Lecture 13: Hashing

Last few days: Collision Resolution

- Open addressing
- Separate overflow

Today: Updating Sorted Files

- Differential Files
- Bloom Filters (a novel application of hashing)

See handout on hashing.

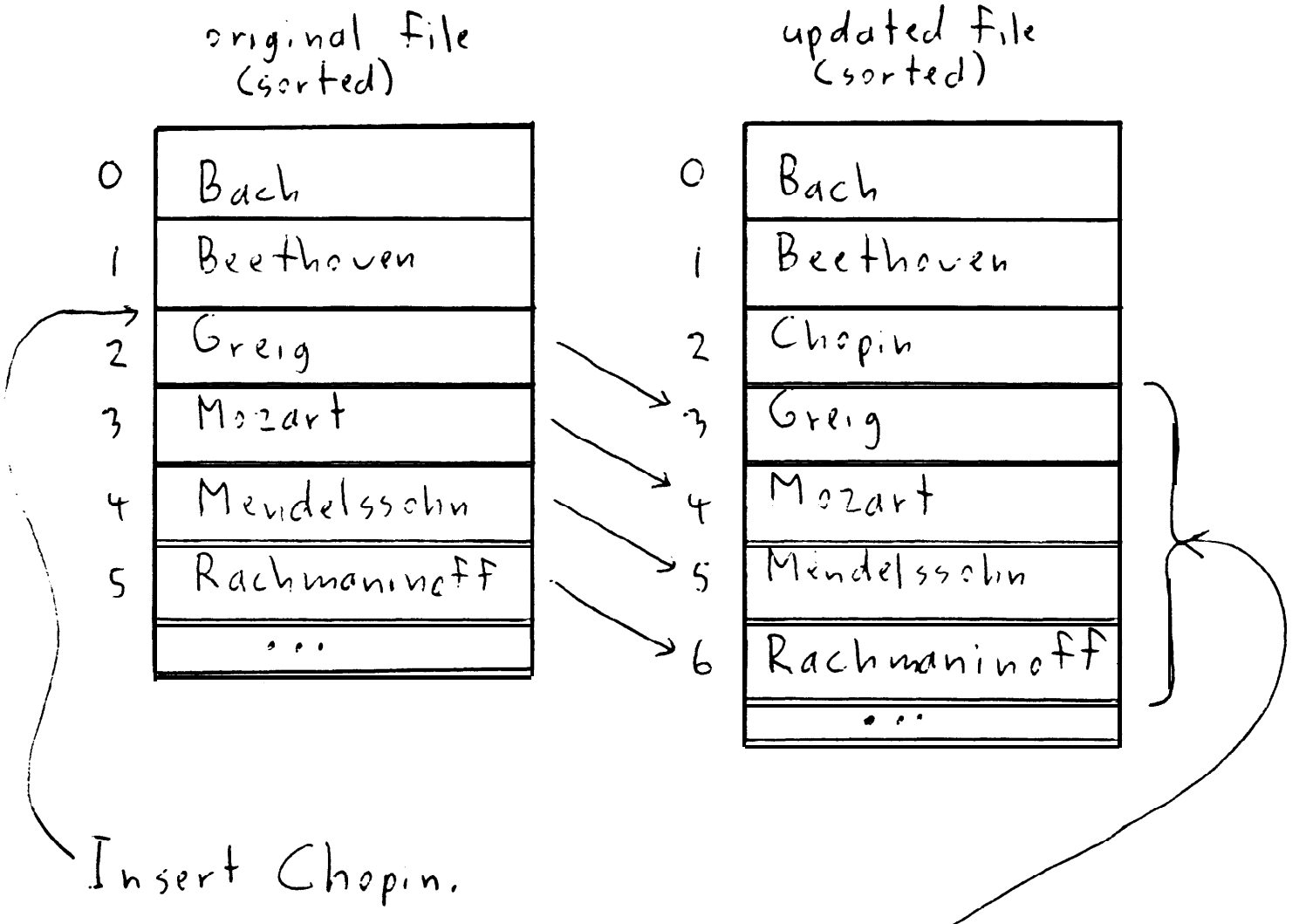
Introduction

Goal: To efficiently update a file while keeping it sorted.

Problem: Inserts and deletes to sorted files are expensive, since much of the file may have to be shifted.

