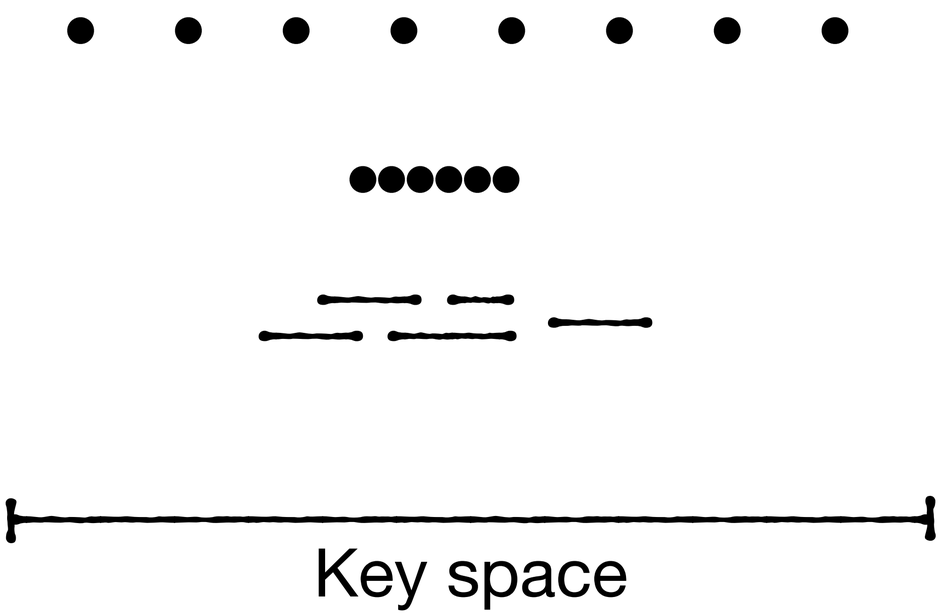
Question 1

Consider an LSM-tree and the three query workloads below. For each workload, comment on the usefulness of having Bloom filters and/or a buffer pool (block cache).

(1) uniformly randomly distributed get queries

(2) get queries with lots of spatial locality

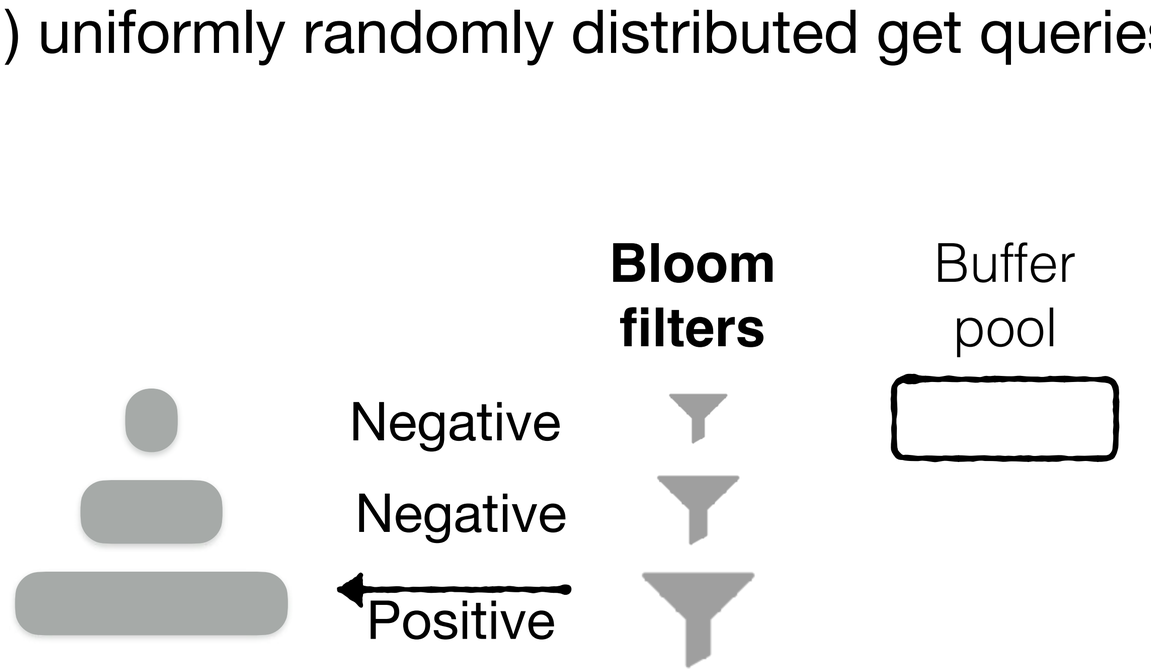
(3) scan queries with lots of spatial locality



Question 1

Consider an LSM-tree and the three query workloads below. For each workload, comment on the usefulness of having Bloom filters and/or a buffer pool (block cache).

