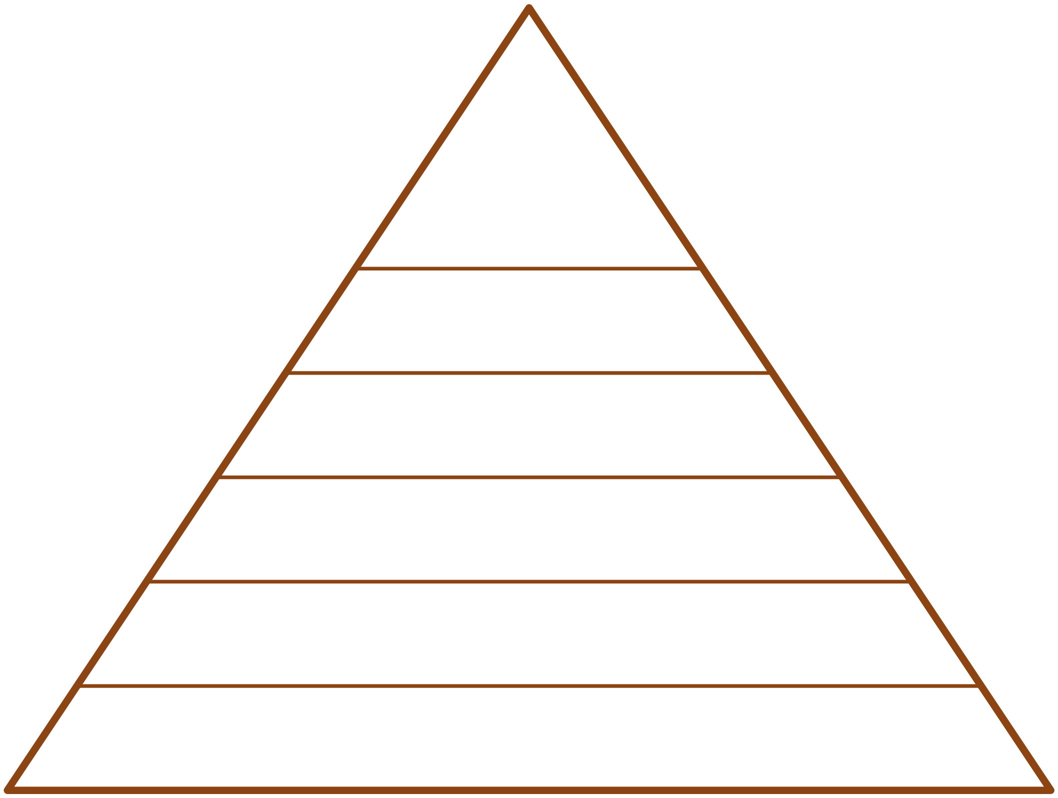
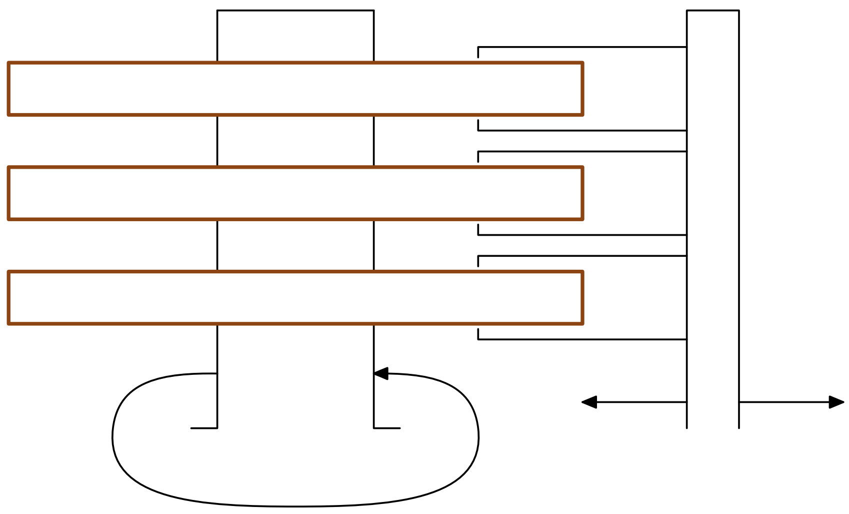
Topic 3:

Files and Indexing

The Storage Pyramid

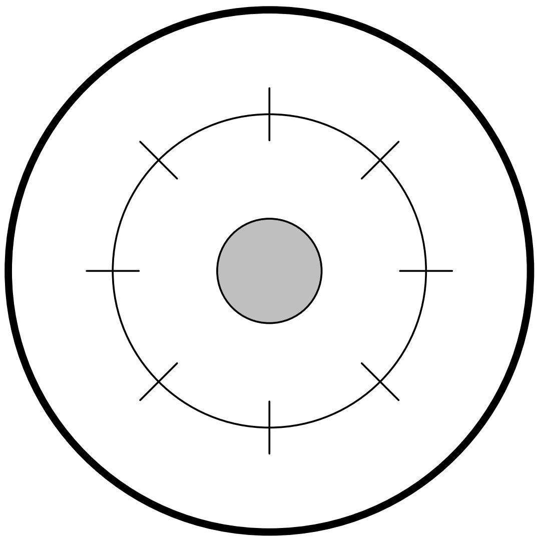


Hard Drive Physical Characteristics (1 / 2)



Side View

Hard Drive Physical Characteristics (2 / 2)



Top View

Sources of Read / Write Delay

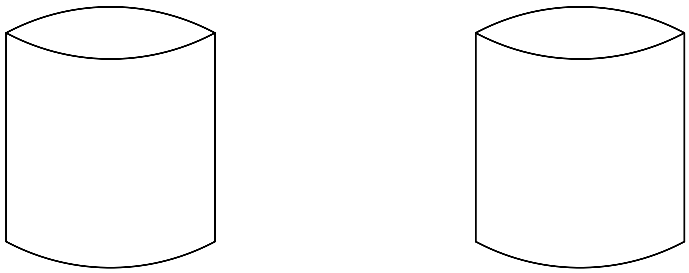
The three major sources of delay (in descending order):

Western Digital 3.5” Hard Drive Specs

	WD450AA (9/2000)	WD1002FAEX (7/2012)	DC HC550 (8/2020)	DC HC690 (5/2025)
Size (GB)	45	1,000	18,000	32,000
Platters & Heads	3 & 6	3 & 6	9 & 18	11 & 22
Bytes per Sector	512	512	512 / 4096	512 / 4096
Sectors per Surface	14,655,144	325,587,528	1,953,147,562	2,840,973,684
Rotations (RPM)	5400	7200	7200	7200
R/W Seeks (ms)	9.5 / 13.4	?? / ??	?? / ??	?? / ??
Latency (ms)	5.4	4.2	4.16	4.16
Cache (MB)	2	64	512	512
Xfer Rate (MiB/s max.)	66.6	126.0	257.0	257.0
Read / Write (W)	6.2	6.8	6.5	9.4
[Power] Idle (W)	6.2	6.1	5.6	5.5
Standby / Sleep (W)	∼1.1	0.7	?.?	?.?

RAID Background (1 / 3): Disk Mirroring

(a) Disk Mirroring



Advantage(s):

Disadvantage(s):

RAID Background (2 / 3): Disk Striping

(b) Disk Striping

Example(s):

Advantage(s):

Disadvantage(s):

RAID Background (3 / 3): Parity Bits

(c) Parity Schemes

Example(s):

Advantage(s):

Disadvantage(s):

Detour: Independent Event Probabilities (1 / 3)

First, some set and probability review!

1. DeMorgan's Laws for Sets:

2. For a sample space S and an event $E \in S$,

   the probability of E's occurrence is:

3. $\sum_{e \in S} p(e) =$

Detour: Independent Event Probabilities (2 / 3)

Next, Independent Events:

4. Events A and B are *independent* when . . .

5. Recall: Principle of Inclusion/Exclusion for 2 Sets is:

   Applied to probabilities:

Detour: Independent Event Probabilities (3 / 3)

Probabilities for Independent Events (cont.):

Recall:

4. $p(A \cap B) = p(A) \cdot p(B)$
5. $p(A \cup B) = p(A) + p(B) - p(A \cap B)$

6. Combining (4) and (5):

7. And thanks to DeMorgan's Laws and (4):

Probability of Hard Disk Drive Failures (1 / 4)

Factors contributing to HDD failures include:

How often does a 'young' (1-3 years old) HDD fail?

Refs: `http://research.google.com/archive/disk_failures.pdf`
`https://www.backblaze.com/blog/best-hard-drive/`

Probability of Hard Disk Drive Failures (2 / 4)

What is the p_f for a striped 2-disk system?

