The author gratefully acknowledges support from the taxpayers of Québec and of Canada who have paid his salary and research grants while this work was developed at McGill University, and from his students (who built the implementations and investigated the data structures and algorithms) and their funding agencies.

Tries: a Data Structure for Secondary Storage

T. H. Merrett
with
Jack Orenstein
Heping Shang
Xiaoyan Zhao
McGill University

Overview

Trie representations
Compression
Spatial Data
Variable Resolution
Text Data
Dynamic Tries & Concurrency
Experimental Results

website `www.cs.mcgill.ca/~tim/tries/tries.html`

00000011

00101100

10000000

10000101

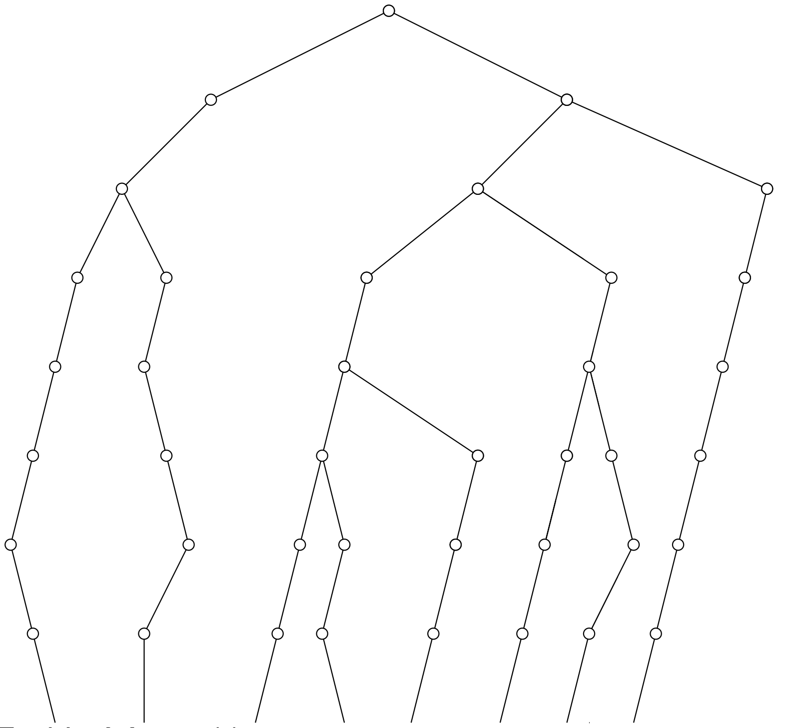
10001000

10100000

10101100

11000000

Eight Records and Their Trie

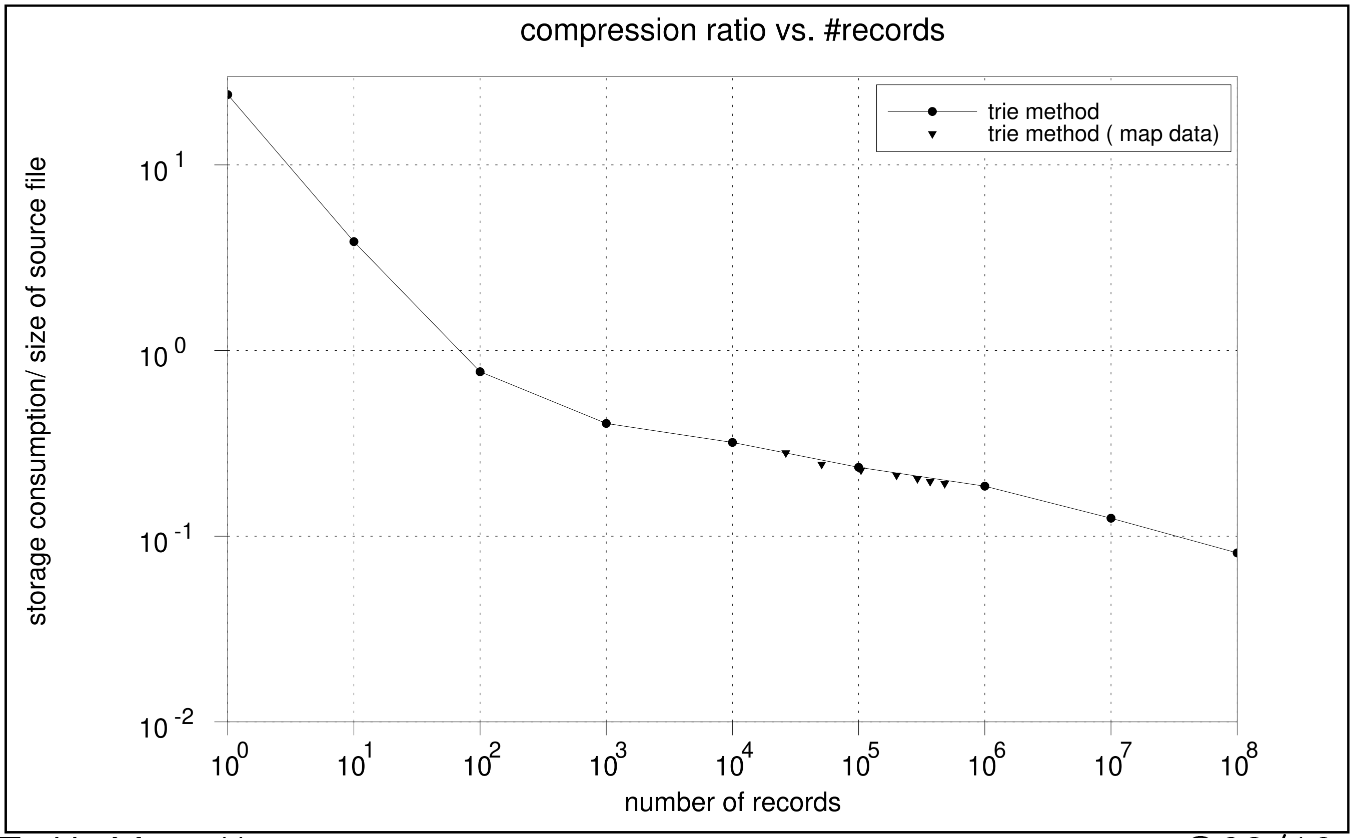


Pointerless Representation

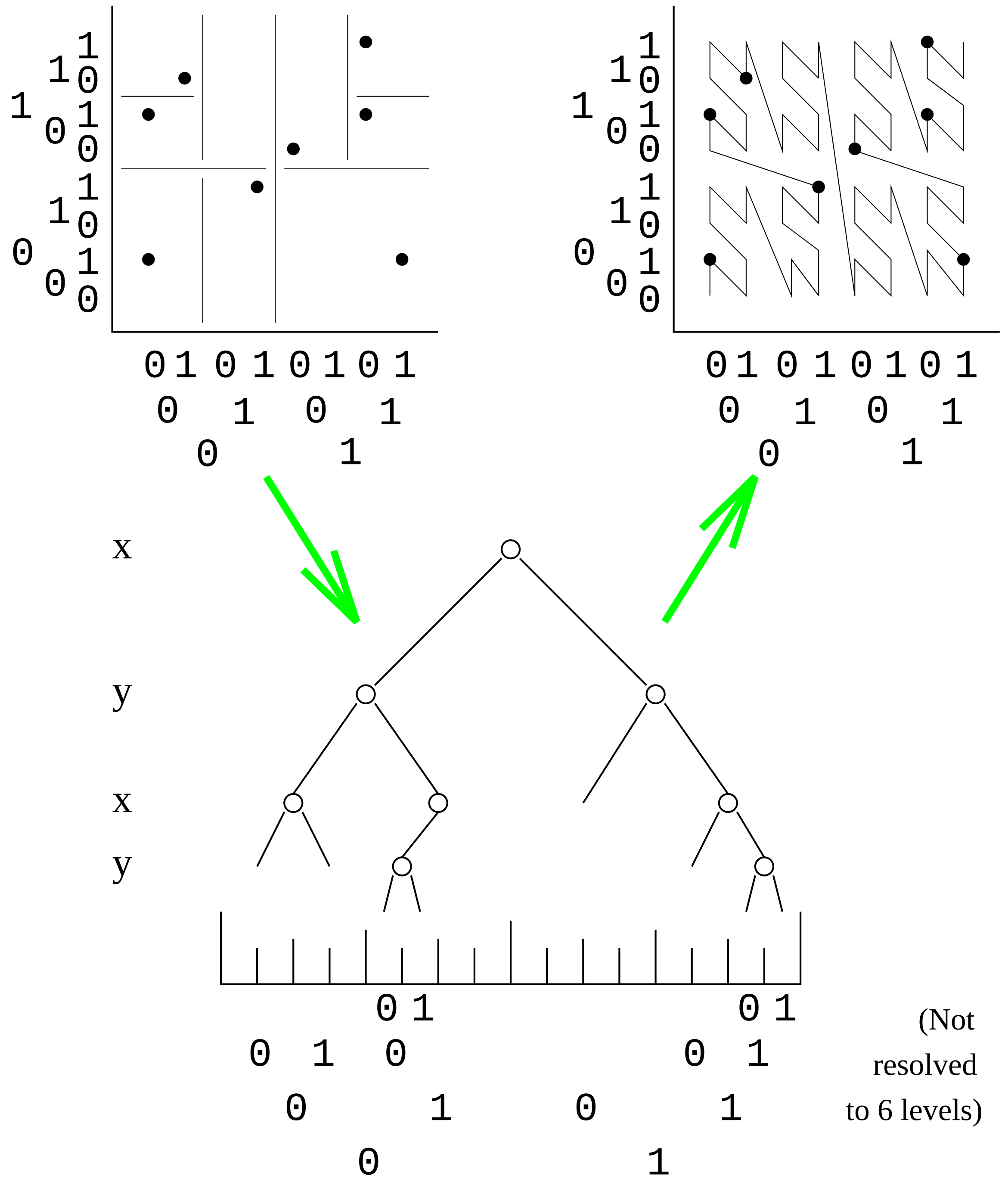
11							
10		11					
11		11					10
10	10	10			10		10
10	01	11			11		10
10	01	11		10	10	01	10
01	10	10	10	10	10	10	10
01	10	10	01	10	10	10	10

