data on a track 1 cm apart from the current position of the actuator head.

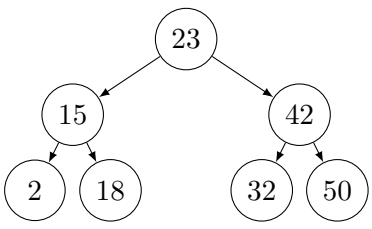
In what time range can we expect to read the data?

For your calculations, you can use the following formulas for the gravitational acceleration g and the linear acceleration.

- $g = 9.8\,\mathrm{ms}^{-2}$
- $d = \frac{1}{2}\,a\,t^2$

Exercise 2

Your college wants to accelerate lookup performance of his database implementation. Therefore, he implements a binary search tree and stores it on disk.



Each entry in this tree consists of:
An 8 Byte key, 8 Byte value, and two 8 Byte pointers, pointing to the sub-trees containing smaller keys and larger keys respectively.

Your college stores 64 K entries in this tree. Each access of a node requires a single random seek of the disk. How long does it take to search this tree for a value?

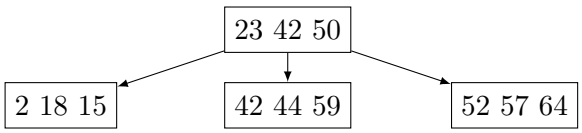
Use the following performance numbers for the disk:

- Seek time: 5 ms
- Bandwidth: 150 MByte/s

Now assume the tree can be read continuously from disk. How long does it take to scan the whole tree for a value?

Exercise 3

Instead of using a binary tree, you decide to store more keys and links to sub-trees in your tree. Assume we still need the same 32 Byte for each entry in this k-ary search tree.



How large would you need to choose k, such that accessing one of 64 K values is faster using the tree than scanning the full dataset?