(1) uniformly randomly distributed get queries



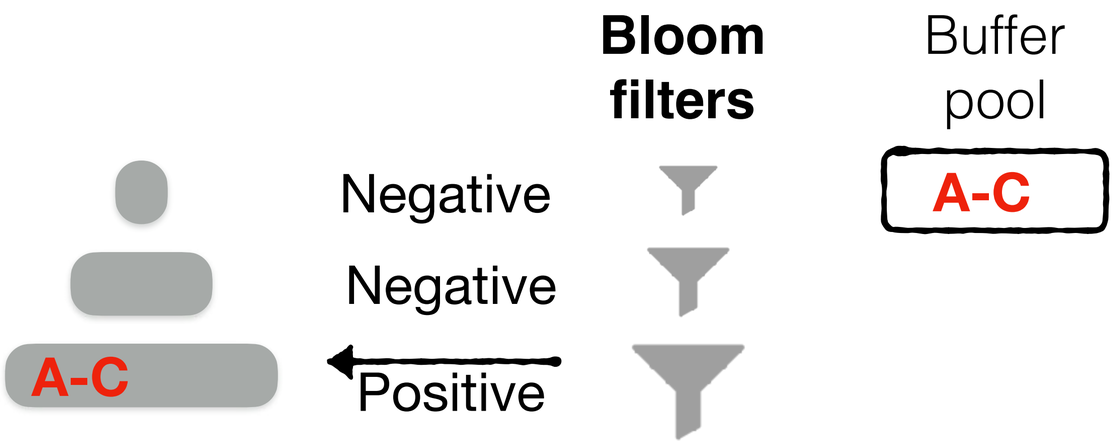
Data recently accessed isn't more likely be accessed again, so a buffer pool isn't so useful, unless it is large enough to fit a high percentage of the data.

In contrast, Bloom filters eliminate storage accesses that do not retrieve the target entry, so they help a lot.

Question 1

Consider an LSM-tree and the three query workloads below. For each workload, comment on the usefulness of having Bloom filters and/or a buffer pool (block cache).

(2) get queries with lots of spatial locality ●●●●●



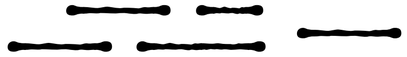
A buffer pool is very useful here to keep hot data pages in memory.

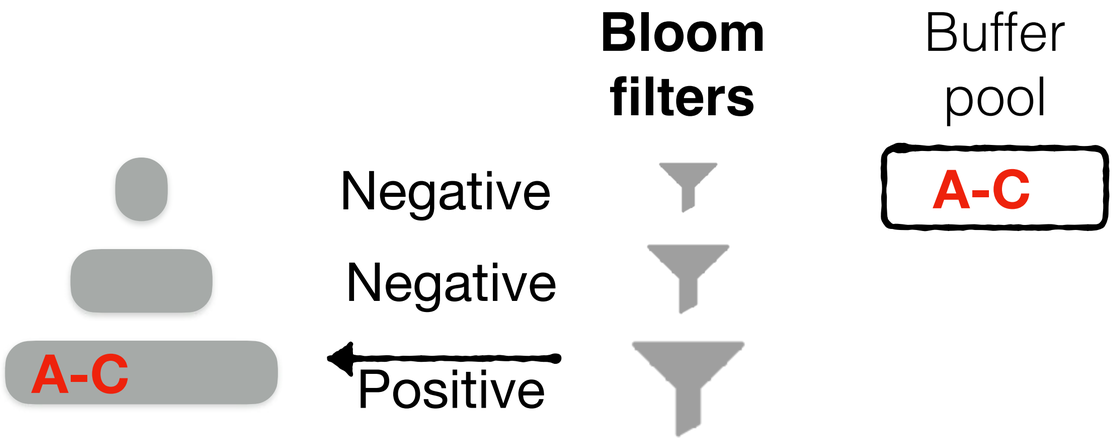
If the working set does not fit entirely in memory, the Bloom filters would still help by fetching data into memory using fewer I/Os.

Question 1

Consider an LSM-tree and the three query workloads below. For each workload, comment on the usefulness of having Bloom filters and/or a buffer pool (block cache).