Compression

2^h records of h bits each	$h2^h$ bits
Complete binary tree, height h,	
2 bits/node	$2(2^h - 1)$ bits
Size ratio, trie:source	$\approx \frac{2}{h}$
Trie compression	$1 - \frac{2}{h}$



kd-Tries (2d-Trie)



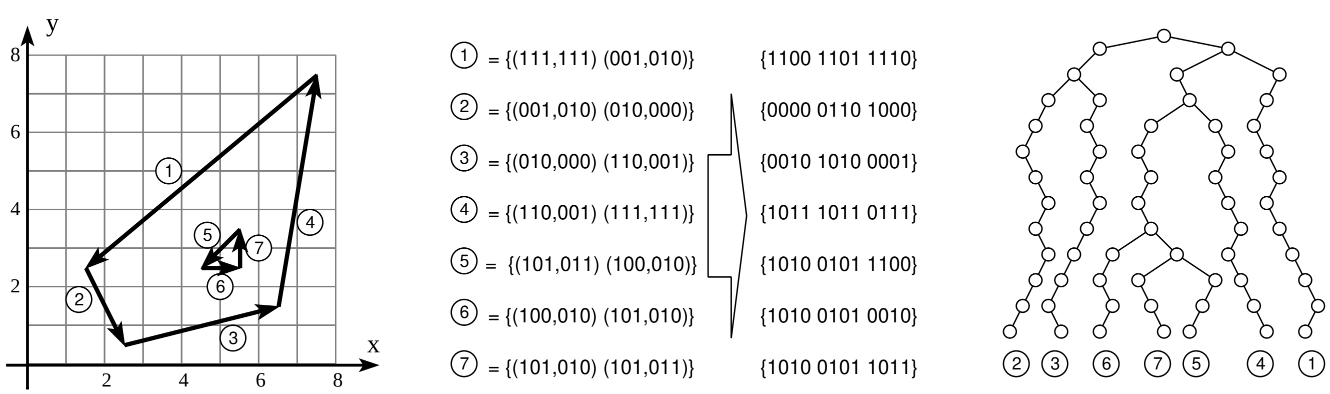
Bit interleaving, e.g.: `(6,5) = (110,101)` $\Rightarrow$ `111001`

kd-tries

2d-tries (∼ "Quad tree")

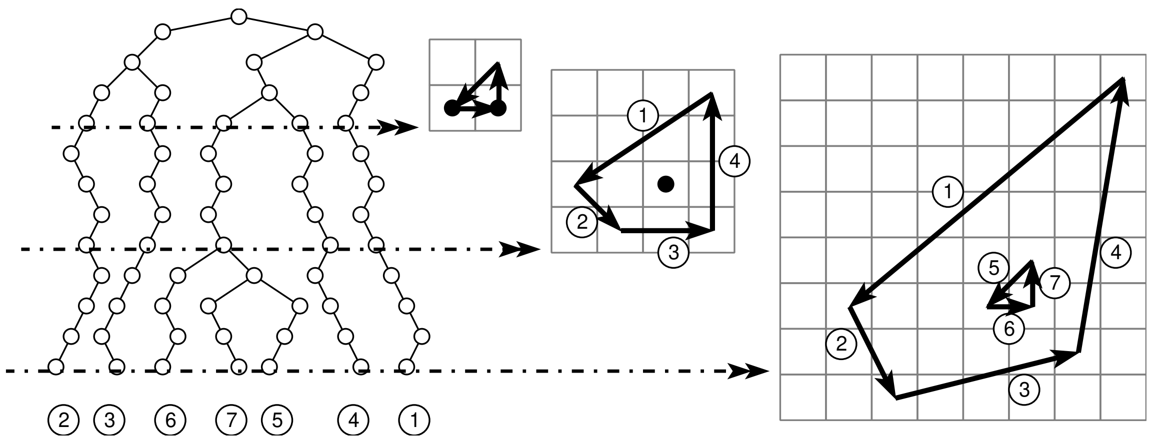
(1, 1)=	(0001, 0001)⇒	(00000011)
(6, 2)=	(0110, 0010)⇒	(00101100)
(8, 0)=	(1000, 0000)⇒	(10000000)
(8, 3)=	(1000, 0011)⇒	(10000101)
(10, 0)=	(1010, 0000)⇒	(10001000)
(12, 0)=	(1100, 0000)⇒	(10100000)
(14, 2)=	(1110, 0010)⇒	(10101100)
(8, 8)=	(1000, 1000)⇒	(11000000)

4d-tries



(a) A Map (b) Bit Interleaving (c) *ZoomTrie*

Variable-Resolution Tries

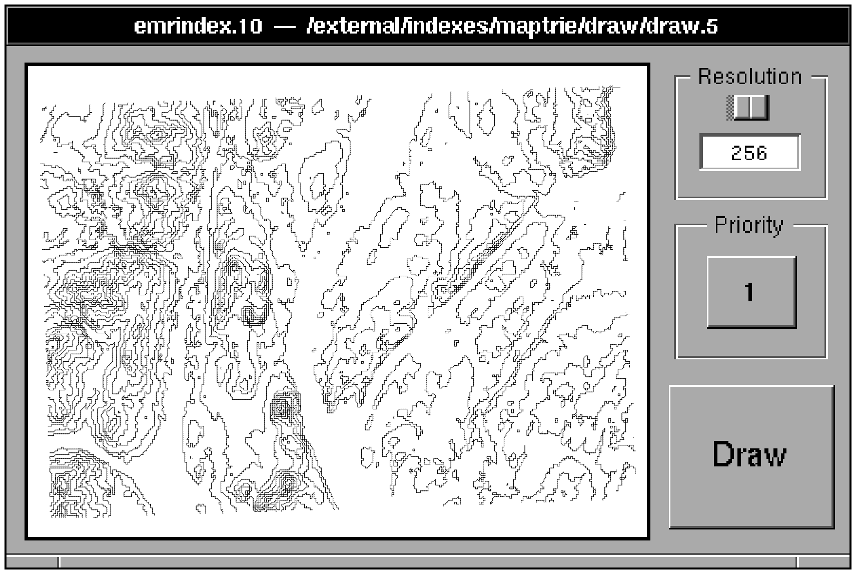


For example:

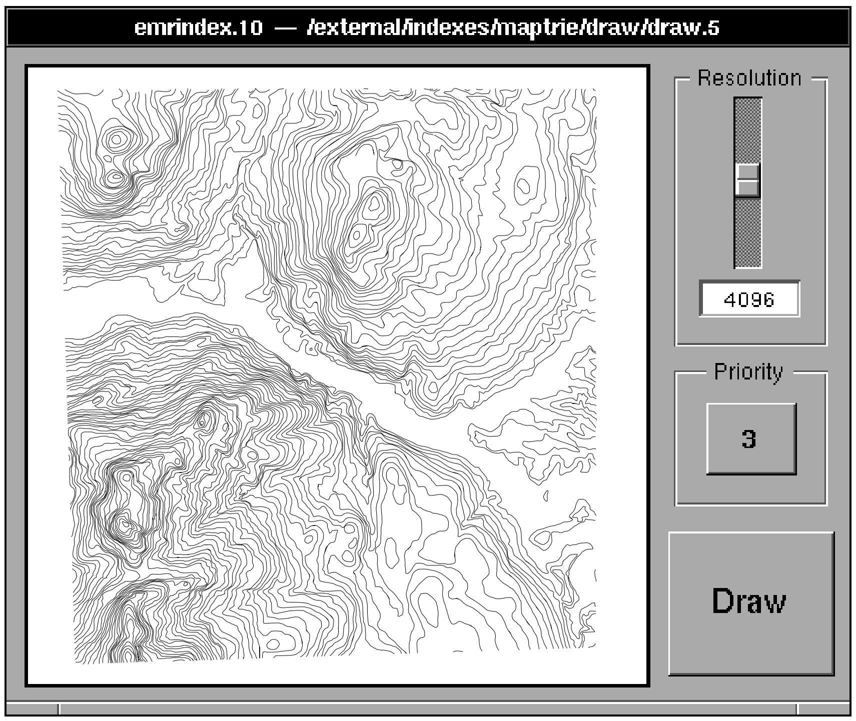
- 1:50,000 topographic series from E.M.R. Canada: 13,000 maps @ 2MB = 26GB.
- Megapixel display: 1 MB
- Can show, say, Magog, at full resolution,
- or all Canada at low resolution,
- or all Quebec at intermediate resolution.
- Total data transferred each time: 1MB

More generally, enhances selectivity.

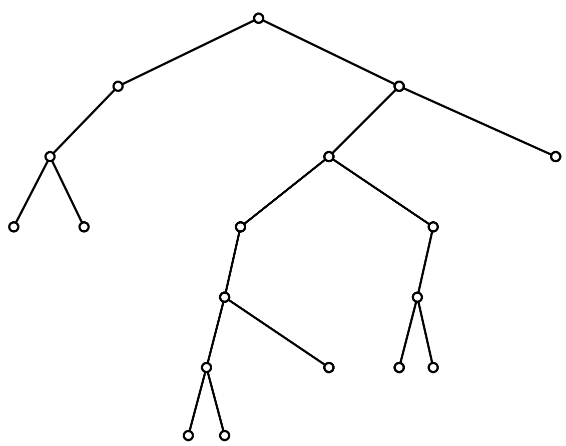
Full Contour Map at Low Resolution



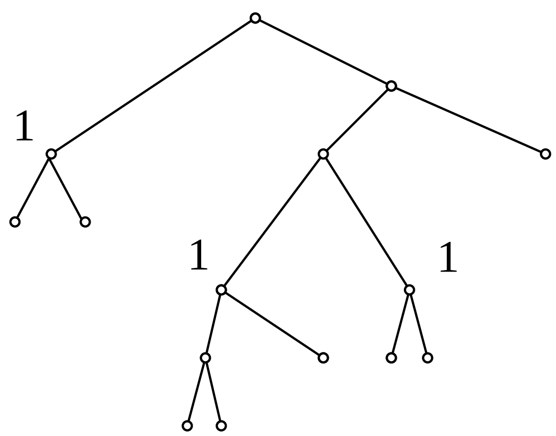
Contour Map Zooming



Truncated Tries

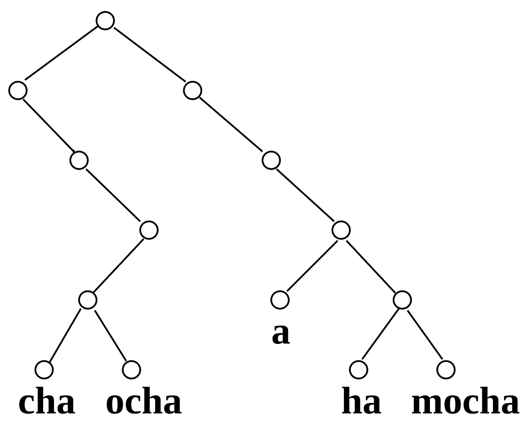


(a) Truncated Trie

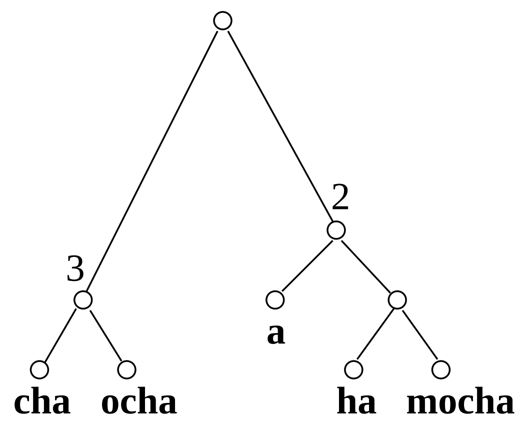


(b) Patricia Trie

Text and PATRICIA



1) Truncated Trie



2) PATRICIA Trie

Sample "text":

`mocha :  1110110101101111011000111110100011100001`

with "starts" every eight bits.

Note that pointers alone, to each byte of a 100 Mb text, would occupy 4 times the size of the text: text indexes are big. Cf. New OED

Using PATRICIA tries, stored 1 bit per node, and simple tricks, a byte-by-byte text index can be reduced to 2.7 times the text size. (Word-by-word: 0.7 times.)

Not only any substring but also arbitrary regular expressions can be found.

k-Approximate String Matching

"edit distance", $D_{i,j}$:

- $D_{i,j-1}$, target + 1 char
- $D_{i-1,j}$, pattern + 1 char
- $D_{i-1,j-1}$, each + 1 char
- $D_{i-2,j-2}$, 2 chars swapped

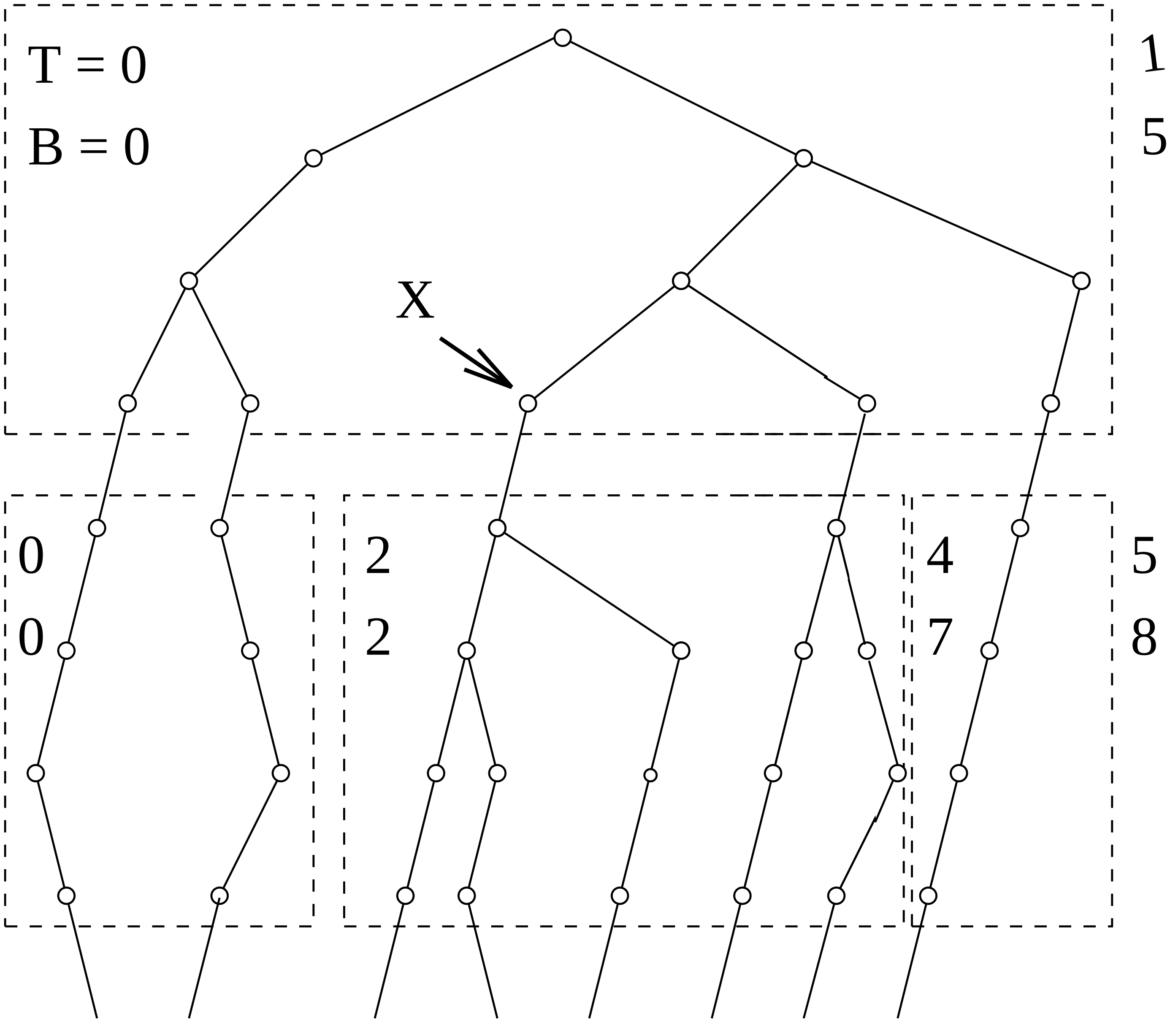
Dynamic programming.

