All Words over the alphabet \{a,b,...z\}.
In the slides, let say that the alphabet is only \{a,b,c,d\}
\(S\) – set of words =\{a,aba, a,aca, addd\}
Need to support the operations

- \(\text{insert}(w)\) – add a new word \(w\) to \(S\).
- \(\text{delete}(w)\) – delete the word \(w\) from \(S\).
- \(\text{find}(w)\) is \(w\) in \(S\) ?

The time for each operation should be \(O(k)\), where \(k\) is
the number of letters in \(w\).

- Usually each word is associated with addition info –
  not discussed here.

---

**Tries - intro**

Alon Efrat

Computer Science Department

University of Arizona

---

**A data-structure for a set of words**

---

**Trie (Tree+Retrve) for \(S\)**

- A tree where each node is a struct consists
- struct node {
  
  - Struct node = ar[4];
  
  - char flag; /* 1 if a word ends at this node. Otherwise 0 */

---

**A trie - example**

\(p>ar["d",a]\)

The label of an edge is the label of the cell from which the edge exits

\(S=\{a,b,dbb\}\) 

This node corresponds to \(dbb\) (not in \(S\), since
flag=0) 

Corresponding to \(dbb\)
Finding if string $s$ in the tree

$p$=root; $i$=0
While(1){
  ■ If $s[i]$ == '\0' then return the flag of $p$;
  ■ If the entry of $p$ correspond to $s[i]$ is NULL return false;
  ■ Set $p$ to be the node pointed by this entry, and set $i$++;
}

Inserting string $s$

■ Try to perform find. If runs into NULL pointers, create new nodes along the way.
■ The flag fields of all new nodes is 0.
■ Set to 1 the flag of the node corresponding to $s$.

Deleting a string $s$

■ Find the node $p$ corresponding to $s$.
■ Set the flag field of $p$ to 0.
■ if $p$ is dead (i.e. flag==0 and all pointers are NULL ) then free($p$), set $p$=parent($p$) and repeat this check.