(3) scan queries with lots of spatial locality





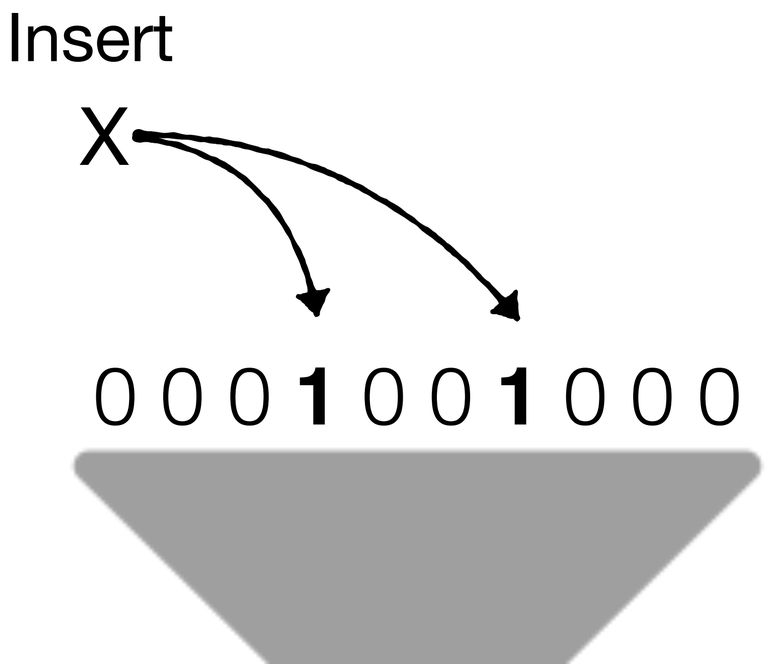
A buffer pool is also useful to keep hot scanned data pages in memory.

The Bloom filters do not help at all as they do not support range queries.

Question 2

Can a Bloom filter handle deletes? Why or why not?

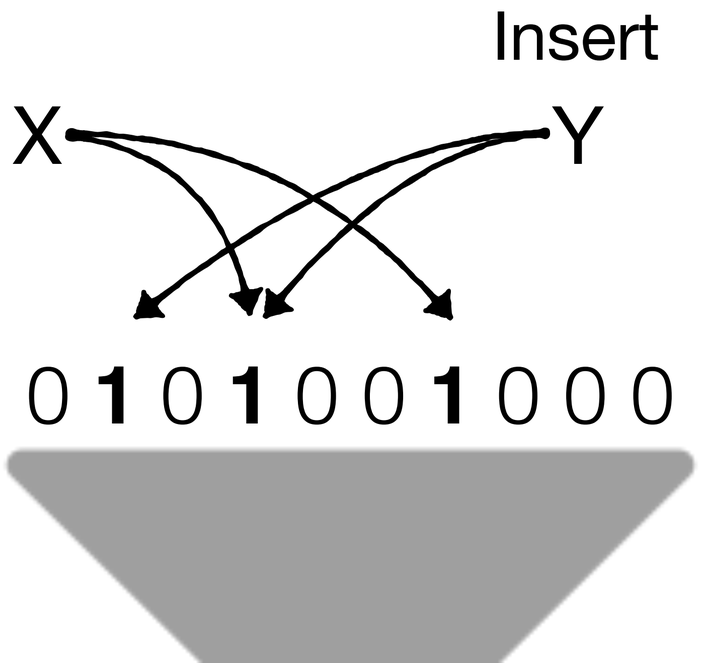
No, as this could lead to false negatives.



Question 2

Can a Bloom filter handle deletes? Why or why not?

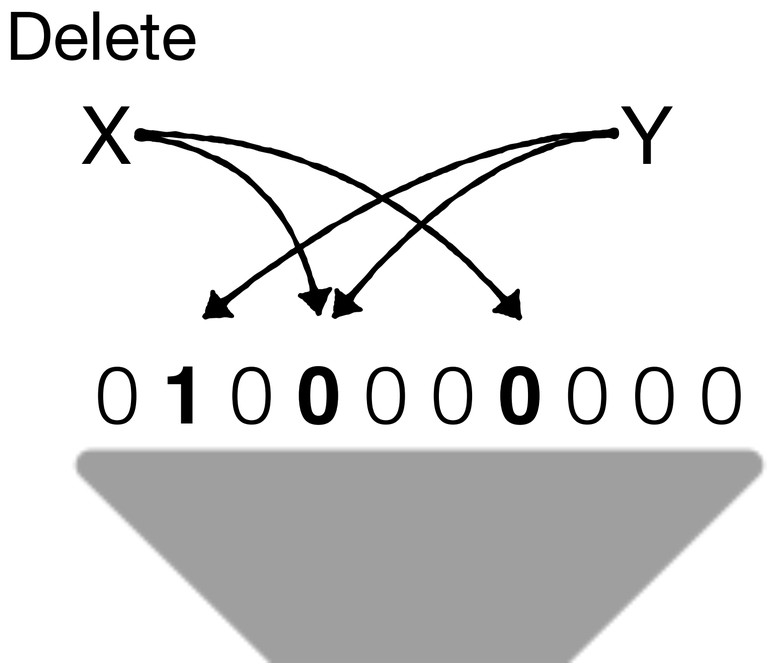
No, as this could lead to false negatives.