One table per target string:
Trie ⇒ shared prefixes ⇒
Tables share columns.

Speedup of trie over *agrep*
(Texts 4.5–8.4MB: N.B. *agrep* is linear)

	k=0	k=1	k=2
min	7.3	1.6	0.2
max	28.4	9.5	0.7

Paging the Trie for Secondary Storage



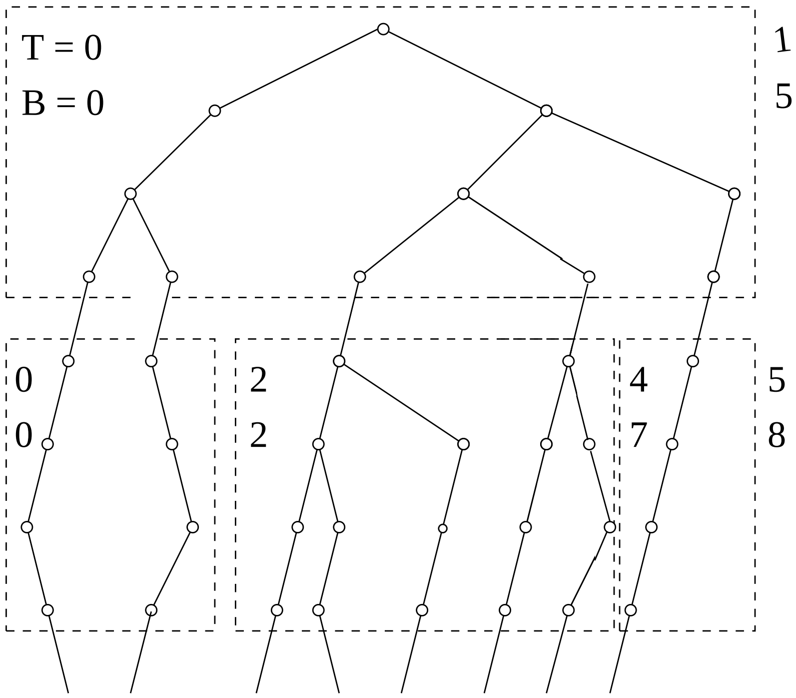
Representing the Paged Trie

With a Search for `10001000`

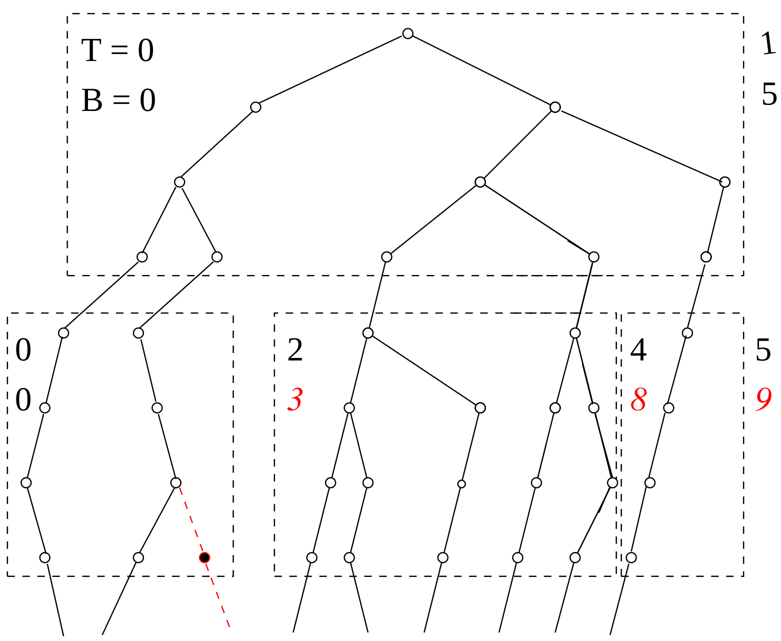
T 0	1x	T 1	
B 0	10 x1	B 5	
	11 x1 10		
	10 10 x0 10 10		

T 0	10 01	T 2	1x 11	T 4	10	T 5
B 0	10 01	B 2	11 x0 10 01	B 7	10	B 8
	01 10		10 10 x0 10 10		10	
	01 10		10 01 x0 10 10		10	

