Exercise Sheet 8

Ex 1

(A) = Quicksort n numbers

(B) = put n numbers in a empty hashtable

Operations of A and B worst case

(A): The worst case is, when the pivot doesn't split the list (e.g. it is the smallest or highest element in the list) for all iterations.

On each recursion the length of the recursion is reduced by one.

There are n-1 recursions with an average length of n/2

$\Rightarrow A = O(n^2)$ OK

(B): The worst case is, when all ellements are maped in the same bucket

The bucket selection has costs O(1)

The costs of appending on a ~~d~~ linked list is O(1)

$\Rightarrow B = O(1)$

Since we are using universal hashing (the one-bucket case does not occur) we have a total runtime of O(n) = O(n) * O(1).

Block operations:

(A): The memory access is randomly. Two following write operatius are not necesary in a close memory adress. The cache as to fetch his content more often

(B): The memory access is in a more predictable way. If an adress is read, its very likly that the adresses close by are read as well. This should be cach efficient.

The mapping should be far more efficient, because the difference of complexity is dominating over the cache efficiency.

We can argue, that for small n, the impact of cache efficency could have an impact.

Ex. 4 Quicksort is somwhat quadratic and maping close to linear (but increasing at the end)
The increas at the End is caused by the cache efficency, which begins to have an impact with increasing data size → more cache operations required OK

Exercise 1:
A) Since quicksort starts from left and right and swaps elements larger / smaller than the pivot element we get a reduction linear to the block size.
So the runtime is O(n^2 / B) = O(n^2)
B) The elements are separated into buckets. Since the buckets might not fit into the cache there might be no speedup (each buckets has to be fetched on each insert).
=> O(n) runtime