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.

The Pointerless Representation of Tries

T. H. Merrett

McGill University

I Tries

- Compression down to 2 / lg n on n data items

  e.g., 90% (1 Mbyte) 93% (1 Gbyte) 95% (1 Tbyte)

  - Good for suffix trees
    better than suffix arrays [FODO'93]
  - Support regex and approximate matching

- Variable resolution

- Multidimensional trees and Z-order

- Dynamic

II Pointerless representation

1. RAM: main memory

2. SS: secondary storage

Orenstein 1983

www.cs.mcgill.cs/∼cs420/logarithmicTxt.ps

Compression

Raw data: 2^h items of h bits each.

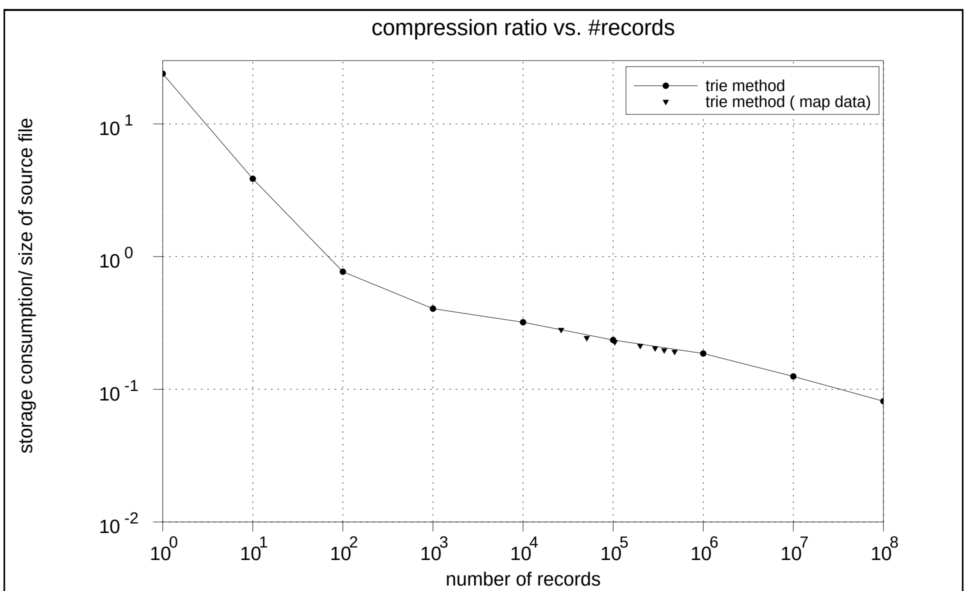
Trie: $2^h - 1$ nodes of 2 bits each
(pointerless representation).

Compression: $h \longrightarrow 2$

For n items, $h = \lg n$.

Theoretical best:
90% (1 Mbyte)   93% (1 Gbyte)   95% (1 Tbyte)

Experiment (log-log scale; 90% at 10^7 records):



Suffix tries *vs.* suffix arrays

Simplistic suffix array size

n lg N / 8 for n suffixes, N bytes

E.g., $3.4n$ for 100 Mbytes

[FODO '93]:

"For an index of 100 million entries, our experiments show size factors of less than 3, as compared with 3.4 for the best previous method.

Our measurements show expected access costs of 0.1 sec., and construction times of 18 to 55 hours, depending on the text characteristics."

www.cs.mcgill.cs/~tim/cv/theses/shang.ps.gz

Regex and approximate matching

[FODO '93]:

"Our organization .. supports searches for general patterns, as well as a variety of special searches, such as proximity, range, longest repetitions and most frequent occurrences."

[IEEE TKDE 8 '96]:

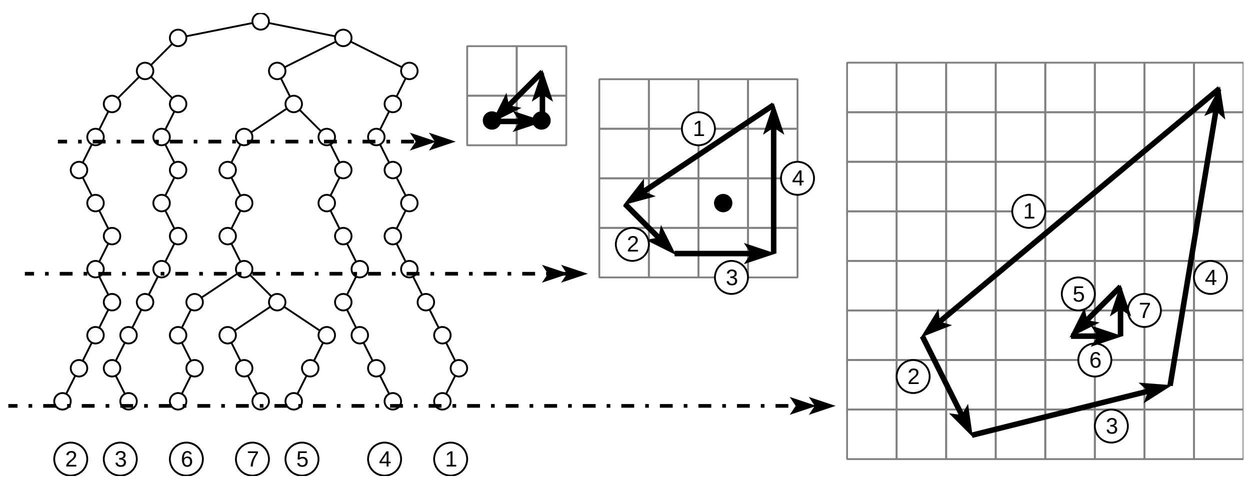
"We discuss a variety of applications and extensions, including best match (for spelling checkers), case insensitivity, and limited approximate regular expression matching."

Variable resolution

For low resolution, access only top of trie.

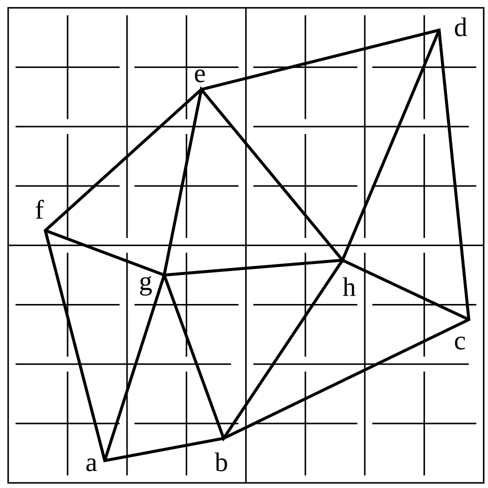
For higher resolution, go deeper.

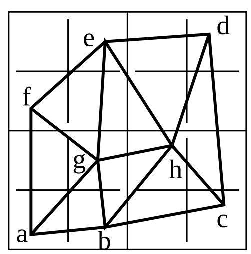
E.g., a simple map:

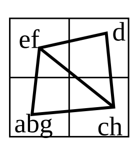


Variable resolution

A triangulated irregular network.

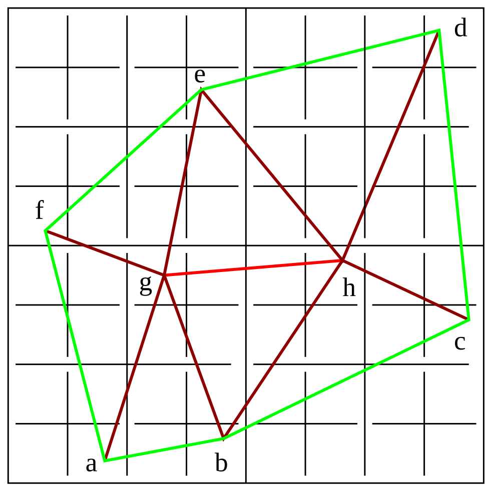




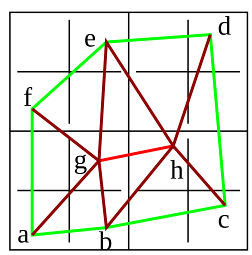


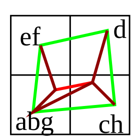
Variable resolution

A triangulated irregular network with heights.