Dynamic Tries

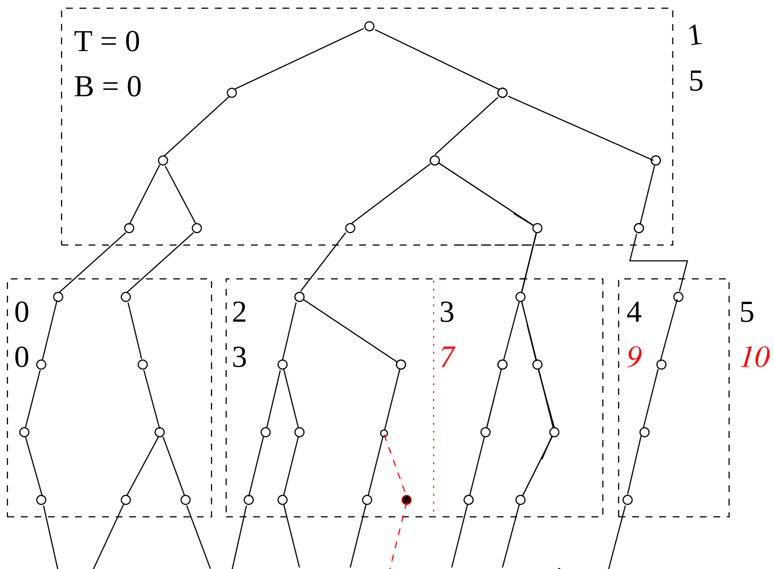


Adding 00101111



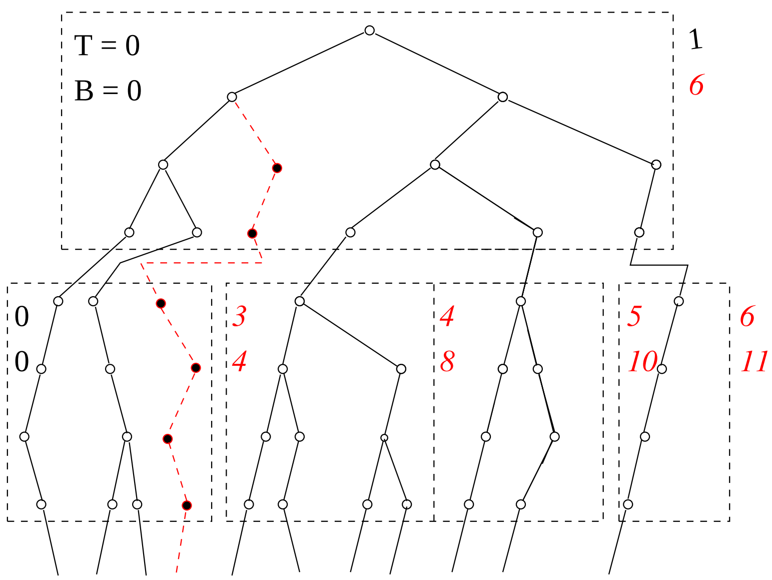
Updating Tries

Adding 10001010



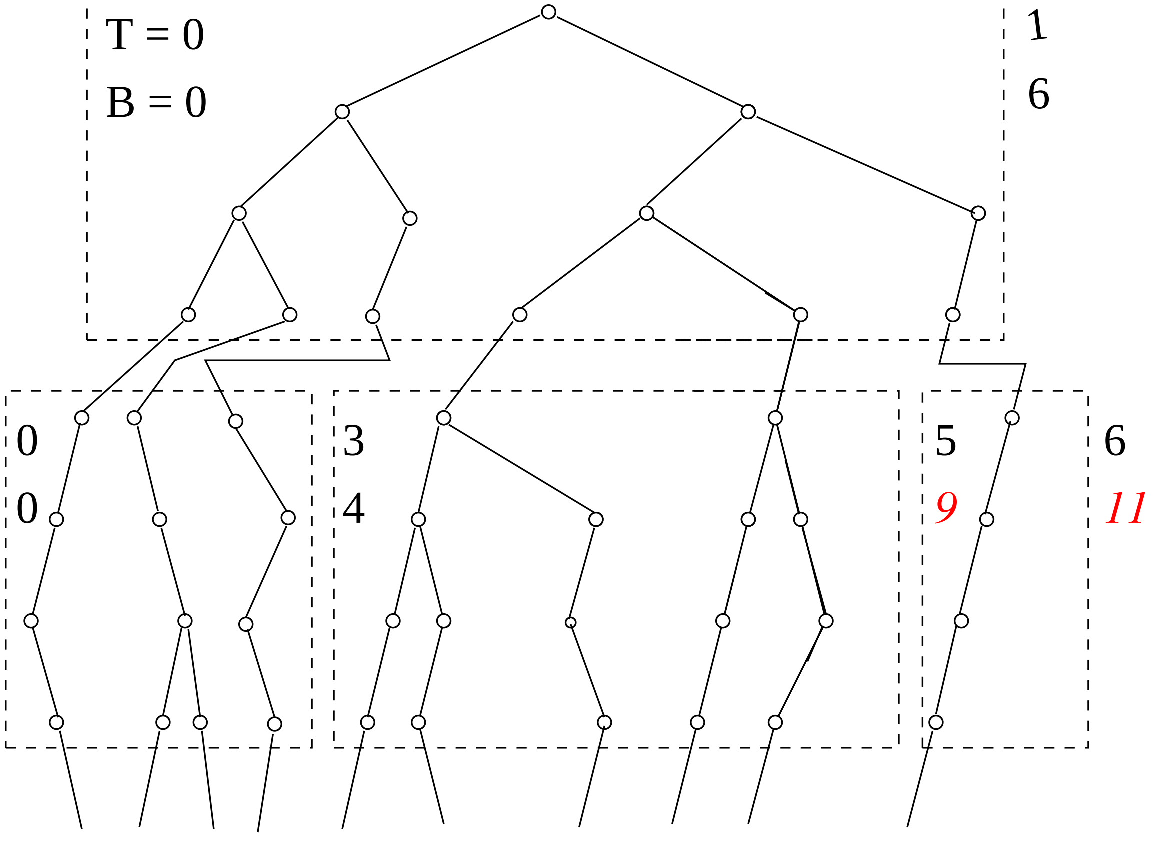
N.B. t node levels per page level; $\geq 2^t - 1$ nodes per page

Adding 01011010



Updating Tries

Deleting 10001000



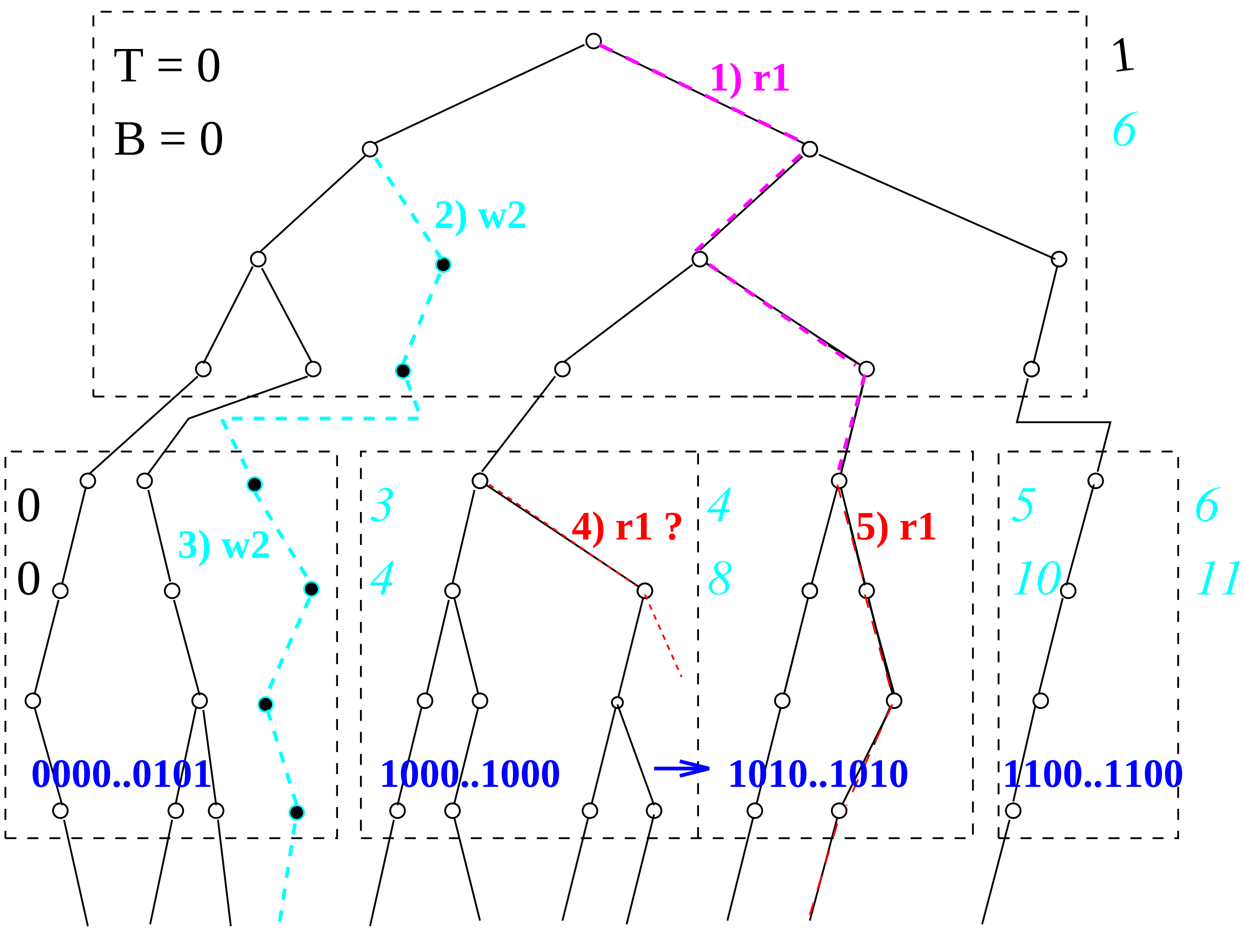
Concurrency

Counts stored separately for dynamic tries:
must lock at least the trailing counts of a level.
This reduces concurrency.

E.g. read 10101100 (transaction 1)
and concurrently
write 01011010 (transaction 2):

