Tries

Some of these lecture slides have been adapted from:

- *Algorithms in C, 3rd Edition*, Robert Sedgewick.

Symbol Table Review

Symbol table review.

- Records with keys.
- INSERT.
- SEARCH.
- Balanced trees use log N key comparisons.
- Hashing uses O(1) probes but probe proportional to key length.

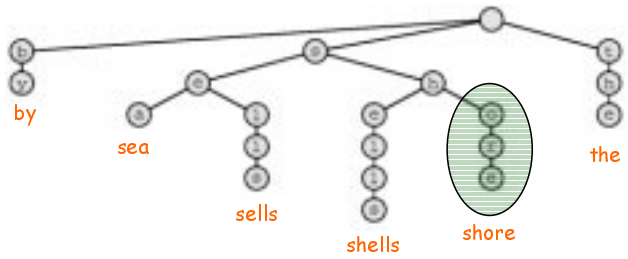
Are key comparisons necessary? No.
Is time proportional to key length required? No.
Best possible. Examine lg N BITS.

Tries

Tries.

- Store characters in internal nodes, not keys.
- Store records in external nodes.
- Use the characters of the key to guide the search ala radix sort.
- NB: from reTRIEval, but pronounced "try."
- You can get at anything if its organized properly in 40 or 100 bits!

Example: sells sea shells by the sea shore



Applications

Modern application: inverted index of Web.

- Insert each word of every web page into trie, storing URL list in leaves.
- Find query keywords in trie, and take intersection of URL lists.
- Use Pagerank algorithm to rank resulting web pages.

More applications.

- Princeton U-CALL.
- Routing tables for IP addresses.
- Storing and querying XML documents.
- Computational biology.
- Data compression. stay tuned
- Associative arrays, associative indexing.

Existence Symbol Table: Operations

Set of Keys (no auxiliary data).

Full set of operations.

- Create.
- Destroy.

generic ops for ADT

- Insert.
- Exists.

ops that characterize existence symbol table

- Count.
- Delete.
- Join.
- Sort.
- Find kth largest.

other ops that many clients need

```
Key k;
STinit();
while(KEYscan(&k) == 1) {
   if (!STsearch(k)) {
      STinsert(k);
      KEYshow(k);
   }
}
```

Existence ST client that removes duplicates from input stream

Key: Operations

Key consists of an array of digits.

- String: '\0'-terminated sequence of characters.
- Bitstring: sequence of 0s and 1s.
- Credit card number: sequence of 16 decimal digits.

key.h (C strings)

```
#define R 256           // extended ASCII
#define NULLdigit '\0'  // string termination char

typedef char Digit;     // each digit is a character
typedef Digit *Key;     // Key is a sequence of digits

int  eq(Key, Key);      // are keys equal?
int  less(Key, Key);    // is first key less than second?

void KEYshow(Key);      // display key
int  KEYscan(Key *);    // read in a key
void KEYfree(Key);      // free memory
Key  KEYcopy(Key);      // copy
```


Existence Symbol Table: Implementations Cost Summary

	Typical Case			Dedup	
Implementation	Search hit	Insert	Space	Moby	Actors
Input *	L	L	L	0.26	15.1
Red-Black	L + log N	log N	C	1.40	97.4
Hashing	L	L	C	0.76	40.6

Actor: 82MB, 11.4M words, 900K distinct.
Moby: 1.2MB, 210K words, 32K distinct.

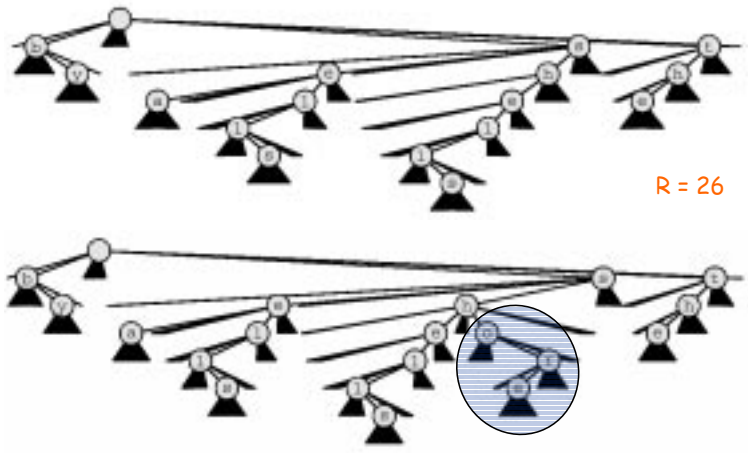
N = number of strings.
L = size of string.
C = number of characters in input.
R = radix.