Question 2

Can a Bloom filter handle deletes? Why or why not?

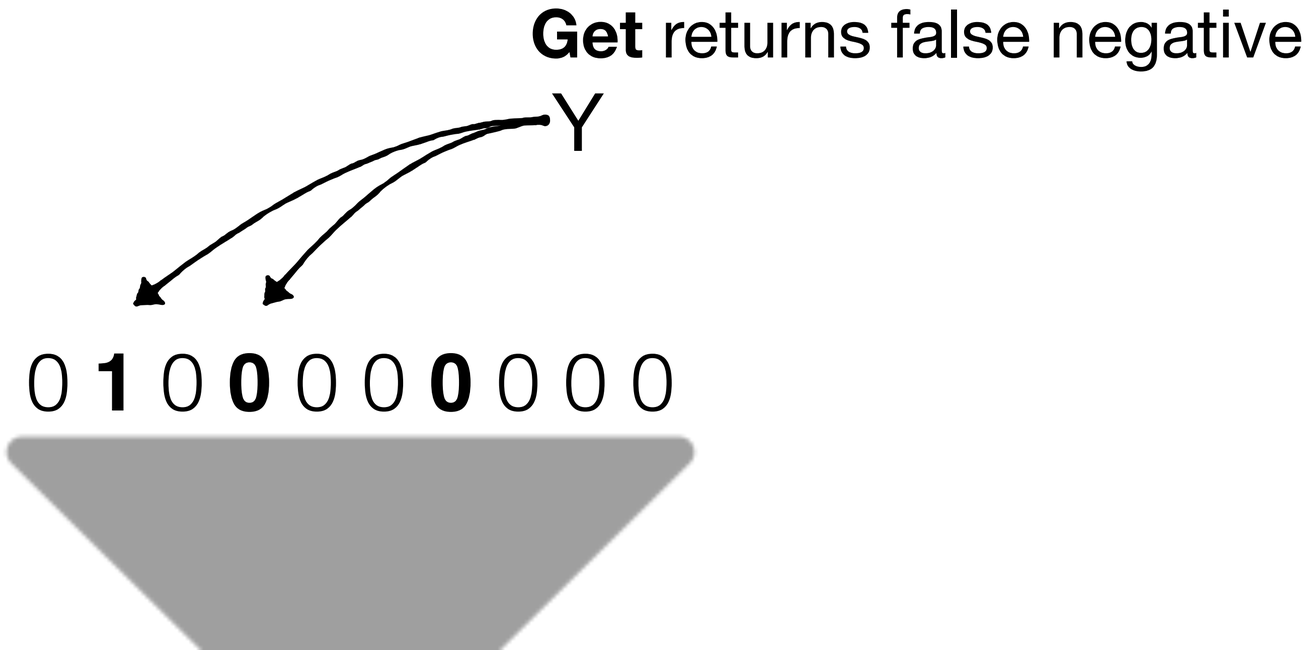
No, as this could lead to false negatives.



Question 2

Can a Bloom filter handle deletes? Why or why not?

No, as this could lead to false negatives.



