Lecture 11: Direct Files

Last Day: Collision Resolution

- Open Addressing
- Linear Probing ...
- Clusters
- Buckets

Today: Table Assisted Hashing

- Simplified version
- Full version
- Insertions & retrievals
- Examples

See handout on hashing

Table Assisted Hashing

- A variation of open addressing, in which we generate a sequence of probe addresses, PA_0, PA_1, PA_2 ...

- But, when searching for a record, we now use a <u>table in main memory</u> to determine at what PA_i the record is stored.

- The table provides information on what records are stored at address PA_i.

- Main Advantage: All retrievals can be done with one File access.

- Inserts + deletes can take longer, but in many applications, retrievals are more Frequent than updates.

- Simplified Version: Without buckets.

  - The table in main memory (MM) records the key of each record in the file.

  - Probing for an available address is now done in MM using the table.

  - Note: the table has only one field, so it is much smaller than the File, so it may fit in MM.

Example: Simplified Version

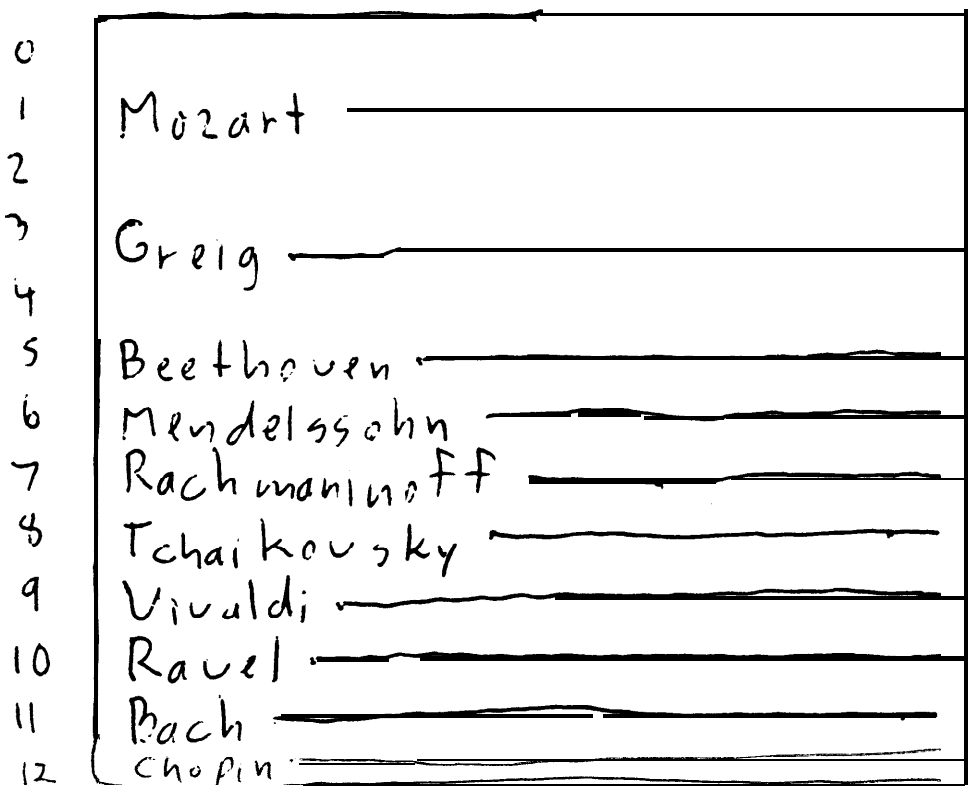
key	hash(key)
Mozart	1
Tchaikovsky	8
Ravel	10
Beethoven	5
Mendelssohn	5
Bach	10
Greig	3
Rachmaninoff	5
Vivaldi	6
Chopin	6

