`http://www.cs.unh.edu/~ruml/cs758`

Tries

Problem Statement

Given a linked list of items, print all possible subsets.

running time?

Searching

Structure	Find	Insert	Delete
List (unsorted)			
List (sorted)			
Array (unsorted)			
Array (sorted)			
Heap			
Hash table			
Binary tree (unbalanced)			
Binary tree (balanced)			

Searching

What about long keys?

Can we detect miss without examining entire key?

trie: test digits of key, branching on values

- some nodes do not hold values
- fixed order
- depth = length
- canonical representation

re<u>trie</u>val

CLRS: ‘trie’ = ‘radix tree’
Wikipedia: ‘trie’ $\neq$ ‘radix tree’
Sedgewick: ‘trie’ $\neq$ ‘digital search tree’

duplicate keys?

what’s their weakness?

Not Tries

Wikipedia 'radix tree' = 'radix trie' = 'patricia trie': compressed trie, every internal node has at least two leaves beneath

Sedgewick: 'digital search tree': value at every node, just like binary trees except test bits instead of full compare

Searching

Structure	Find	Insert	Delete
List (unsorted)			
List (sorted)			
Array (unsorted)			
Array (sorted)			
Heap			
Hash table			
Binary tree (unbalanced)			
Binary tree (balanced)			
Trie			

Problem Statement

Given a list of records, which may contain duplicates, return a list containing each record at most once.

Break

- asst 4
- asst 5

Algorithms

Algorithms

Beyond craftsmanship lies invention, and it is here that lean, spare, fast programs are born. Almost always these are the result of strategic breakthrough rather than tactical cleverness. Sometimes the strategic breakthrough will be a new algorithm, such the Cooley-Tukey Fast Fourier transform or the substitution of an $n \log n$ sort for an n^2 set of comparisons.

Much more often, strategic breakthrough will come from redoing the representation of the data or tables. This is where the heart of a program lies. Show me your flowchart and conceal your tables, and I shall continue to be mystified. Show me your tables, and I won't usually need your flowchart; it'll be obvious.

— Fred Brooks, 1974 (lead on IBM System/360, Turing Award 1999)

Algorithms: Modern Version

Smart data structures and dumb code works a lot better than the other way around.

— Guy Steele, 2002 (ACM Fellow, inventor of Scheme, editor of *The Hacker's Dictionary*)

Types of Algorithms

- divide and conquer
- dynamic programming
- greedy
- backtracking
- (reduction)

Dynamic Programming

Fibonacci Numbers

$$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{for } n \geq 2 \end{cases}$$

What is the complexity of the naive algorithm?
How to make this efficient?

Memoization

- recursive decomposition
- polynomial number of subproblems
- cache results in look-up table

one form of *dynamic programming*

EOLQs

- What's still confusing?
- What question didn't you get to ask today?
- What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.
Thanks!