$\Rightarrow$ Remember, the system fails when either drive fails!

(Let D$\#_f$ be the event of Disk # failing.)

$$\begin{aligned} p_f &= p(\text{D1}_f \cup \text{D2}_f) && \text{Either or both!} \\ &= p(\text{D1}_f) + p(\text{D2}_f) - p(\text{D1}_f) \cdot p(\text{D2}_f) && \text{Princ. Inc./Ex. \&} \\ &= 0.02 + 0.02 - (0.02)^2 && \text{\ldots Indep. events} \\ &= 0.0396 \text{ (3.96\%)} \end{aligned}$$

Probability of Hard Disk Drive Failures (3 / 4)

New point of view: Be an optimist!

The probability that Disk D# does <u>not</u> fail:

$$p(\text{D\#}_{nf}) = 1 - p(\text{D\#}_f) = 1 - 0.02 = 0.98$$

What is the p_{nf} for a striped 2-disk system?

$p_{nf} = p(\overline{\text{D1}_f \cup \text{D2}_f})$	[Neither fails!]
$= p(\overline{\text{D1}_f} \cap \overline{\text{D2}_f})$	[De Morgan's]
$= p(\text{D1}_{nf} \cap \text{D2}_{nf})$	[$\overline{\text{D\#}_f} = \text{D\#}_{nf}$]
$= p(\text{D1}_{nf}) \cdot p(\text{D2}_{nf})$	[Independent events assumed]
$= p(\text{D\#}_{nf})^2$	[Foreshadowing ...]
$= (0.98)^2$	[From above]
$= 0.9604$ (96.04%)	[= $1 - 0.0396$]

Probability of Hard Disk Drive Failures (4 / 4)

What if we have *dozens* of HDDs? Say, three dozen?

No problem; being optimistic scales nicely!

$p_{nf} = p(\overline{\text{D1}_f \cup \ldots \cup \text{D36}_f})$	[None fail!]
$= p(\overline{\text{D1}_f} \cap \ldots \cap \overline{\text{D36}_f})$	[Massive De Morgan's]
$= p(\text{D1}_{nf} \cap \ldots \cap \text{D36}_{nf})$	[$\overline{\text{D\#}_f} = \text{D\#}_{nf}$]
$= p(\text{D1}_{nf}) \cdot \ldots \cdot p(\text{D36}_{nf})$	[Independent events assumed]
$= (0.98)^{36}$	[From last slide]
$= 0.4832\ldots$ (48.32%)	[$p_f = 1 - p_{nf} = 0.5168$]

Remember: Assuming independence is convenient, not realistic!

RAID: Redundant Arrays of Independent* Disks (1 / 2)

* *Originally "Inexpensive"*

Level 0: Striped Volume (N data disks)

Level 1: Mirrored (N data disks + N mirror disks)

RAID: Redundant Arrays of Independent Disks (2 / 2)

Level 5: Block–Interleaved Distributed Parity (N+1 disks)

Level 6: "Double Parity"

SSDs: Solid–State Device (Flash) Storage

- NAND–based non–volatile RAM
- Not a new idea: Used to have “RAM drives”

  (Even though the 1981 IBM PC had 256 KB RAM – max!)

Advantage(s):

Disadvantage(s):

File Granularity Hierarchy

Bits

⇓

Bytes

⇓

Fields

⇓

Records

⇓

Blocks

⇓

Files

⇓

Databases

File Blocking (1 / 2)

Definition: Blocking Factor (bf)

Definition: Internal Fragmentation

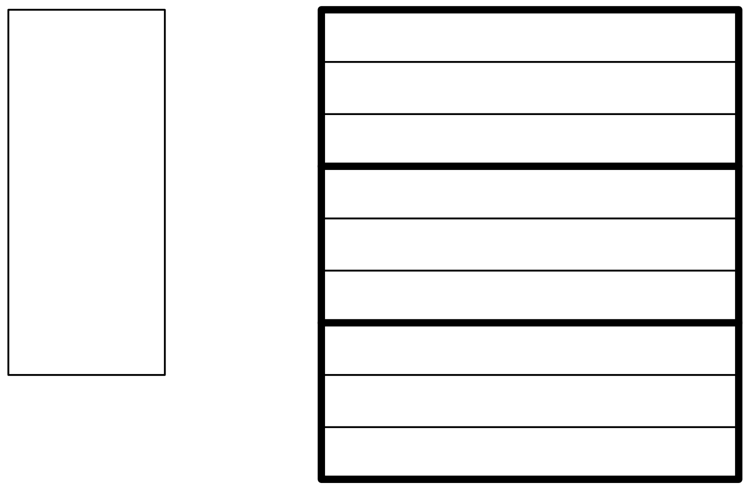
Example(s):