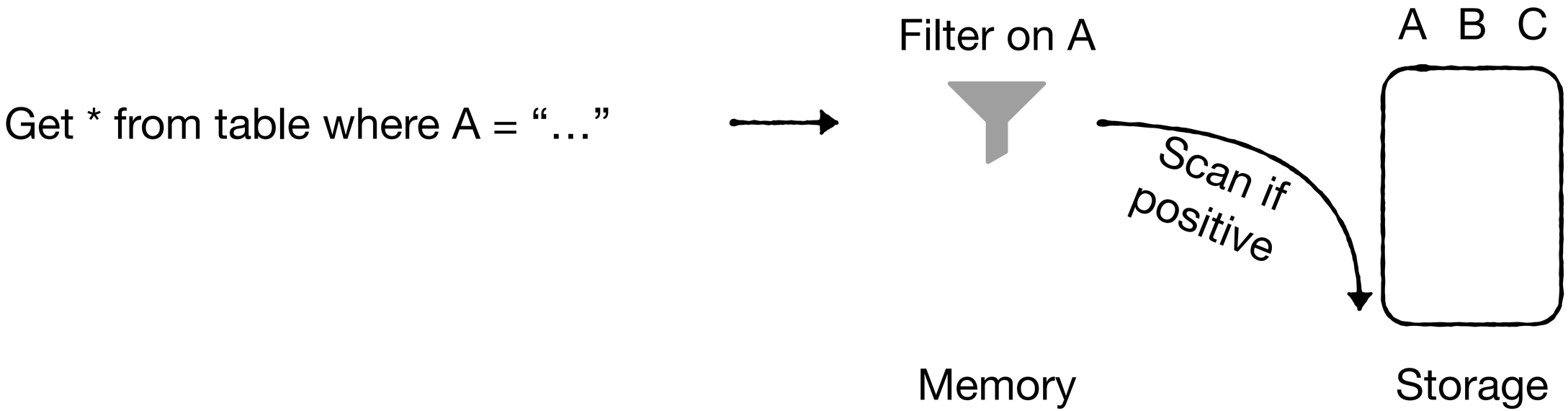
Question 2

Can a Bloom filter handle deletes? Why or why not?

Good exam answer: While it's possible to set the corresponding bits for a given key from 1's back to 0's in a Bloom filter to reflect a deletion, this can lead to false negatives when querying for other keys that had also mapped to at least one of the same bits. This is not acceptable in DB applications as it would lead to data loss (not finding relevant existing data for queries searching for it).

Question 3

Suppose we apply a Bloom filter on a given column in a DB table. What's the influence on the filter as the data in the table grows? Should we adapt the filter in any way?

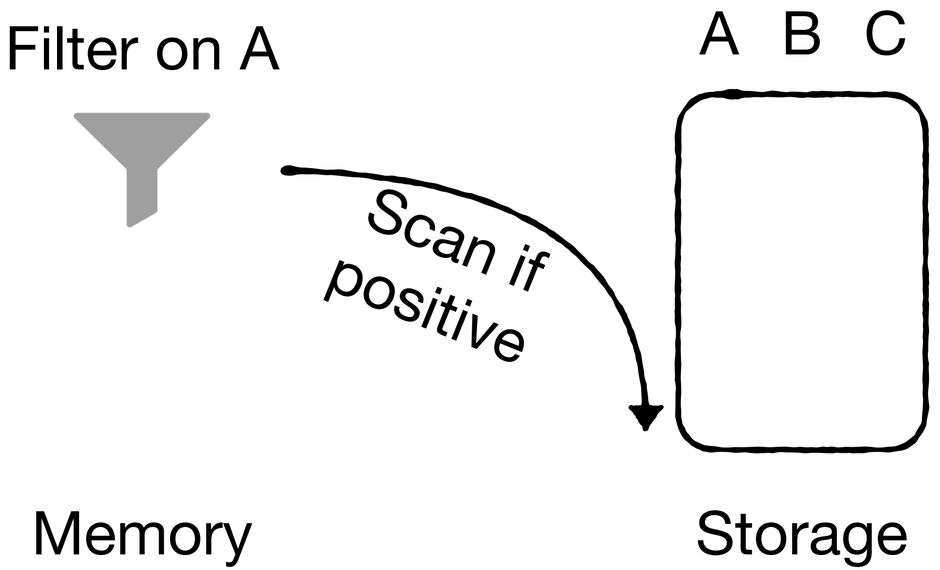


Question 3

Suppose we apply a Bloom filter on a given column in a DB table. What's the influence on the filter as the data in the table grows? Should we adapt the filter in any way?