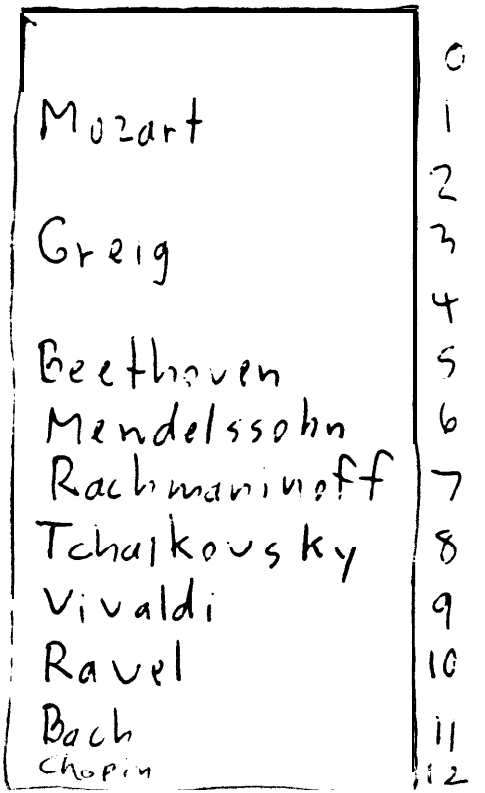
Linear Probing

Step = 1

ie, PA_i =

(hash + i) mod 12





Example Retrieval

Retrieve Chopin (hash value = 6)

① Get entry 6 from table. (Not Chopin)

② Get entry 7 from table. (Not Chopin)

③ Get entry 8 from table. (Not Chopin)

④ Get entry 9 from table (Not Chopin)

⑤ Get entry 10 from table (Not Chopin)

⑥ Get entry 11 from table (Not Chopin)

⑦ Get entry 12 from table. (Chopin!)

Get record 12 from File.

Cost = 1 file access.

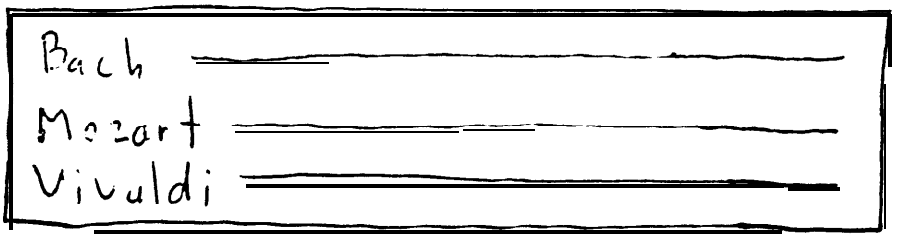
Note: Without the table in MM, the cost would be 7 file accesses.

Table Assisted Hashing: Real Version

- The table has one entry for each bucket in the file.

- Each table entry stores the largest key in its bucket. (Thus, the table is called the Max Key Table.)

eg.



Bucket

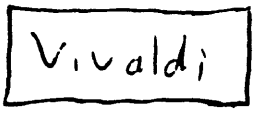


Table Entry

- Alphabetically, Bach < Mozart < Vivaldi

- Note: With buckets, the table is smaller, and more likely to fit in MM.

- With buckets, searches are still simple. (as we shall see), but insertions are more complex.

- First, given a key and a hash function, we generate a sequence of probe addresses, PA_0, PA_1, PA_2, ...

Example

$$PA_i = (hash + i) \bmod 5$$

i.e. Linear probing with step = 1

and File size = 5 buckets

key	hash(key)
Ravel	2
Vivaldi	2
Mozart	2
Mendelssohn	4
Tchaikovsky	4
Greig	2
Beethoven	a
Bach	0
Eisner	2

bucket size = 2 records

Example (cont)

hash File (disk)