* only reads in data.

Challenge: As fast as hashing, as flexible as BST.

R-Way Existence Trie: Example

Example. sells sea shells by the sea shore



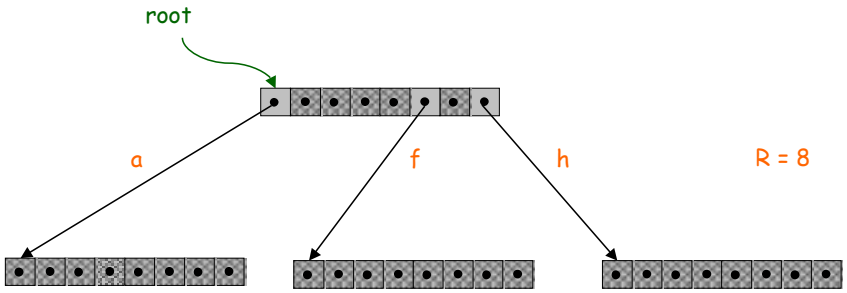
R-Way Existence Trie: Implementation

R-way existence trie: a link.
Link: pointer to a node.
Node: struct of R links.

rway-existence.c

```
typedef struct STnode* link;
struct STnode {
   link next[R];
};

static link root;
```



R-Way Existence Trie: Implementation

Code is short and sweet.

rway-existence.c (Sedgewick Program 15.7)

```
int searchR(link x, Key k, int i) {
   Digit d = k[i];                  // ith character
   if (x == NULL) return 0;         // not found
   if (d == NULLdigit && x->next[NULLdigit]) return 1;
   return searchR(x->next[d], k, i+1);
}

link insertR(link x, Key k, int i) {
   Digit d = k[i];                  // ith character
   if (x == NULL) x = NEWnode();    // add node
   if (d == NULLdigit && !x->next[NULLdigit])
      x->next[NULLdigit] = NEWnode();
   if (d == NULLdigit) return x;
   x->next[d] = insertR(x->next[d], k, i+1);
   return x;
}
```


Existence Symbol Table: Implementations Cost Summary

	Typical Case			Dedup	
Implementation	Search hit	Insert	Space	Moby	Actors
Input	L	L	L	0.26	15.1
Red-Black	L + log N	log N	C	1.40	97.4
Hashing	L	L	C	0.76	40.6
R-Way Trie	L	L	R N + C	1.12	Crash

R = 256

R-way trie: Faster than hashing for small R, but slow and wastes memory if R is large.

Goal: Use less space.

Correspondence With Sorting Algorithms

BSTs correspond to quicksort recursive partitioning structure.

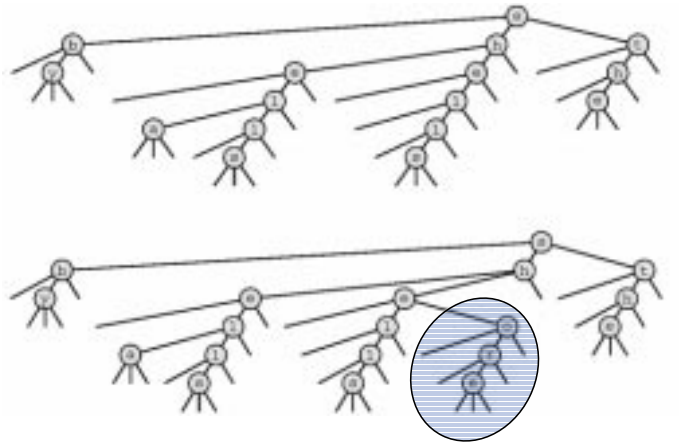
R-way tries correspond to MSD radix sort.

Q. What corresponds to 3-way radix quicksort?

Existence TST: Example

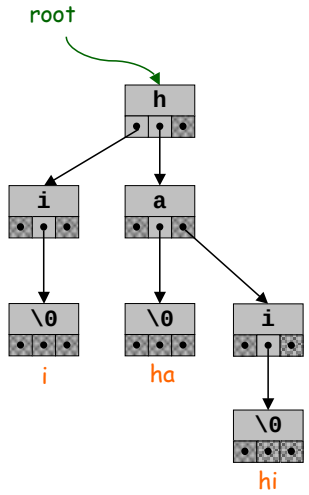
Ternary search tree example: sells sea shells by the sea shore

Observation: Few wasted links!