File Blocking (2 / 2)

Locating records within blocks:

Indexing

Definition: Index



A Few Words about Keys

Some of the types of keys:

One Classification of Indices

Primary Index (1 / 2)

Characteristics:

- The indexed field is ______________________ .
- The index records are _____________ on the key.
- The DB file records are _____________ on the key.

Primary Index (2 / 2)

Example(s):

University	RID
Eau Claire	→
Green Bay	→
La Crosse	→
Madison	→
Milwaukee	→
Oshkosh	→
Parkside	→
Platteville	→
River Falls	→
Stevens Point	→
Stout	→
Superior	→
Whitewater	→

PRIMARY INDEX

University	**Founded**
Eau Claire	1916
Green Bay	1968
La Crosse	1909
Madison	1848
Milwaukee	1885
Oshkosh	1871
Parkside	1968
Platteville	1866
River Falls	1874
Stevens Point	1894
Stout	1891
Superior	1893
Whitewater	1868

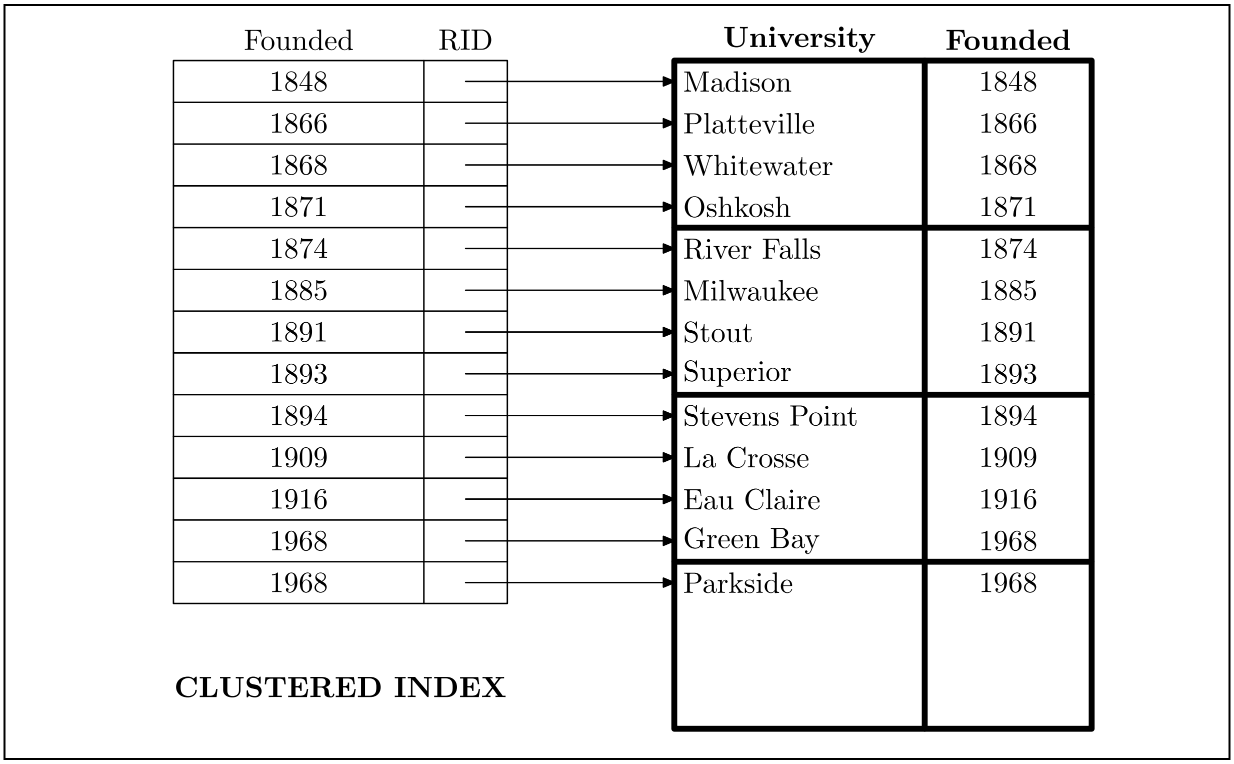
Clustered Index (1 / 2)

Characteristics:

- The indexed field is ____________________ .
- The index records are ____________ on the key.
- The DB file records are ____________ on the key.