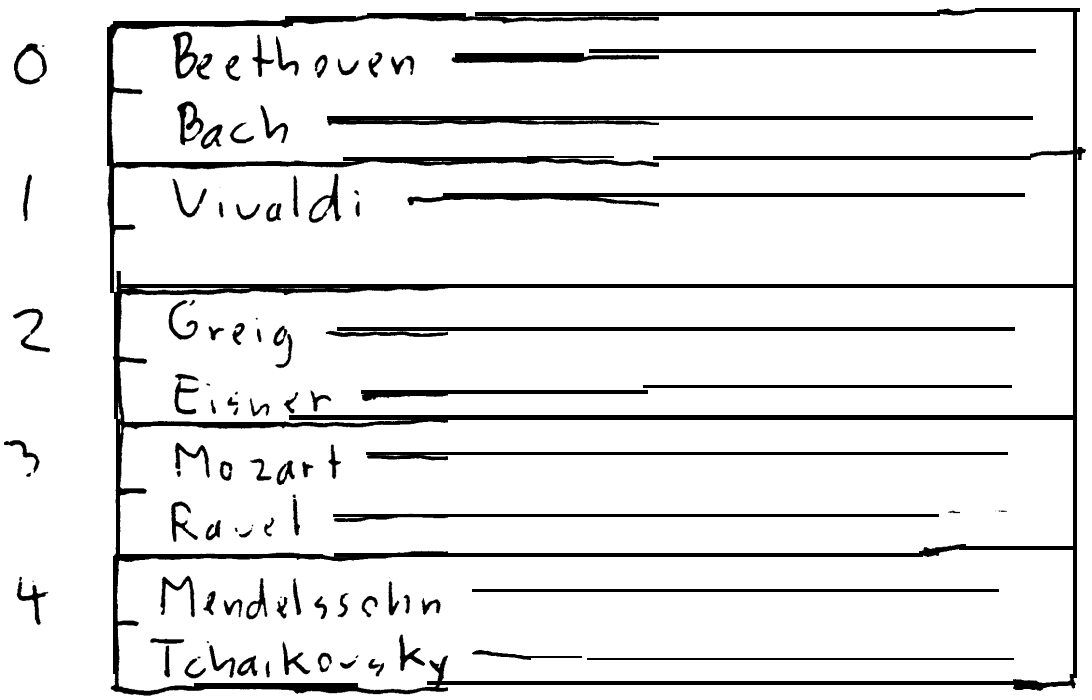
0	Beethoven
	Bach
1	Vivaldi
2	~~Ravel~~ Greig
	~~Vivaldi~~ ~~Mozart~~ Eisner
3	~~Vivaldi~~ Mozart
	Ravel
4	Mendelssohn
	Tchaikovsky

Max Key Table (MM)

0	Beethoven
1	Vivaldi
2	~~Ravel~~ ~~Vivaldi~~ ~~Ravel~~ ~~Mozart~~ Greig
3	~~Vivaldi~~ Ravel
4	~~Mendelssohn~~ Tchaikovsky

Example (cont.)

hash File (disk)



entire records (keys + data)

Final File Contents

Max key table (MM)

keys only

Insertion Algorithm

Insert record R with key K.

- Look in buckets in order of probe sequence.
- If room in current bucket, then insert the record, update max key table, done.
- If no room, then:
  - case (a): K is greater than all keys presently in the bucket.

    ∴ R does not belong to this bucket.

    Try next bucket in probe sequence.
  - case (b): There is a record in the bucket with key greater than K.

    ∴ R belongs to this bucket.

    Purge the record with maximum key, insert R, update max key table.

    Reinsert purged record into the file

Retrievals

- They are cheap: 1 file access each.

- Given a key, k, search the table for the bucket in which key k is stored.

- ie, Find the smallest i such that

$$k \leq \text{maxkey}[PA_i(k)]$$

- Finally read bucket i from the file, and extract the record with key k.

Example 1

Retrieve Mozart

- $PA_0(\text{Mozart}) = 2$

  maxkey(2) = Greig < Mozart

  ∴ Mozart is not in bucket 2.

- $PA_1(\text{Mozart}) = 3$

  maxkey(3) = Ravel ≥ Mozart

  ∴ If Mozart is in the file

  then he must be in bucket 3

- Read bucket 3 from the file, and search it for Mozart.

- Cost = 1 file access

Example 2

Retrieve Vivaldi:

- PA_0(Vivaldi) = 2

  max key (2) = Greig < Vivaldi

- PA_1(Vivaldi) = 3

  max key (3) = Ravel < Vivaldi

- PA_2(Vivaldi) = 4

  max key (4) = Tchaikovsky < Vivaldi

- PA_3(Vivaldi) = 0

  max key (0) = Beethoven < Vivaldi

- PA_4(Vivaldi) = 1

  max key (1) = Vivaldi

- Retrieve bucket 1 from the file, and search it for Vivaldi.
- Cost = 1 file access

Question: Can the Max Key table fit in MM?

Example: Suppose that

- each key is 8 bytes,
- each record is 1000 bytes,
- each bucket holds 8 records,
- the file has 100,000 buckets.

∴ File can hold 800,000 records.

∴ File needs 800,000,000 bytes ≈ 800 MB

∴ Too big for main memory.

∴ File must be stored on disk.

But, table size is:

100,000 buckets × 8 bytes/bucket

= 800,000 bytes = 800 KB = 0.8 MB

Table can easily fit in most main memories.