Example

Insert a record into a sorted file.



Updating Sorted Files

Deletes: Physical deletion is expensive since much of the file must be shifted up by one position to fill in the gap created by the deleted record.

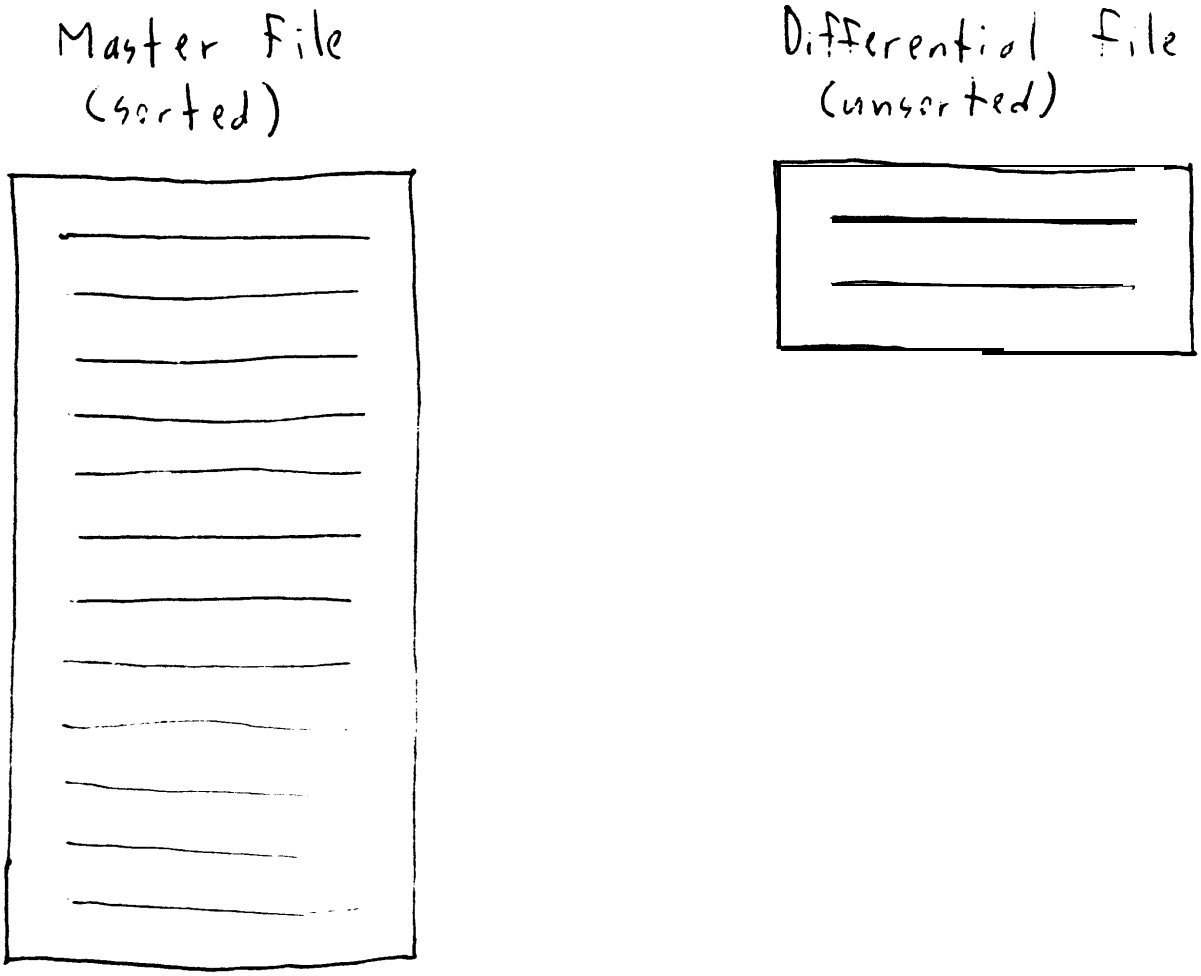
Instead, we can "mark" a record for deletion by putting a D in a special field, instead of physically deleting it.

Modifications: These are easy.

eg. we can modify Greig's birth date or salary without shifting any records.

Inserts: These are a problem.

Solution: Differential Files



Records to be inserted into the master file are appended to the differential file instead.