Clustered Index (2 / 2)

Example(s):



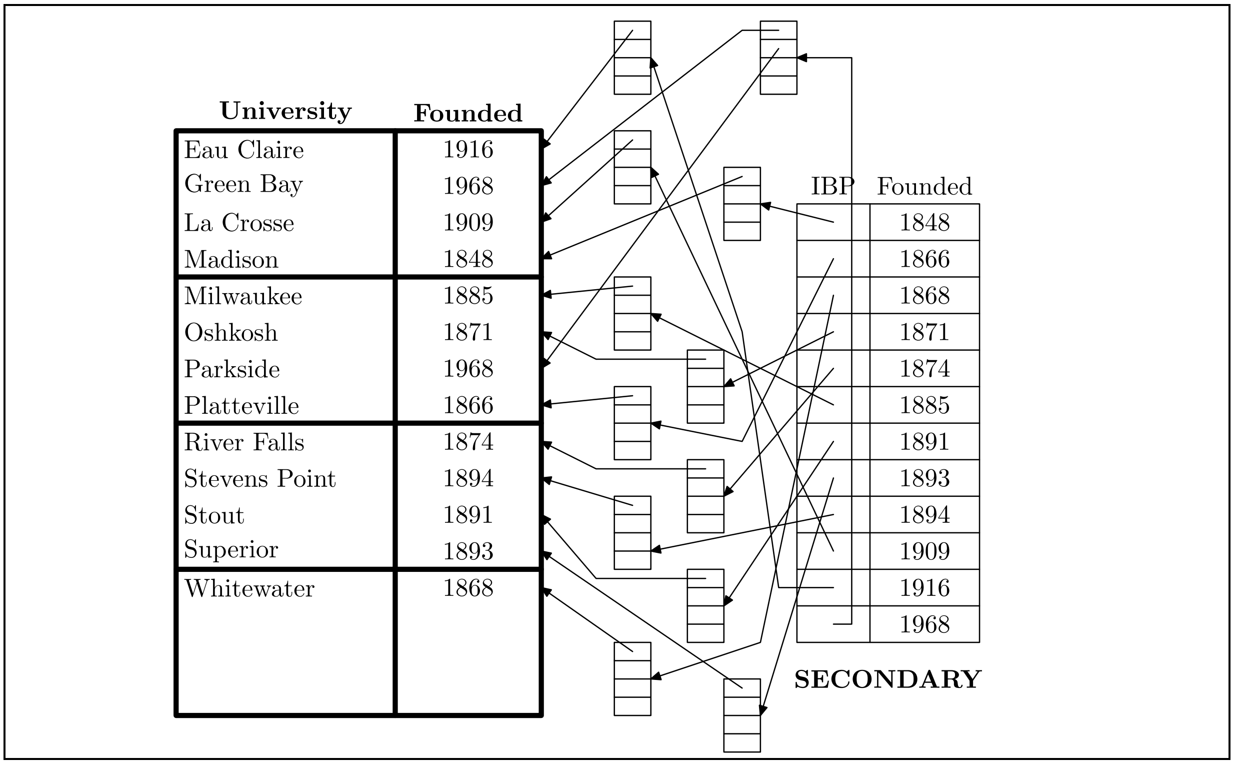
Secondary Index (1 / 2)

Characteristics:

- The indexed field is ______________________ .
- The index records are _____________ on the key.
- The DB file records are _____________ on the key.