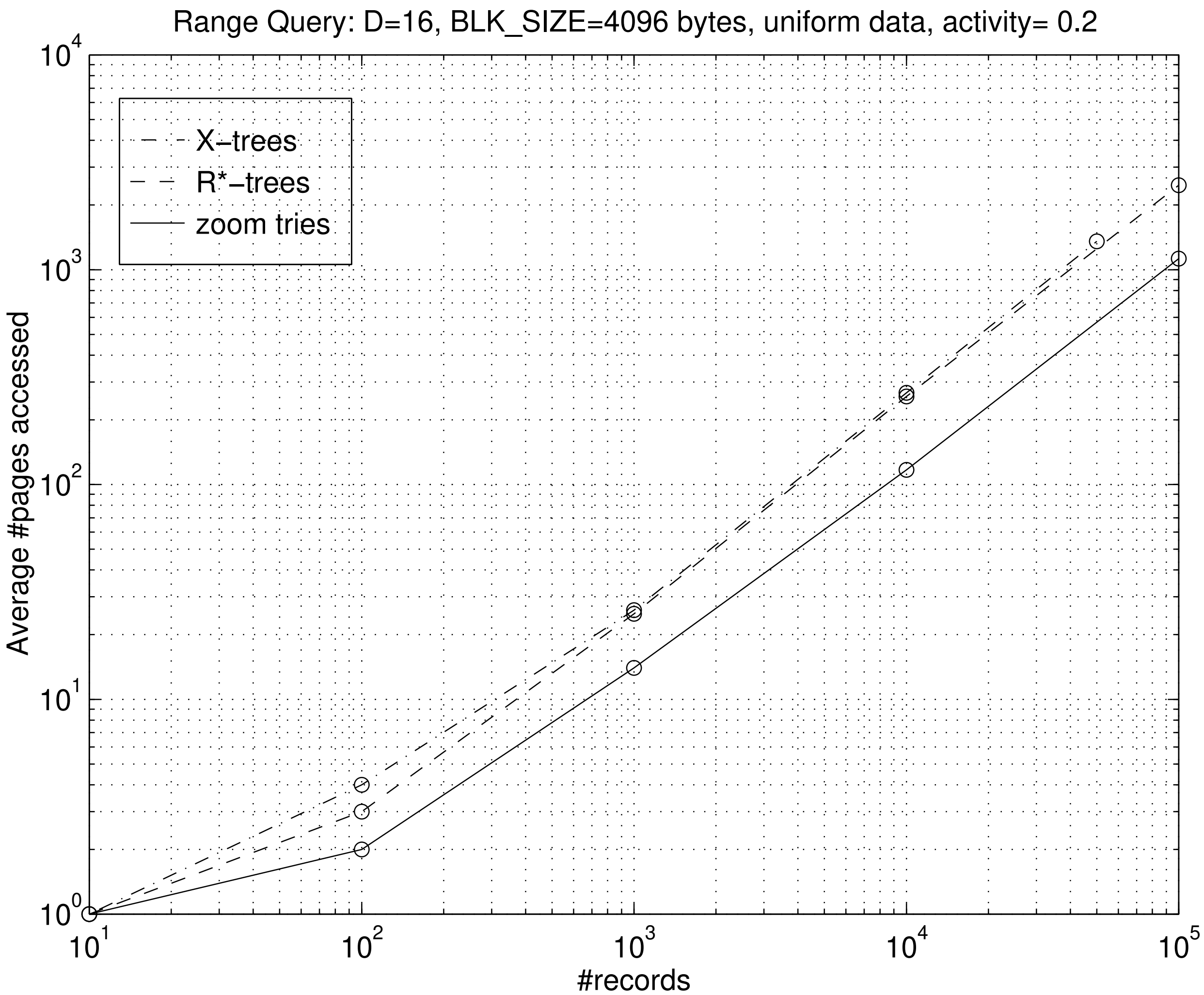
r1 1010	#4
w2 0101	#4 $\rightarrow$ #5
w2 1010	
r1 1100	not found!

- B-trees can support read/write concurrency by advancing the read page if a write happens to have messed it up.
- Tries do not have enough redundancy to check for a mess.
- So add redundancy: on each page, store the lowest and highest prefix from preceeding pages.



- Two writes can lock each other out of the whole trie.
- B-trees have the same problem because root may be split.