Existence TST: Implementation

Existence TST: a link.
Link: pointer to a node.
Node: struct of four fields
- digit d
- left link (TST with smaller keys)
- middle link (TST with equal keys)
- right link (TST with larger keys)



tst-existence.c

```
typedef struct STnode *link;
struct STnode {
    Digit d;
    link l, m, r;
};

static link root;
```


Existence TST: Implementation

tst-existence.c (Sedgewick Program 15.8)

```
int searchR(link x, Key k, int i) {
    Digit d = k[i];                  // ith character
    if (x == NULL) return 0;         // not found
    if (d == NULLdigit && x->d == NULLdigit) return 1;
    if      (d < x->d) return searchR(x->l, k, i);
    else if (d > x->d) return searchR(x->r, k, i);
    else               return searchR(x->m, k, i+1);
}
int STsearch(Key k) { return searchR(root, k, 0); }

link insertR(link x, Key k, int i) {
    Digit d = k[i];                  // ith character
    if (x == NULL) x = NEWnode(d);   // add node
    if (d == NULLdigit && x->d == NULLdigit) return x;
    if      (d < x->d) x->l = insertR(x->l, k, i);
    else if (d > x->d) x->r = insertR(x->r, k, i);
    else if            x->m = insertR(x->m, k, i+1);
    return x;
}
void STinsert(Key k)  { root = insertR(root, k, 0); }
```


Existence Symbol Table: Implementations Cost Summary

	Typical Case			Dedup	
Implementation	Search hit	Insert	Space	Moby	Actors
Input *	L	L	L	0.26	15.1
Red-Black	L + log N	log N	C	1.40	97.4
Hashing	L	L	C	0.76	40.6
R-Way Trie	L	L	R N + C	1.12	Memory
TST	L + log N	L + log N	C	0.72	38.7

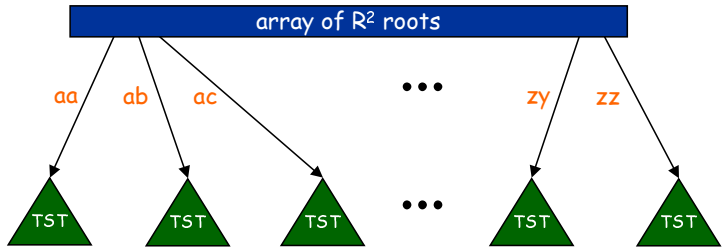
no arithmetic

TST: As fast as hashing, as flexible as BST.

Existence TST With R^2 Branching At Root

Hybrid of R-way and TST.

- Do R-way or R^2-way branching at root.
- Each of R^2 root nodes points to a TST.



Q. What about one letter words?

Existence Symbol Table: Implementations Cost Summary

	Typical Case			Dedup	
Implementation	Search hit	Insert	Space	Moby	Actors
Input	L	L	L	0.26	15.1
Red-Black	L + log N	log N	C	1.40	97.4
Hashing	L	L	C	0.76	40.6
R-Way Trie	L	L	R N + C	1.12	Memory
TST	L + log N	L + log N	C	0.72	38.7
TST with R^2	L + log N	L + log N	C	0.51	32.7

no arithmetic

Result: Faster than hashing, as flexible as BST.

Existence TST Summary

Advantages.

- Very fast search hits.
- Search misses even faster. examine only a few digits of the key!
- Linear space.
- Adapts gracefully to irregularities in keys.
- Supports even more general symbol table ops.

Bottom line: Faster than hashing and even more flexible.

TST: Partial Matches

TST search with wildcards.

- `co....er`
- `.c..nce`

Code writes itself!

- If query digit is '.' OR if it's less than current digit go LEFT.
- If query digit is '.' OR if it's equal to current digit go MIDDLE.
- If query digit is '.' OR if it's greater than current digit go RIGHT.
- Maintain path in an array or queue.

TST: Partial Matches

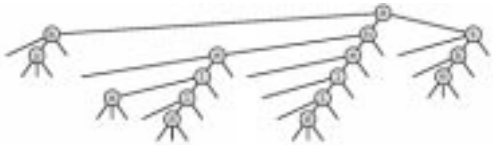
tst-existence.c (Sedgewick Program 15.9)

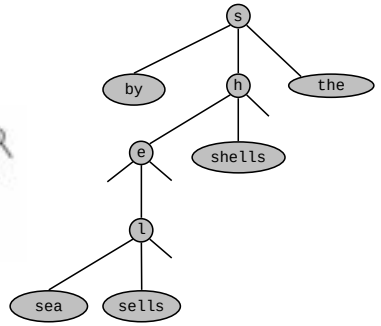
```
void matchR(link x, char *k, int i) {
    static char word[MAXK + 1];
    char d = k[i];                  // ith character
    if (x == NULL) return;
    if (d == '\0' && x->d == '\0') {
       word[i] = d;
       printf("%s\n", word);        // print all matches
    }
    if (d == x->d || d == '.') {
       word[i] = d;
       matchR(x->m, k, i+1);
    }
    if (d  < x->d || d == '.') matchR(x->l, k, i);
    if (d  > d->d || d == '.') matchR(x->r, k, i);
}
void STmatch(char *k) { return matchR(root, k, 0); }
```


TST Symbol Table

TST implementation of symbol table ADT.