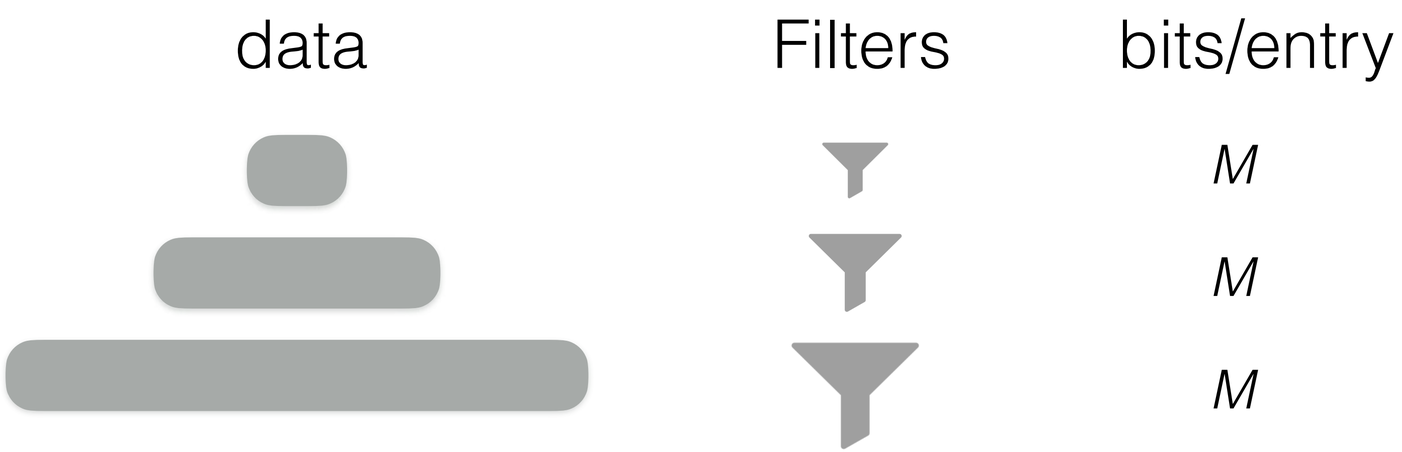
If new unique values of column A are inserted, the false positive rate for the filter would deteriorate.

To compensate, we can rebuild the filter by scanning the table whenever the FPR increases above some threshold.



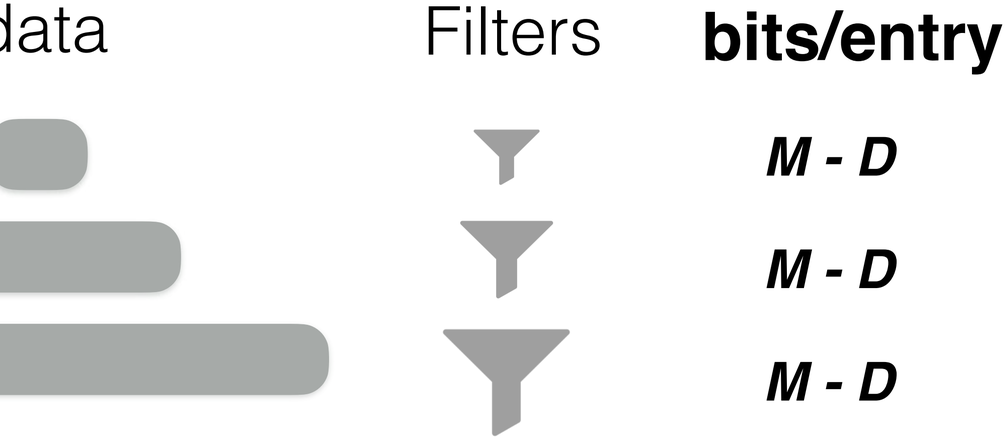
Question 4

We have a leveled LSM-tree with a uniformly allocated Bloom filter with M bits/entry for each level. Our query workload consists of get requests to keys that usually exist at the largest level. Suppose we are near our memory capacity and should free some space used by the filters. Option 1 is to reduce the number of bits/entry for each filter. Option 2 is to drop the filter for the largest level. Compare and contrast these two approaches.



Question 4

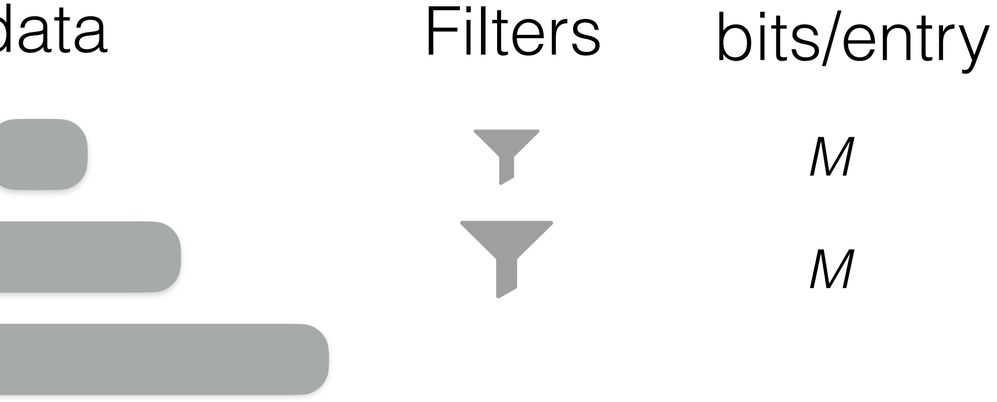
We have a leveled LSM-tree with a uniformly allocated Bloom filter with M bits/entry for each level. Our query workload consists of get requests to keys that usually exist at the largest level. Suppose we are near our memory capacity and should free some space used by the filters. Option 1 is to reduce the number of bits/entry for each filter. Option 2 is to drop the filter for the largest level. Compare and contrast these two approaches.