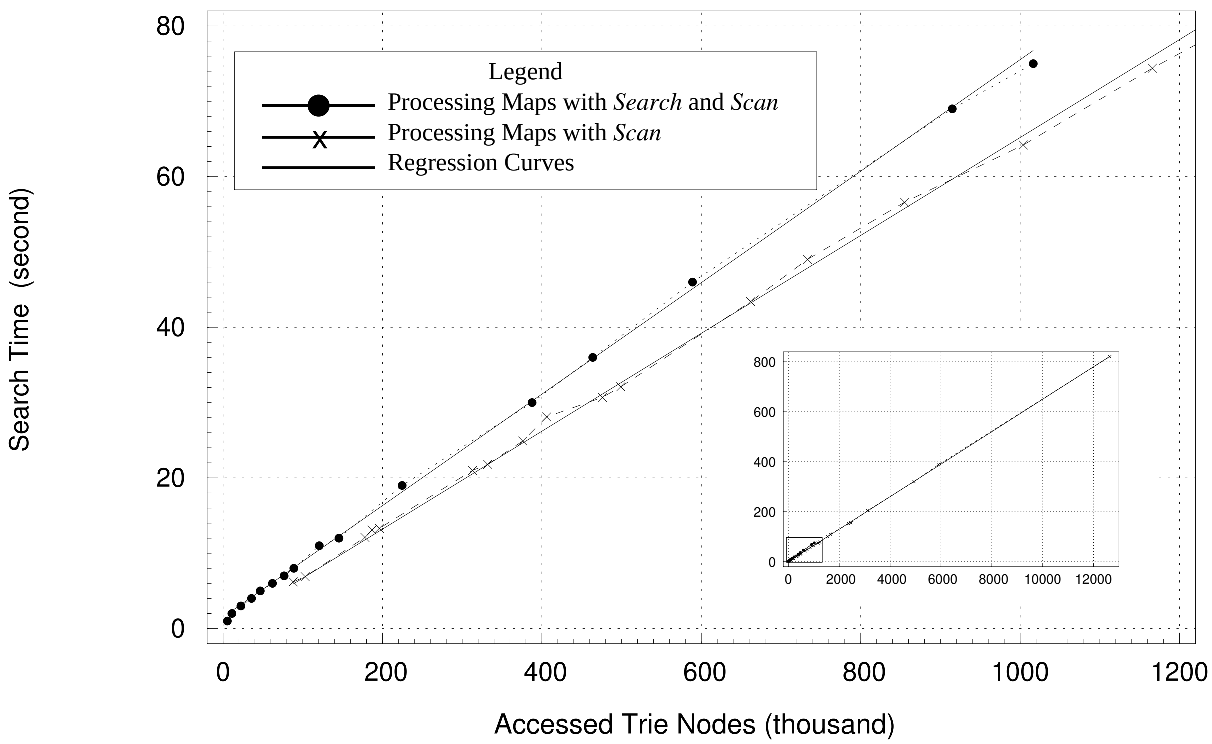
Dependence on File Size is Logarithmic



T. H. Merrett ©98/10

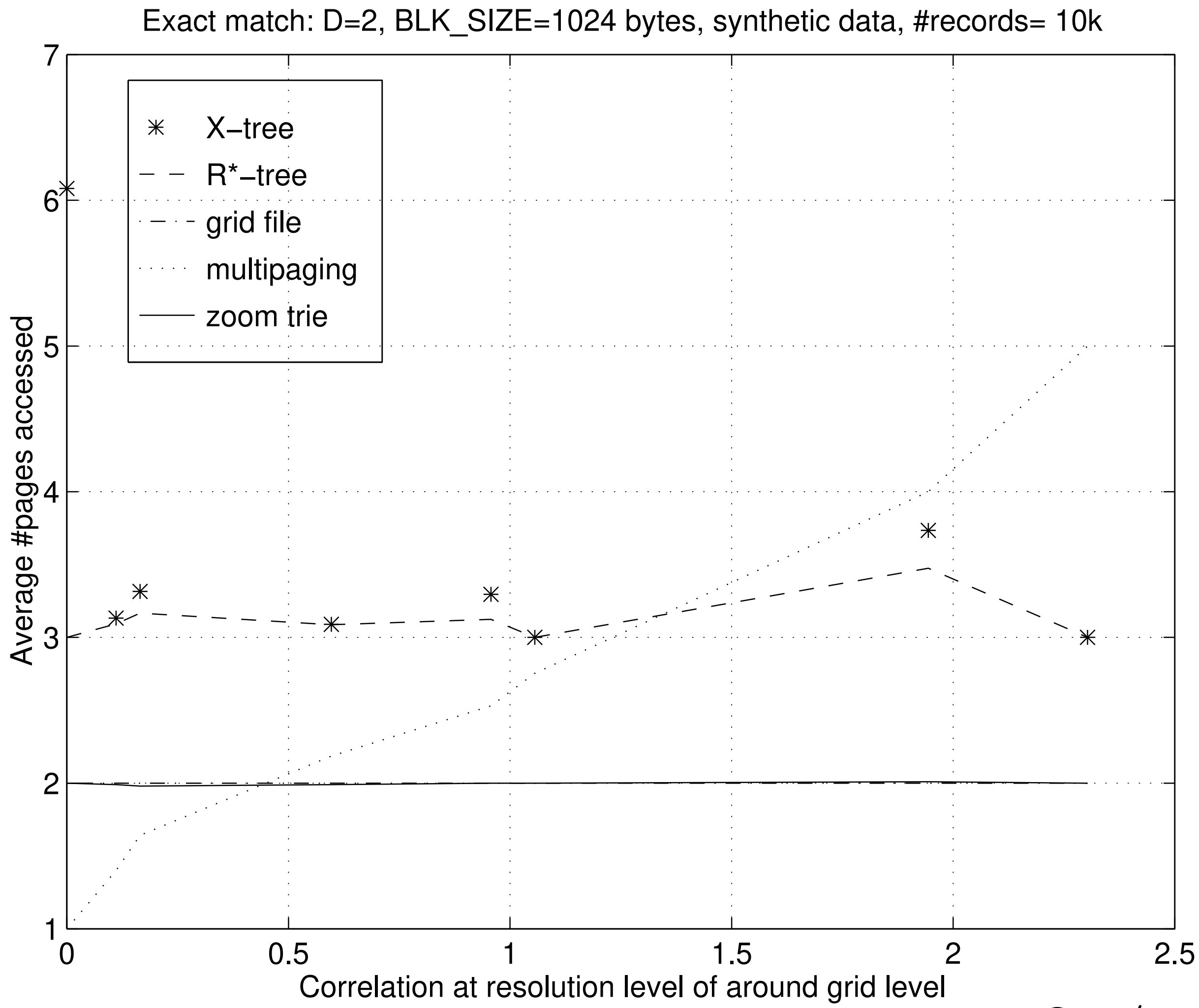
Search Costs are Optimal

Map Displays

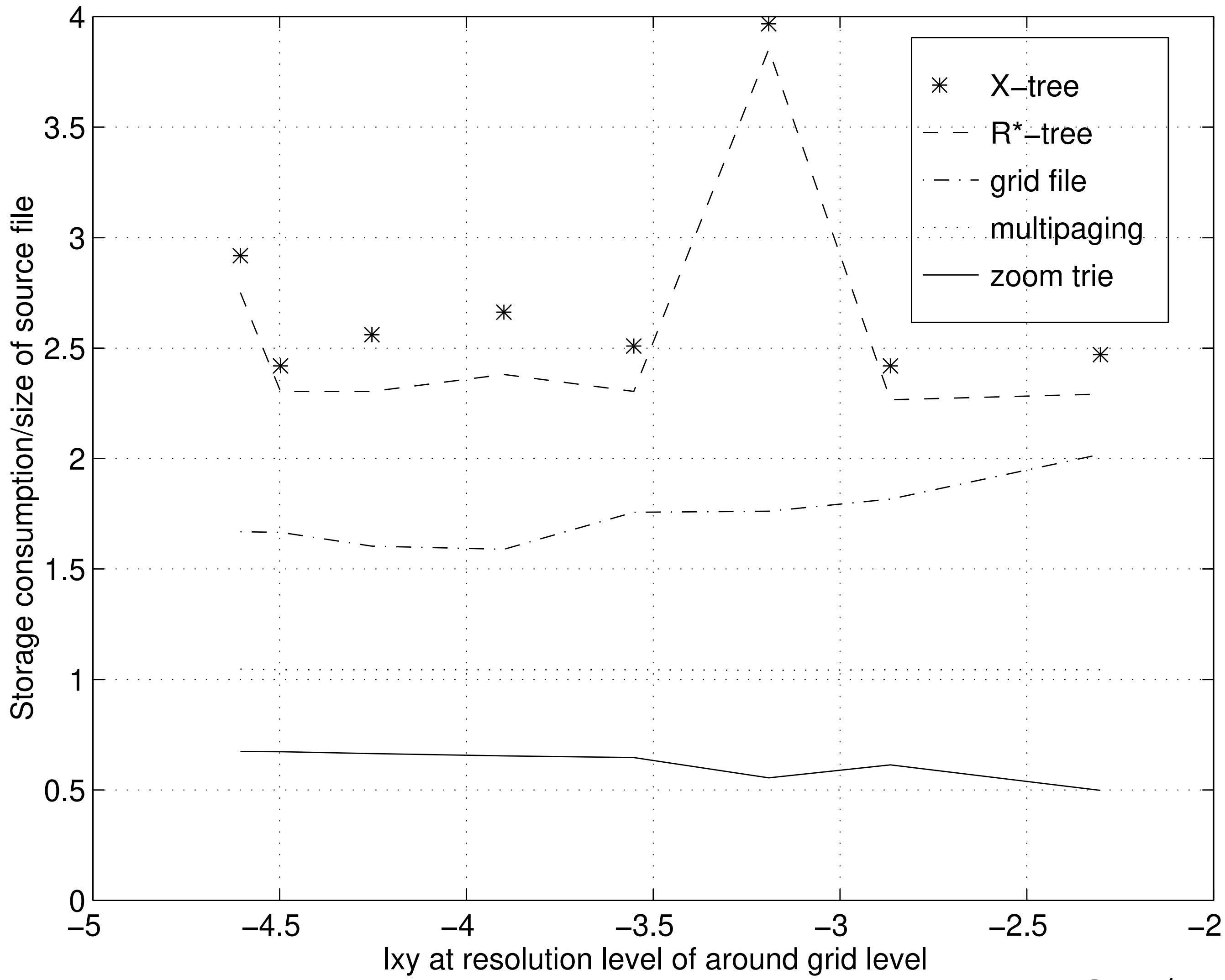


T. H. Merrett ©98/10

Data Distributions Have No Effect



Storage consumption: D=2, BLK_SIZE=1024 bytes, synthetic data, #records= 10k



T. H. Merrett

The "Curse of Dimensionality"

(OLAP and Feature Vector Applications)

Selectivity blowup:
Applies to any d-dim. paging that partitions the axes. Assumes (1) data distribution is Cartesian product, (2) range query has same shape as space.

		$f=2$		$f=5$		$f=10$	
d	$s^{1/d}$	n	s_{eff}	n	s_{eff}	n	s_{eff}
1	.0001	2	.5	5	.2	10	.1
2	.01	4	.25	25	.04	100	.01
4	.1	16	.06	625	.002	$1_{10}4$	.0001
8	.31	256	.004	$3.9_{10}5$	.0007	$1_{10}8$	.0007
16	.56	64K	1	$1.5_{10}11$	.0003		.0003
32	.75	$4.3_{10}9$	1		.0008		.0008
64	.87		1		1		.001
128	.93		1		1		1
256	.96		1		1		1
512	.98		1		1		1
1024	.99		1		1		1

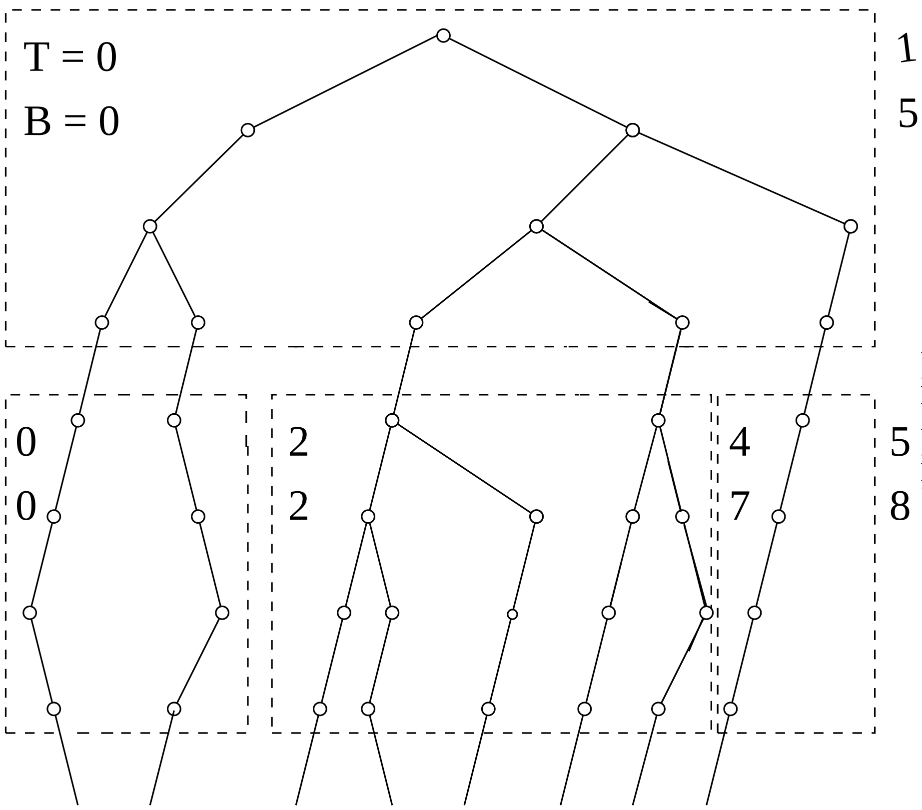
N.B. $f=\infty$: $s_{\text{eff}} =$ s

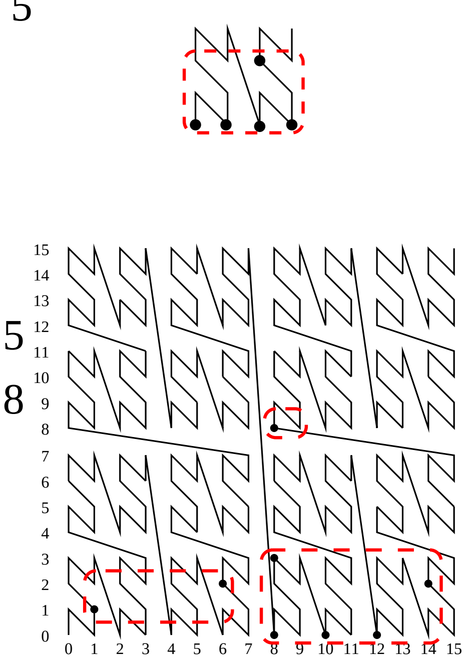
s: selectivity; s_{eff}: effective selectivity due to paging;

f: number of page partitions per axis; n: number of pages.