Secondary Index (2 / 2)

Example(s):



Another Index Categorization: Dense vs. Sparse (1 / 2)

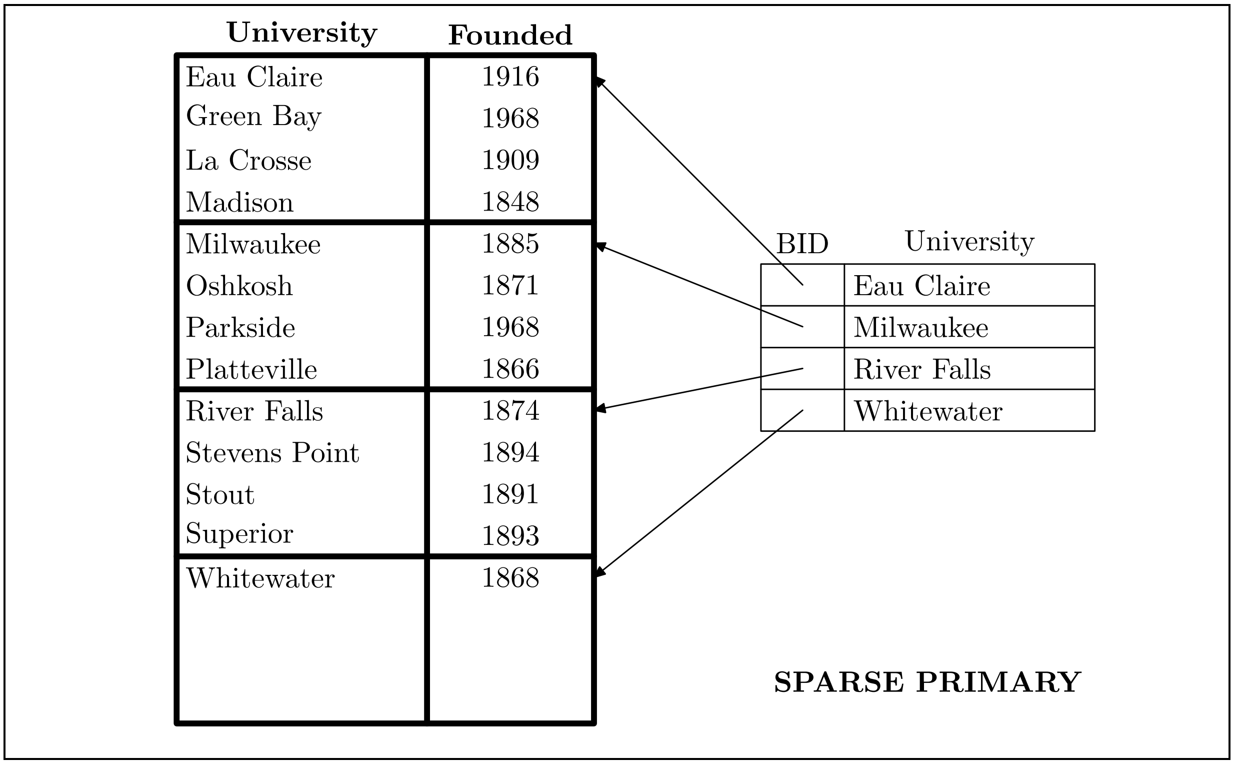
Dense Indices:

Sparse Indices:

Notes:

Another Index Categorization: Dense vs. Sparse (2 / 2)

Example(s):



Review of Internal Hashing

- Goal: $O(1)$ search performance
- Key $\longrightarrow$ Hash Coding $\longrightarrow$ Compression Mapping $\longrightarrow$ Hash Table Index
- Collision Resolution: Chaining v. Open Addressing
- Problem:

Dynamic Hashing

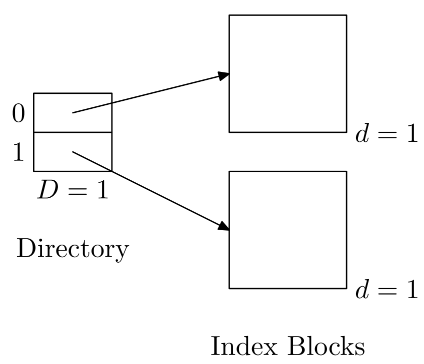
Two components:

Example(s): Let bf=3. Insert 1101, 1000, 0101, 0010, 1110, and 1010:

Extendible Hashing: Basics

Improvement over Dynamic Hashing:

Directory is an array $\Rightarrow$



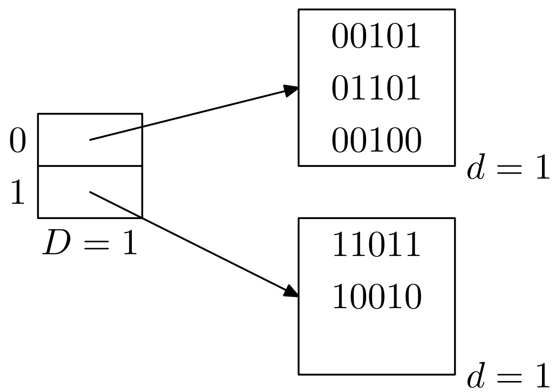
Extendible Hashing: Insertion (1 / 2)