Option 1 would lead to more false positives and thus worse query performance. Furthermore, if we need this change immediately, it requires scanning the data to build new smaller filters.

Question 4

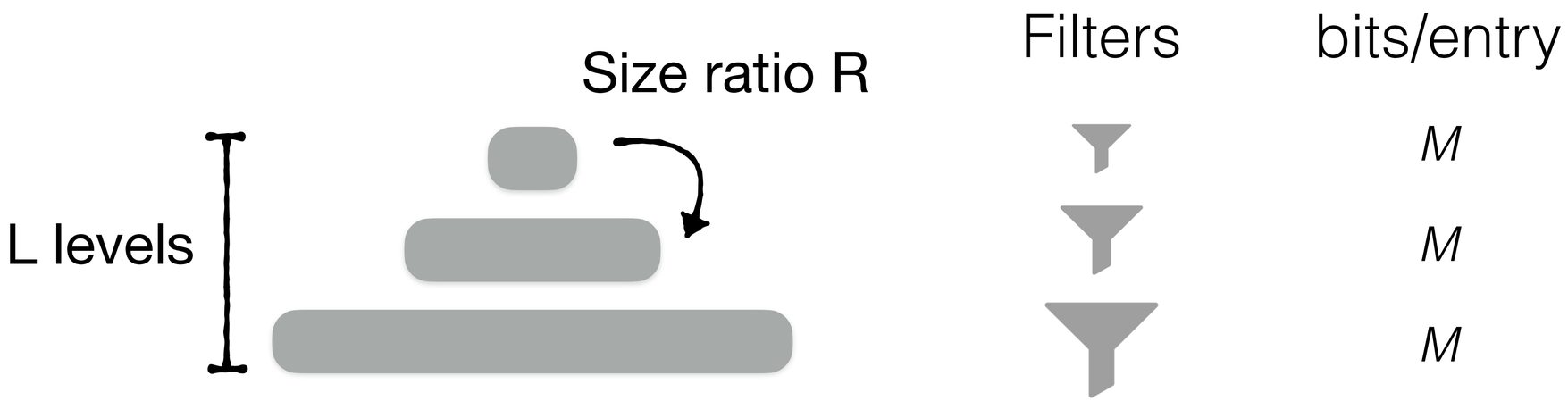
We have a leveled LSM-tree with a uniformly allocated Bloom filter with M bits/entry for each level. Our query workload consists of get requests to keys that usually exist at the largest level. Suppose we are near our memory capacity and should free some space used by the filters. Option 1 is to reduce the number of bits/entry for each filter. Option 2 is to drop the filter for the largest level. Compare and contrast these two approaches.



Option 2 would immediately reduce the filters' memory footprint by a factor of *R*, the size ratio. It would not impact query performance as the largest level's bloom filter only helps with queries to non-existing entries, of which there are none in our workload. This is the better choice.