So it's not all that bad, but ..

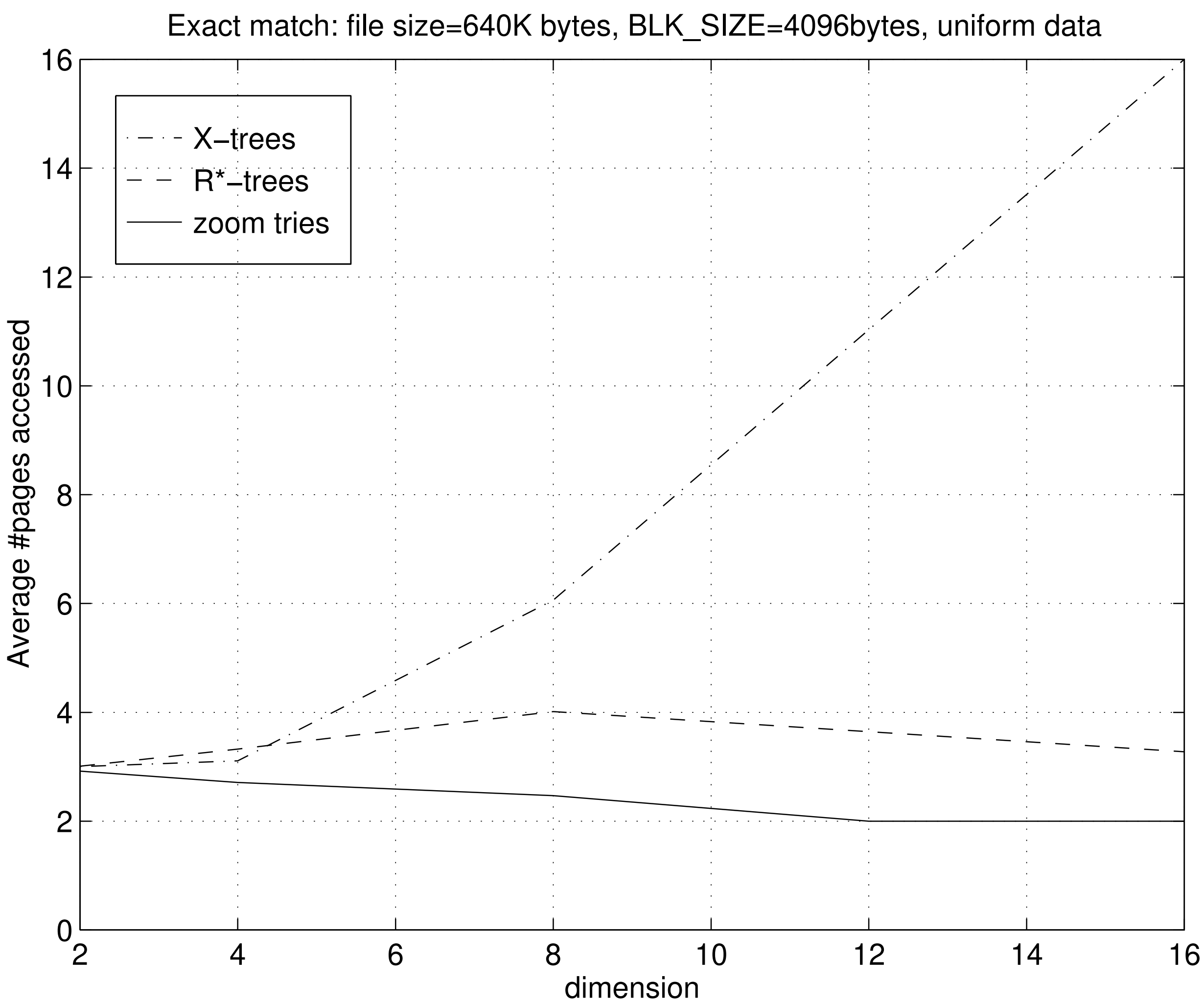
But trie pages follow Z-order, in *one*-dim.

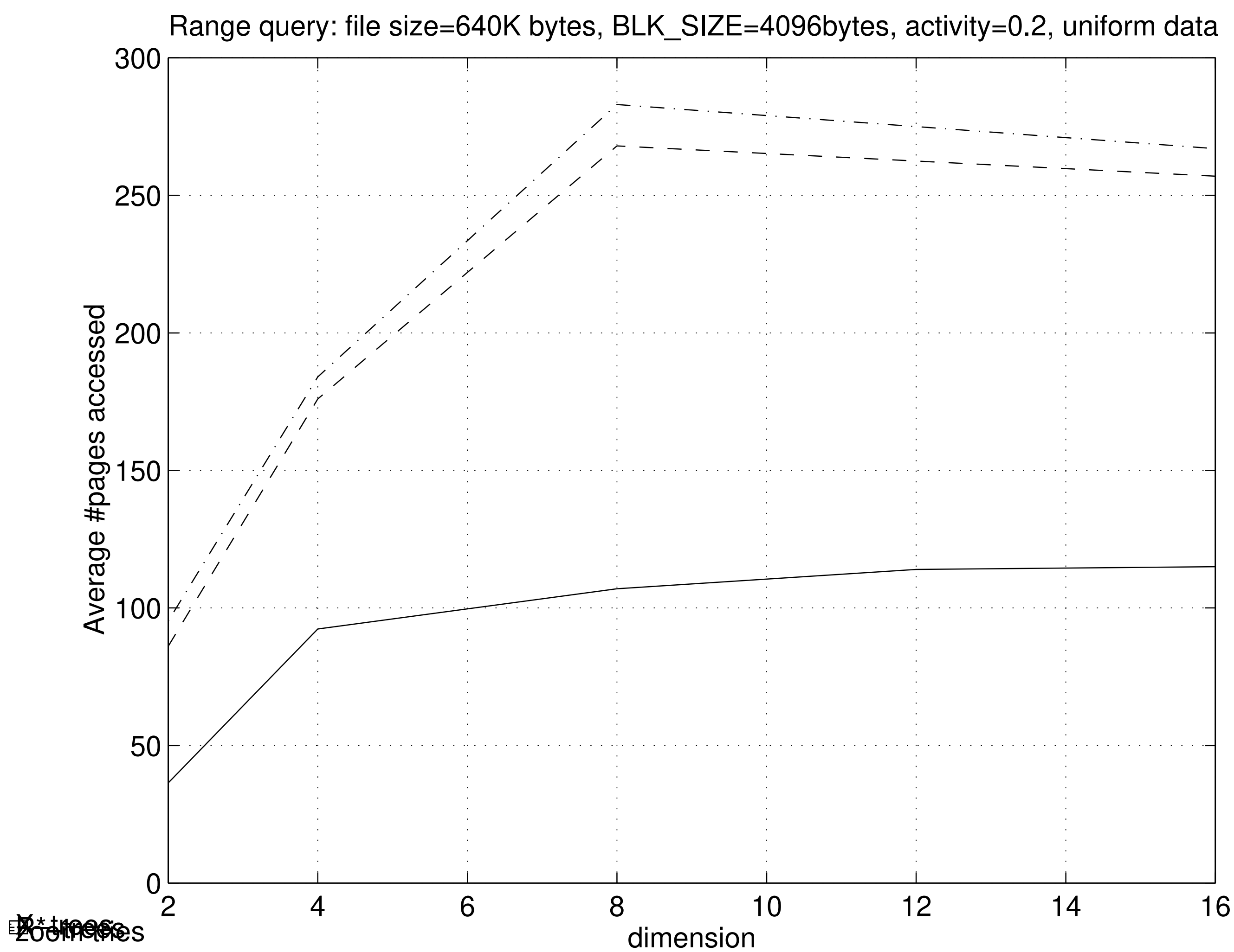




Not a grid.

Data Dimensions Have No Effect




Range query: file size=640K bytes, BLK_SIZE=4096bytes, activity=0.2, uniform data
Average #pages accessed
dimension
0
50
100
150
200
250
300
2
4
6
8
10
12
14
16
R*-trees
Zoom-tries