When a key is inserted into a full index block:

- The block becomes k blocks
- The depth of each is one more than the original's
- Existing content is distributed to the new blocks
- If any $d > D$, split ('double') the directory:
  - increase global depth by one
  - create new directory of k^D pointers
  - copy existing block pointers
  - add pointers to new blocks

Extendible Hashing: Insertion (2 / 2)

After Inserting 11011, 00101, 01101, 10010, and 00100:



(Assume max. 3 keys/node)

After Inserting 01110:

Extendible Hashing: Deletion

Question: Do you have lots of disk space available?

If so:

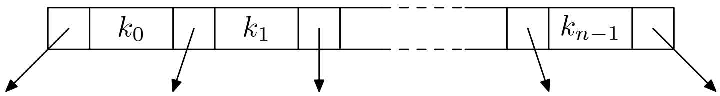
If not:

B–Trees: Structure

But first: Know that "B" does <u>not</u> stand for "binary"!

"Bayer"? (Rudolf Bayer & Edward McCreight, '72)
"Balanced"? (It is!) "Boeing"? (McCreight's employer?)

Structure of a B–Tree node:



— A node holding n keys holds $n+1$ pointers

— Each key is stored in the index exactly once ($\because$ dense)

— A node's keys are stored in (ascending) sorted order

— Pointer 0's subtree has all keys $<$ key k_0

— Pointer i's subtree has all keys $> k_{i-1}$ and $< k_i$

— Pointer n's subtree has all keys $> k_{n-1}$

B-Trees: Definition

Definition: B-Tree of Order M (*a la* D. Comer[♮])

. . . .

. . . .

. . . .

. . . .

. . . .

♮ Comer, D. “The Ubiquitous B–Tree,” ACM Computing Surveys 11(2), June 1979, pp. 121-137.

B-Tree: Insertion (1 / 2)

- Find the leaf node that should contain the new key value
- If leaf has capacity, insert the key into it.

  Otherwise:
  - Form a set of the leaf’s keys plus the insertion key
  - Promote the set’s median value to the parent
  - Create two nodes to hold the key values that are $<$ and $>$ the median, respectively.
  - Attach nodes as children on either side of the median

B-Tree: Insertion (2 / 2)

Example: Insert 40, 20, 60, 10, 80, 5, 15, and 25

into a B-Tree of Order 2:

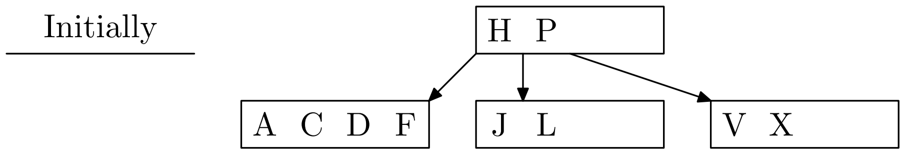
B-Tree: Deletion (1 / 2)

When a deletion leaves a node under-full:

- If the under-full node is a leaf node:
  - If a neighboring sibling has above-minimum occupancy, borrow:
    - Move separating value from parent to under-full node
    - Move appropriate value (smallest / largest) from neighbor to parent
  - Otherwise, concatenate:
    - Merge node, a neighboring sibling, and the parent's separating value into one node
    - (Note that this can leave the parent under-full, so recurse!)
- Otherwise, the under-full node is an internal node:
  - Replace deleted key with its inorder predecessor or successor
  - Recurse if necessary

B-Tree: Deletion (2 / 2)

Example(s): Still assuming M = 2



B-Tree: Capacity

What is the key capacity of a B-Tree of Order M?

Example(s):

B-Tree: Order Determination

The Idea: Select order to best fit disk block capacity

Remember: A node of a B-Tree of Order M can hold

$2M$ keys and $2M + 1$ pointers

Example(s):

B^{+}–Tree: A B-Tree for Indexing

Like a B-Tree, but:

B$^+$–Tree: Insertion

Example(s):

B$^+$–Tree: Advantages and Disadvantages over B-Trees

Advantage(s):

Disadvantage(s):