a,b,c,d,e,f: height 0   g,h: height 7

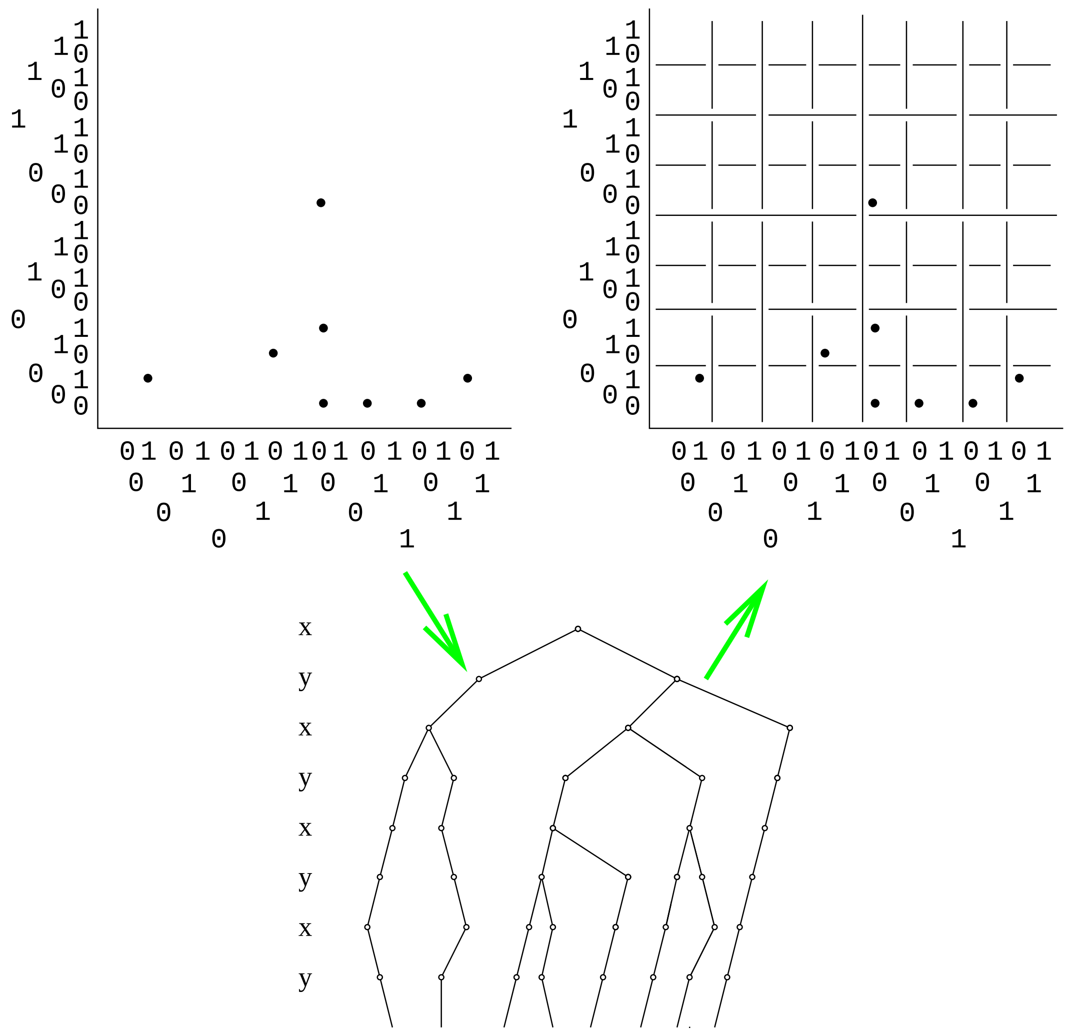




(The 3d-trie keeps the height distinction at lower resolutions.)

