Keyword Dictionary Compression

Using Efficient Trie Implementation

Toshiyuki Masui

SHARP Corporation

Center for Machine Translation
Carnegie Mellon University

April 9, 1991

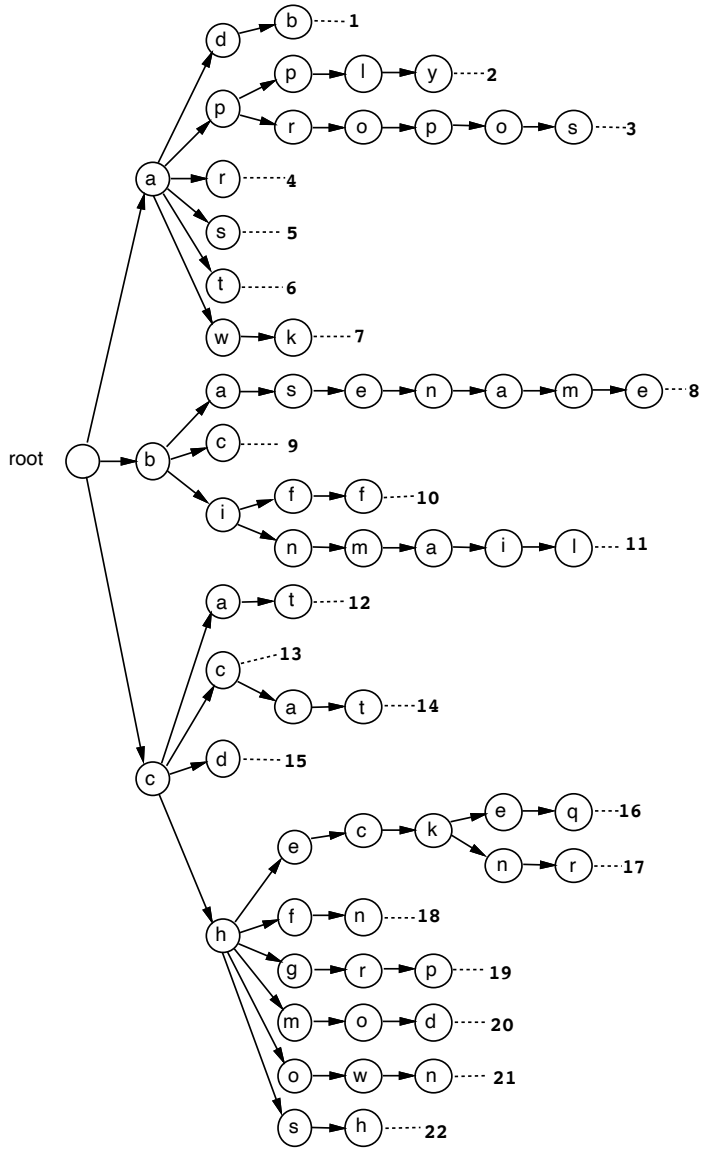
Keyword Dictionary

keyword	attribute value
adb	**1**
apply	**2**
apropos	**3**
ar	**4**
as	**5**
at	**6**
awk	**7**
basename	**8**
bc	**9**
biff	**10**
binmail	**11**
cat	**12**
cc	**13**
ccat	**14**
cd	**15**
checkeq	**16**
checknr	**17**
chfn	**18**
chgrp	**19**
chmod	**20**
chown	**21**
chsh	**22**

Keyword Dictionary Implementation

- Hash Table
  - Used in many applications
  - Requires a large hash table
  - Requires computation of the hash value every time a keyword is searched
  - Requires extra search time when hash table is nearly full
- Trie
  - Fast search
  - Requires huge space with poor implementation
  - Can be compressed with proper node representation techniques

Structure of a Keyword Dictionary Trie



Requirements for Trie Compression

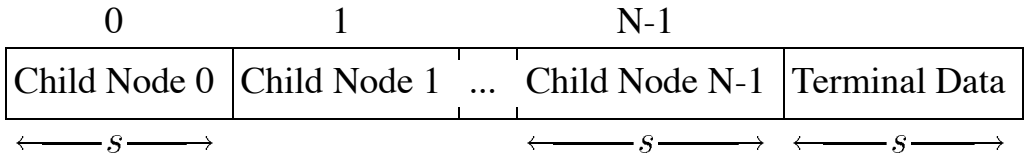
- Links should be preserved to allow fast access
- Nodes and data should be compressed

Definition

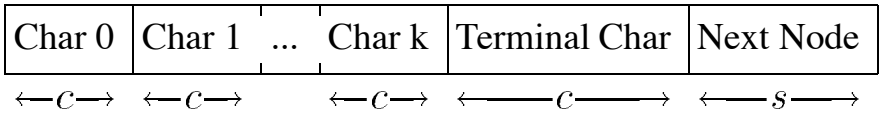
N: # of characters (alphabets)
C: # of children of a node
c: # of bytes required to store one character
s: # of bytes required to hold a pointer to another node

Techniques for Trie Node Compression [1][4] [6]