Retrieval

- First, search the differential file (sequentially).
  If the record is found, then we are done.
  If the record is not found, then ...

- Search the Master file (possibly using binary search).
  If the record is not found, then it was never inserted into the file.

Note: Searching the diff'l file requires disk accesses

- Also, most records are not in the diff'l file.

- So, we do not want to search the diff'l file unless we have good reason to think the record is there

Solution: Bloom Filters

- Bloom Filtering is a hashing-based technique to avoid unnecessary searches of a differential file.

- It is an (imperfect) test for the presence of a key in a diff'l file.

- The test outcome is either:

  No, the record is not in the file,

  OR

  Maybe the record is in the file.

Bloom Filters: Inserting records

- Keep an array, $B(n)$, of N bits in main memory
- Use a number of hash functions,

$$h_1, h_2, \cdots, h_t$$

where $0 \leq h_i(key) \leq N-1$ for each i.

- When inserting a record with key K,

First set bits $h_1(K), \cdots, h_t(K)$ of the array to 1.

Then insert the record into the diff'l file.

Bloom Filters: Retrieving records

To retrieve a record with key K,

- First test bits $h_1(k), \ldots, h_t(k)$ of the array

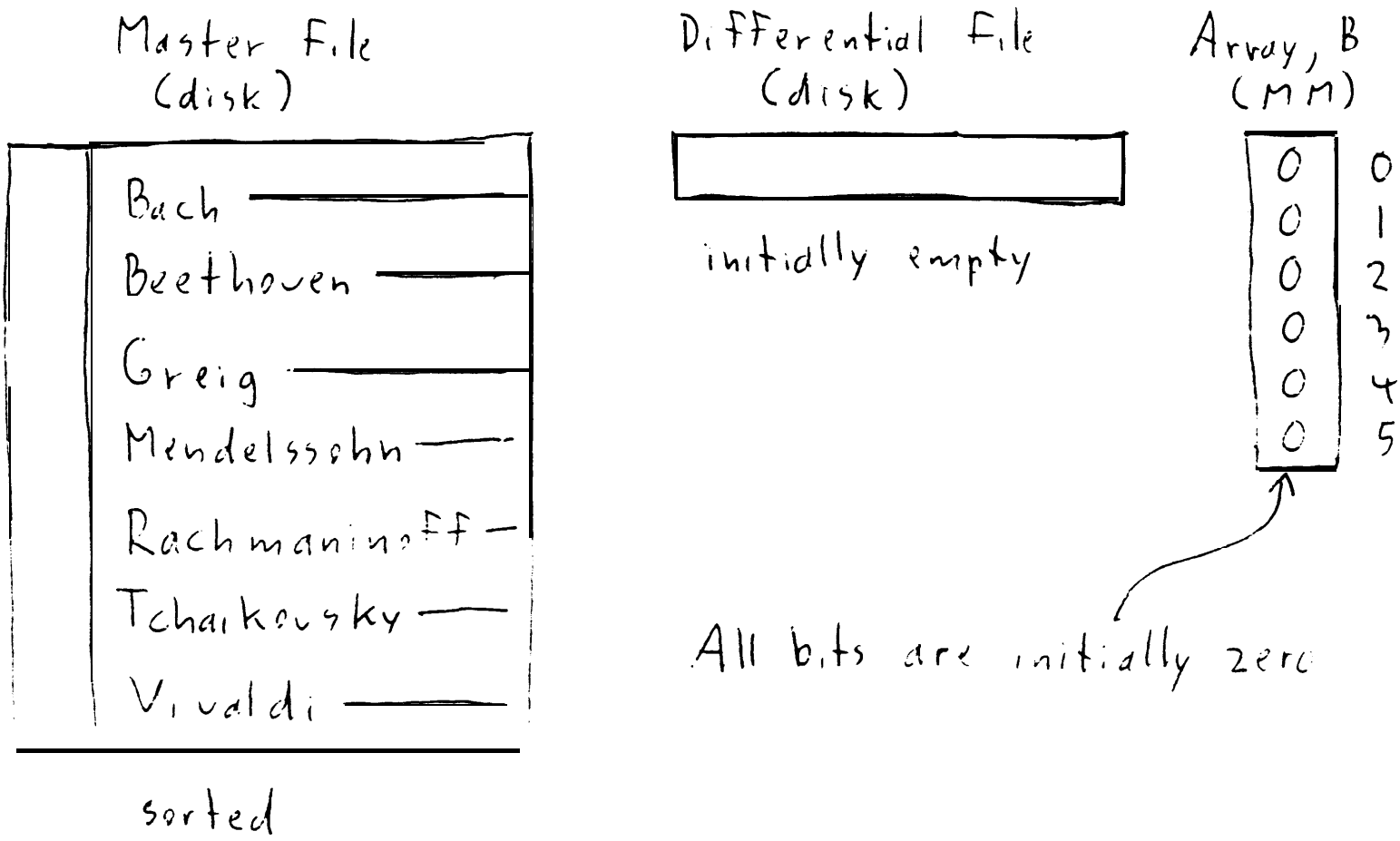
There are two cases:

(a) If any of these bits is 0, then the record is <u>not</u> in the differential file. So, do <u>not</u> search the differential file, but search the master file instead.

(b) If all these bits are 1, then the record <u>may</u> be in the diff'l file, so we must search it.

If the record <u>is</u> found in the diff'l file, then we are done; <u>otherwise</u> we must also search the master file

Bloom Filters : Example



key	h_1(key)	h_2(key)
Bach	0	3
Chopin	3	5
Mendelssohn	1	5
Mozart	1	3
Rachmaninoff	2	4

Bloom Filters: Example (cont.)

Insert

Mozart

Chopin

Retrieve	In Diff file?	Search Diff File?
Mozart	maybe	yes
Bach	no	no
Mendelssohn	maybe	yes

Delete	In Diff File?	Search Diff. File?
Rachmaninoff	no	no
Mendelssohn	maybe	yes
Chopin	maybe	yes

Bloom Filters: Example (cont.)

Master File (disk)

	Buch
	Beethoven
	Greig
D	Mendelssohn
D	Rachmaninoff
	Tchaikovsky
	Vivaldi

sorted

Differential File (disk)

	Mozart
D	Chopin

unsorted

Array, B (MM)

0	0
1	1
0	2
1	3
0	4
1	5

New File and array contents

Merging

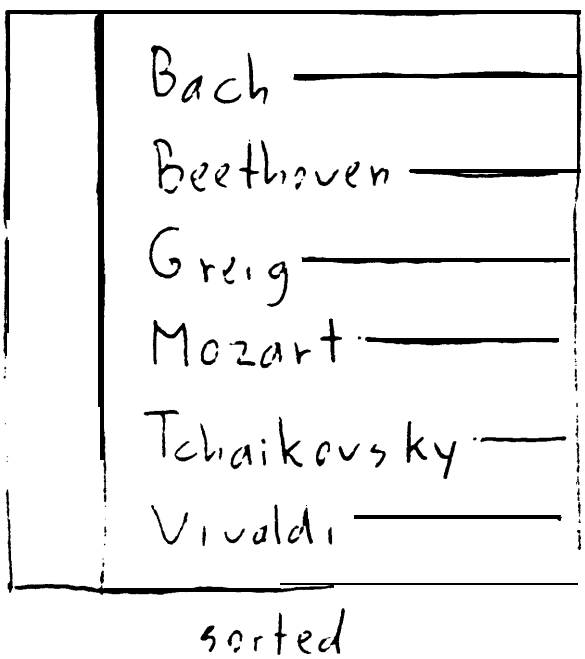
- When most of the bits in the array, B, are 1, then the Bloom Filter is useless since you will have to search the differential file most of the time.

- At this point, the differential file is merged with the master file, to produce a new, sorted master file, in which records marked D are physically deleted.

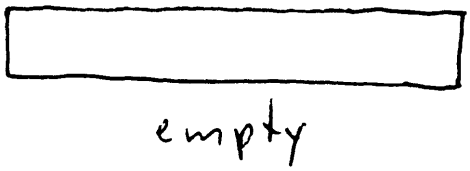
- Also, a new, empty differential file is created, and all array bits are set to 0.

Bloom Filters: Example (cont.)

New Master File (disk)



New Diff'l File (disk)



Array B (MM)

0	0
0	1
0	2
0	3
0	4
0	5

Final File and array contents, after merging