Question 5

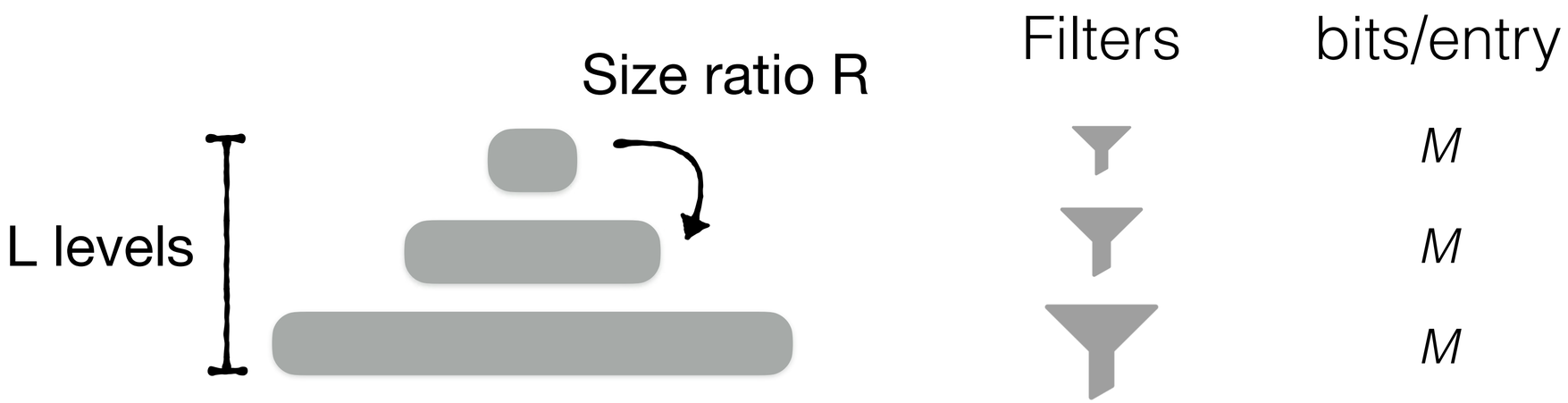
What is the cost of building Bloom filters for a leveled LSM-tree, measured in terms of amortized worst-case memory accesses per insertion? Assume the same number of bits per entry are assigned to filters at all levels.



Question 5

What is the cost of building Bloom filters for a leveled LSM-tree, measured in terms of amortized worst-case memory accesses per insertion? Assume the same number of bits per entry are assigned to filters at all levels.

An insertion into a bloom filter costs O(M) memory accesses.

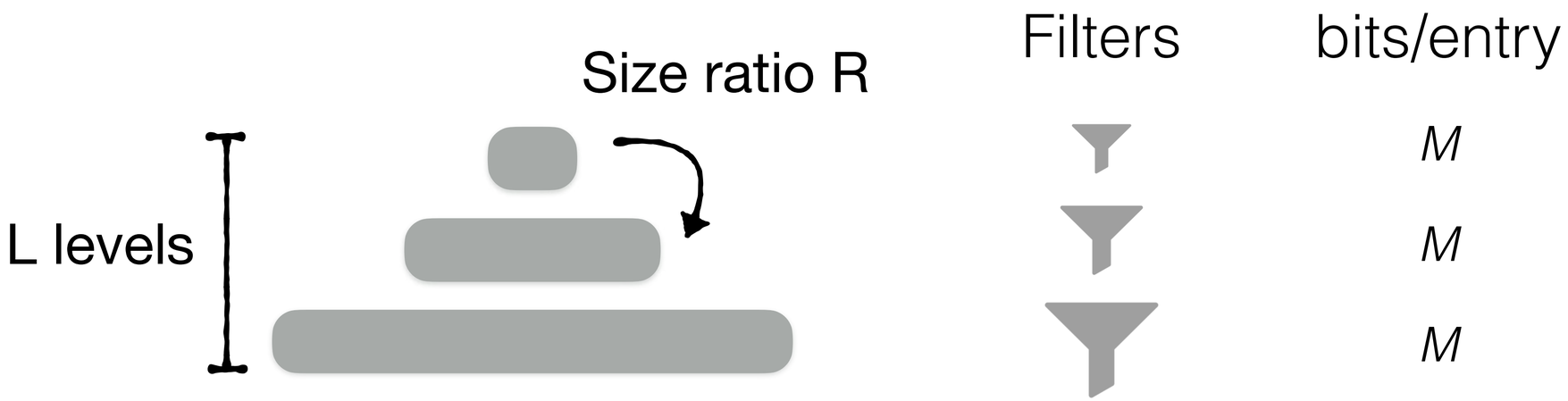


Question 5

What is the cost of building Bloom filters for a leveled LSM-tree, measured in terms of amortized worst-case memory accesses per insertion? Assume the same number of bits per entry are assigned to filters at all levels.

An insertion into a bloom filter costs O(M) memory accesses.

An entry is merged O(L * R) times across the tree. For each merge, the entry is inserted in a Bloom filter.



Question 5

What is the cost of building Bloom filters for a leveled LSM-tree, measured in terms of amortized worst-case memory accesses per insertion? Assume the same number of bits per entry are assigned to filters at all levels.

An insertion into a bloom filter costs O(M) memory accesses.

An entry is merged O(L * R) times across the tree. For each merge, the entry is inserted in a Bloom filter.

Hence, the overall cost is O(L * R * M).