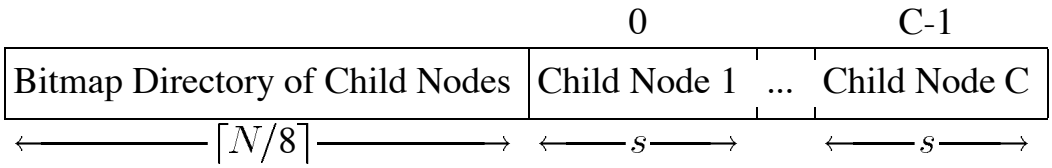
- Array Representation
  - Simplest representation
  - Requires $(s(N+1))$ bytes



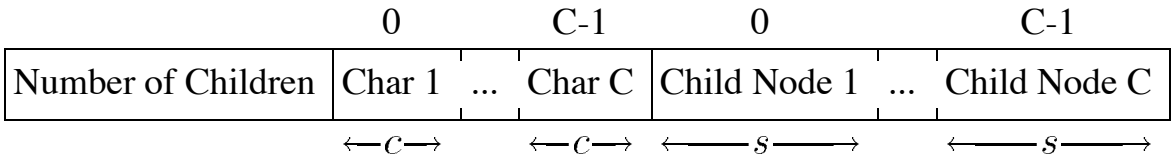
- String Representation[2][3]
  - Ideal when a node has only one child



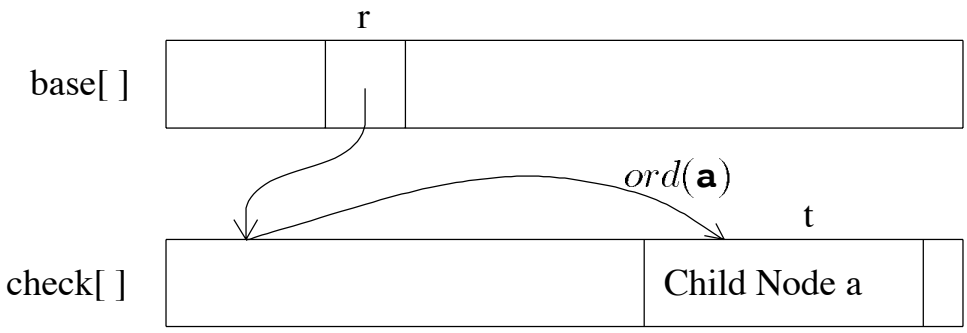
- Bitmap Representation[5]
  - Requires $(\lceil N/8 \rceil + c \times s)$ bytes



- List Representation
  - Requires $(C(c+s) + \lceil log_2 N/8 \rceil$ bytes

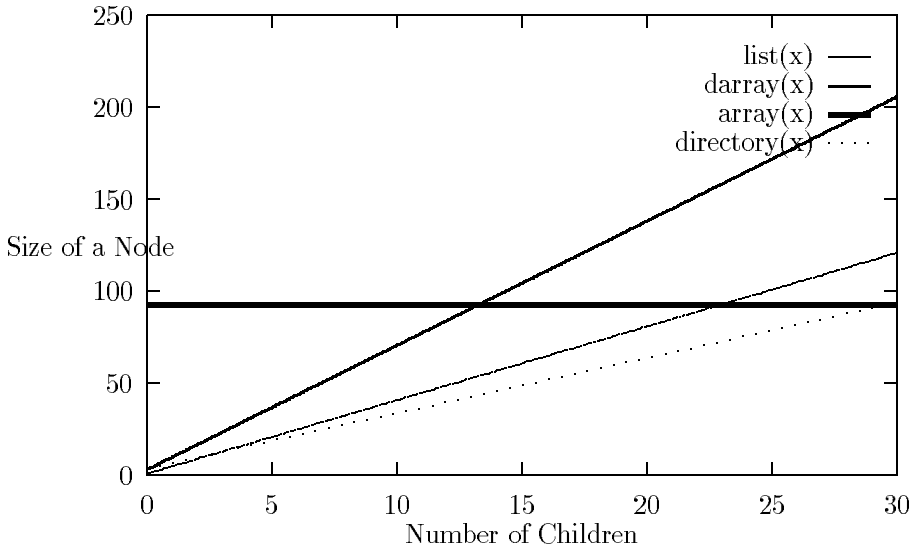


- Double Array Representation[2][3]
  - One of the node interleaving techniques
  - Consists of two arrays (base[] and check[])
  - Requires $(s + c \times s)$ bytes if interleaving is ideal
  - Usually requires more space than the combination of other techniques



Combination of the Techniques

Memory Consumption Based On Each Technique



Strategy:

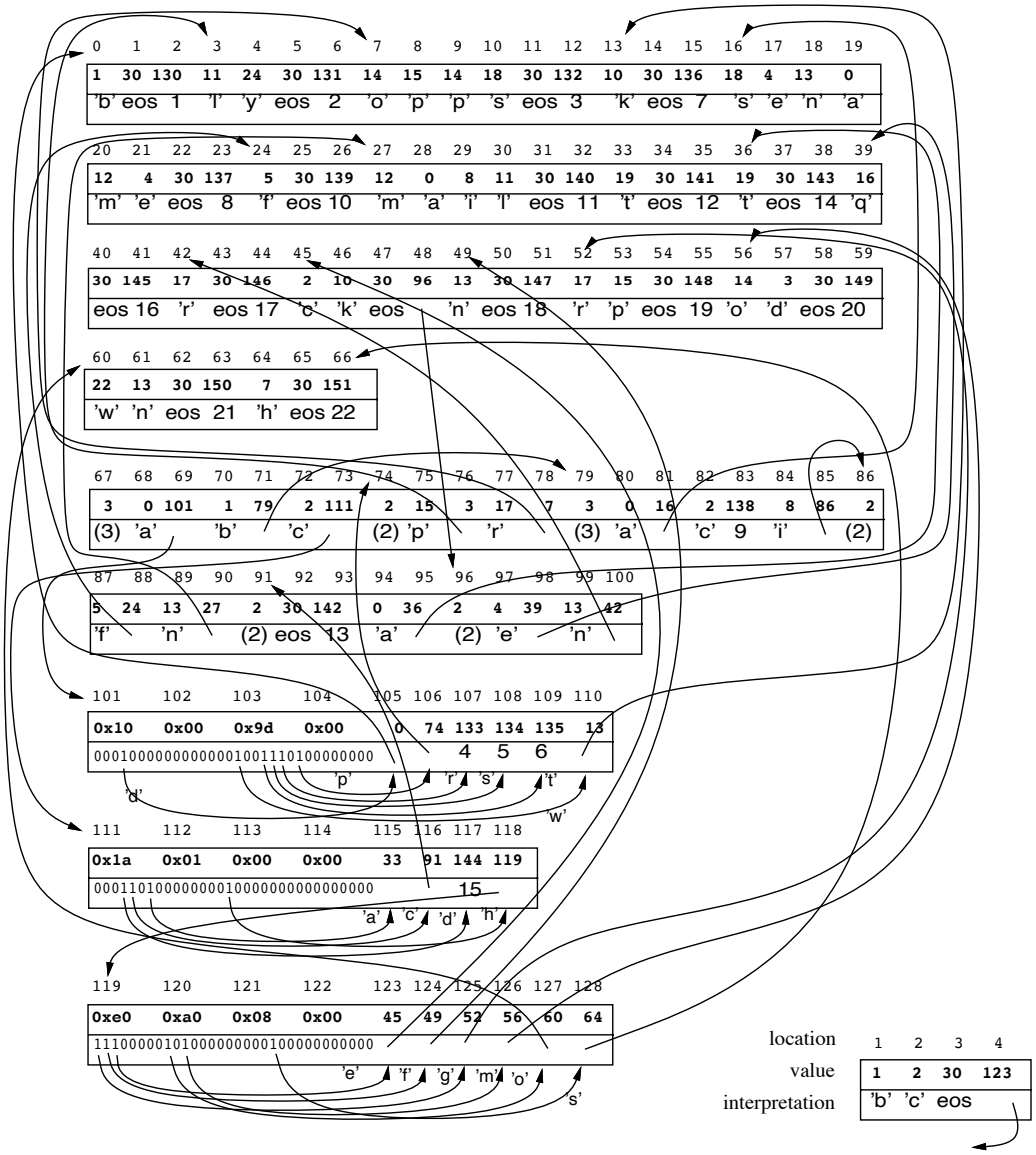
- The node has only one child
  $\Rightarrow$ Use string representation
- The node has small number of children
  $\Rightarrow$ Use list representation
- The node has many children
  $\Rightarrow$ Use bitmap directory representation

- Almost all the characters have their corresponding children
  $\Rightarrow$ Use array representation

When the number of alphabets is 30, node representation should be chosen like below.

Number of Children	Node Representation Technique
1	String
2-3	List
4-28	Bitmap Directory
29-30	Array

Example of a Compressed Trie



Bibliography

References

[1] Al-Suwaiyel, M. and Horowitz, E. *Algorithm for Trie Compaction*. **ACM Transactions on Database Systems**, vol. 9 (1984), pp. 243–263.

[2] Aoe, J. *An Efficient Digital Search Algorithm by Using a Double-Array Structure*. **IEEE Transactions on Software Engineering**, vol. 15 (1989), pp. 1066–1077.

[3] Aoe, J. *An Efficient Implementation of String Pattern Matching Machines for a Finite Number of Keywords*. **SIGIR Forum**, vol. 23 (1989), pp. 22–33.

[4] Comer, D. *Heuristics for Trie Index Minimization*. **ACM Transactions on Database Systems**, vol. 6 (1979), pp. 383–395.

[5] Purdin, T. D. M. *Compressing Tries for Storing Dictionaries*. in: **Proceedings of the 1990 Symposium on Applied Computing**, IEEE, ACM, edited by H. Berghel, J. Talburt, and D. Roach. 1990, pp. 336–340.

[6] Ramesh, R., Babu, A. V. G., and Kincad, J. P. *Variable-depth Trie Index Optimization: Theory and Experimental Results*. **ACM Transactions on Database Systems**, vol. 14 (1989), pp. 41–74.