Multidimensional tries and Z-order

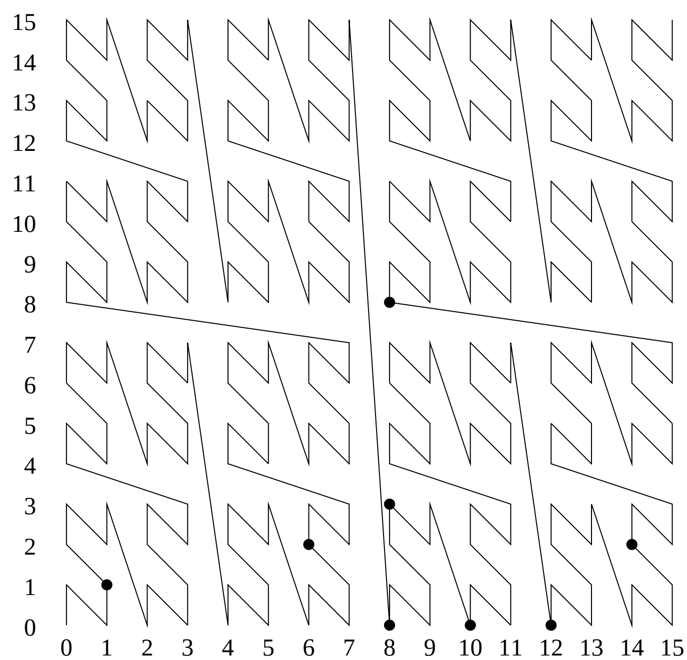
Trie interpreted in 2D:
Bentley's "discriminator"



Bit interleaving, e.g.,: (6,2) = (0110,0010) ⇒ 00101100

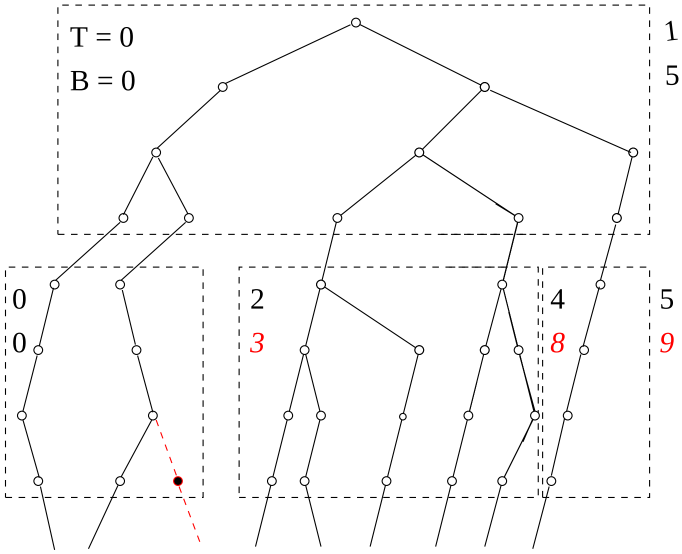
Multidimensional tries and Z-order, cont.