- Store Items in leaves of trie.
- Search hit ends at leaf with Key;
  search miss ends at NULL or leaf with different Key.
- Internal nodes store characters; external nodes store Items.
  - use separate internal and external nodes?
  - collapse (and split) 1-way branches at bottom?

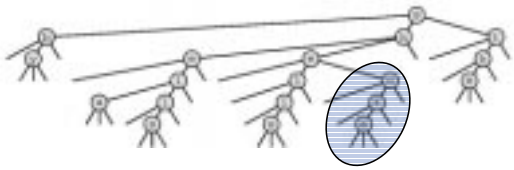


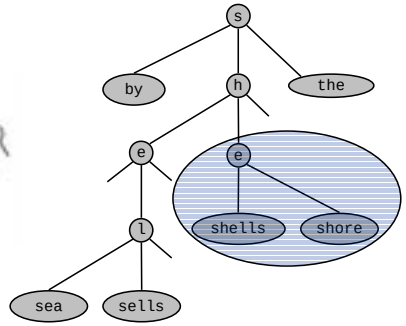


TST Symbol Table

TST implementation of symbol table ADT.

- Store Items in leaves of trie.
- Search hit ends at leaf with Key;
  search miss ends at NULL or leaf with different Key.
- Internal nodes store characters; external nodes store Items.
  - use separate internal and external nodes?
  - collapse (and split) 1-way branches at bottom?





Existence Symbol Table: Implementations Cost Summary

	Typical Case		
Implementation	Search hit	Insert	Space
Input	L	L	L
Red-Black	L + log N	log N	C
Hashing	L	L	C
R-Way Trie	L	L	R N + C
TST	L + log N	L + log N	C
TST with R^2	L + log N	L + log N	C
R-way collapse 1-way	$\log_R N$	$\log_R N$	RN + C
TST collapse 1-way	log N	log N	C

Search, insert time is independent of key length!

- Can use with very long keys.

Associative Arrays

Associative array.

- In C, arrays indexed by integers.
- In Perl, JavaScript, csh, PHP, Python, . . .
  - president["Princeton"] = "Tilghman"

```
# collect data
foreach student ($argv)
   foreach input (input100.txt input1000.txt input10000.txt)
      foreach program (worstfit bestfit)
         t[$student][$input][$program] = ‘time $program < $input‘
      end
   end
end

# compute statistics
. . .
```

Idealized excerpt from COS 226 timing script

Associative Indexing

Associative index.

- Given list of N strings, want to associate an index between 0 and N-1 with each string.
- Recall union find where we assumed objects were labeled 0 to N-1.

Why useful?

- Running algorithm with indices (instead of ST lookup) is faster.
- No need to modify Item type – add index field to `struct Stnode`.

Associative Indexing: Application

Connectivity problem.

- N objects: 0 to N-1
- Find: is there a connection between A and B?
- Union: add a connection between A and B.

Fun version. (see Assignment 8)

- N objects: "Kevin Bacon", "Kate Hudson", . . .
- Find: is there a chain of movies connecting Kevin to Kate?
- Union: Kevin and Kate appeared in "How To Lose a Guy in 10 Days" together, add connection

Real version.

- N objects: "www.cs.princeton.edu", "www.harvard.edu"

Associative Indexing: Application

```
while(1) {
   scanf("%d", &p);
   scanf("%d", &q);
   . . .
}
```

Integer input pairs

```
while (1) {
   KEYscan(&pkey);
   p = STindex(pkey);              // get index
   if (p == -1) {                  // not found
      p = STinsert(pkey);          // insert it
      names[p] = KEYcopy(pkey);    // copy to array
   }

   KEYscan(&qkey);
   . . .
}
```

String input pairs

Symbol Table Summary

Binary search trees.

- Randomized.
- Red-black.

Hash tables.

- Separate chaining.
- Linear probing.

Tries.

- R-way.
- TST.

Determine the ST ops for your application, and choose the best one.