A 1D ordering of the same 2D data



Dynamic

E.g., adding 00101111

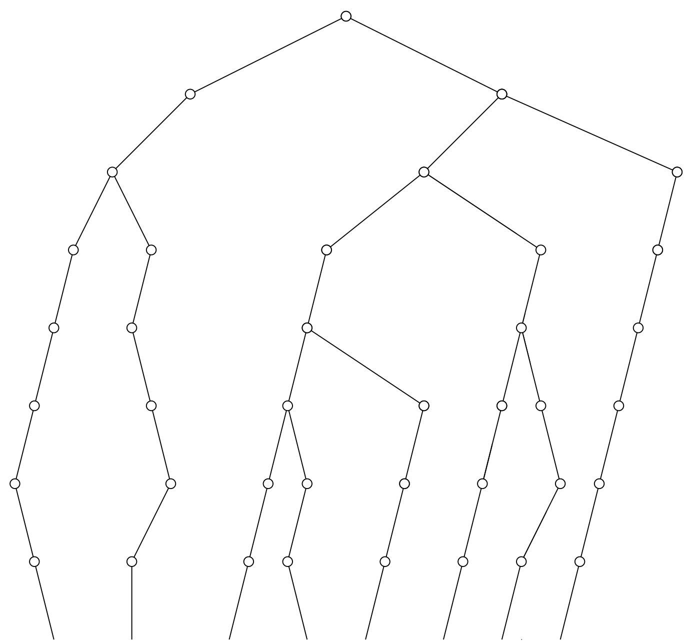


II Pointerless representation

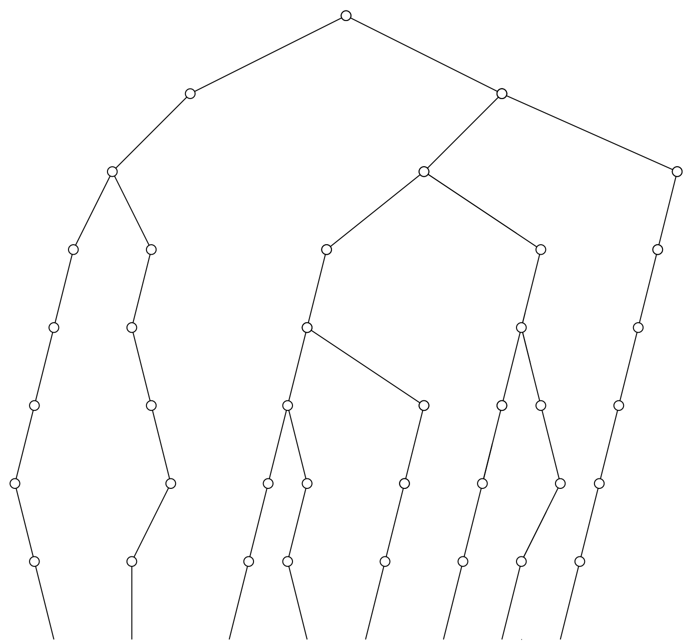
Trie in RAM

E.g., eight data values

00000011	00101100	10000000	10000101
10001000	10100000	10101100	11000000



Two bits per node



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

Two bits per node, cont.

```
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
```

```
11                          2 on-bits mean 2 nodes next level
10 11                       3 on-bits mean 3 nodes next level
11 11 10                    5 on-bits mean 5 nodes next level
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
```

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

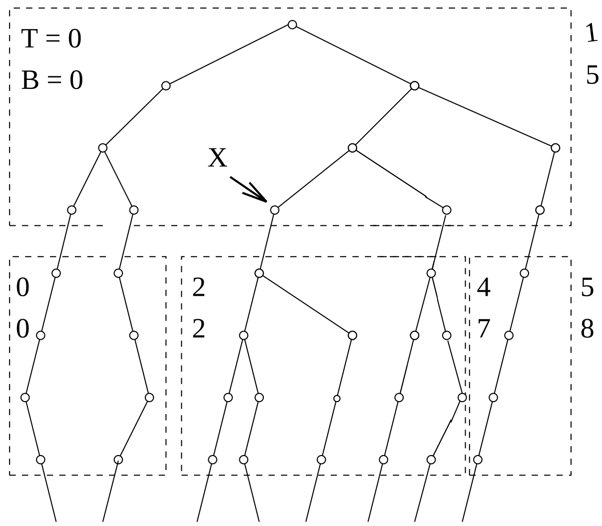
Searching

Search for 10001000

```
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
```

```
1x                          look in 2nd node, next level
10 x1                       look in 2nd node, next level
11 x1 10                    look in 3rd node, next level
10 10 x0 10 10                                        ..
10 01 1x 11 10
10 01 11 x0 10 01 10
01 10 10 10 x0 10 10 10
01 10 10 01 x0 10 10 10
```

Trie on SS



T 0	11					T 1			
B 0	10 11					B 5			
	11 11 10								
	10 10 10 10 10								

T 0	10 01	T 2	11 11	T 4	10	T 5	
B 0	10 01	B 2	11 10 10 01	B 7	10	B 8	
	01 10		10 10 10 10 10		10		
	01 10		10 01 10 10 10		10		

Search [Orenstein, '83]

```
1x
10 x1
11 x1 10
10 10 x0 10 10
10 01 1x 11 10
10 01 11 x0 10 01 10
01 10 10 10 x0 10 10 10
01 10 10 01 x0 10 10 10
```

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

.. look in 3rd node, next level ..

3rd node must be in page headed T=2:

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		

Conclusion

Two bits per node, shared storage of prefixes, hence compression.

Multidimensional tries and variable resolution both follow.

Paged representation (Orenstein) adds only 2 integers per page.

Dynamic tries follow.

• Orenstein, *Algorithms for Implementing Relational Databases*, 1983, Ph.D. Thesis, McGill University, School of Computer Science

• Merrett & Shang, *Trie Methods for Representing Text*, FODO'93 LNCS 730, 1993, 130–45.

• Shang, *Trie Methods for Text and Spatial Data on Secondary Storage*, 1995, www.cs.mcgill.ca/~tim/cv/students.html

• Shang & Merrett, *Tries for Approximate String Matching*, IEEE TKDE **8** (4), 1996, 540–7.

• www.cs.mcgill.cs/~cs420/logarithmicTxt.ps