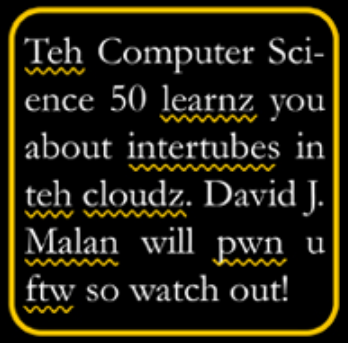


CS 50 Walkthrough 6

Problem Set 6: Mispellings

Marta Bralic

Slides courtesy of: Keito Uchiyama

Problem Set 6: Mispellings

- Topics:
  - More data structures, more pointers
  - More File I/O
- You implement:
  - A dictionary for a very fast spell checker

The Distribution Code

- texts – a symlink
- speller.c – a spellchecker
- dictionary.h – the header file
- dictionary.c – a dictionary implementation
- Makefile
- questions.txt

What to implement

- **load**() – loads a dictionary into memory
- **size**() – gets the size of the dictionary
- **unload**() – unloads a dictionary from memory
- **check**() – checks if a given word is in the dictionary

Your options

- Slow but simple: Linear search every time
  - don't do this!
- Hash tables
- Tries

Hash Tables

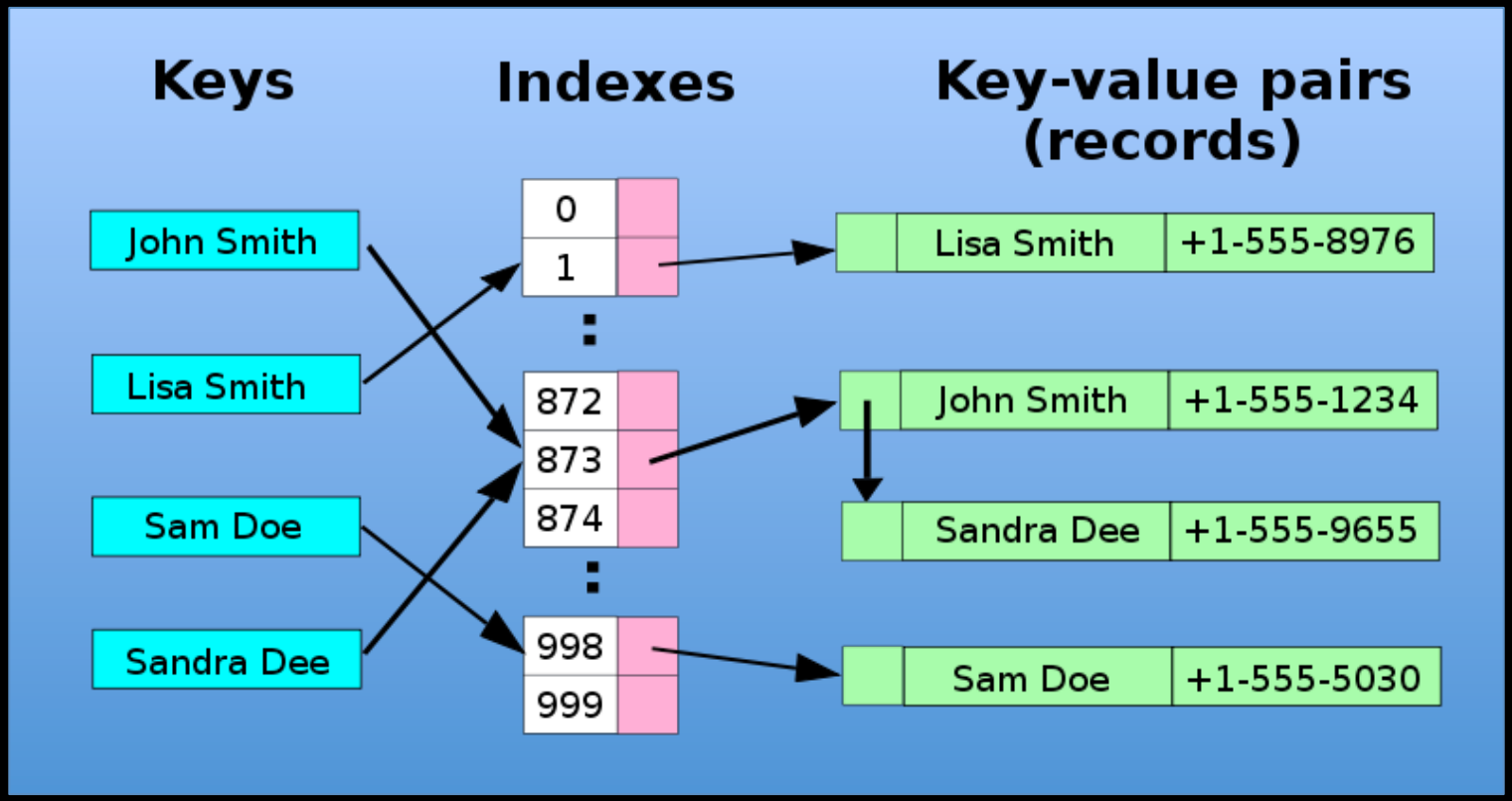


Image courtesy User:Davidgothberg and User:Helix84, Wikimedia Commons

Hash Tables - Operations

- Initializing our hash table
- Adding dictionary words
- Checking words
- Unloading words

Initializing

```
//Here's how our node is defined
typedef struct node {
    char word[LENGTH + 1];
    struct node *next;
} node;

//We have our main directory of node
//pointers
node *myarray[ARRAYSIZE];

// for each element i in myarray:
//     myarray[i] = NULL
```

Hash Tables – a Hash Function

```
function myHashFunction(string):
   int hashresult

   foreach character in string:
      hashresult += character - 65

   return hashresult % ARRAYSIZE
```

Loading Dictionary Words

- fopen(dict) same as in pset5
- while !feof(dict)
  - create nodes for them
  - put these nodes in the hash table

Creating Nodes

- malloc space for new node (node *newnode)
  - store each letter i of the word in that node
    - fgetc(dptr) is that letter
    - newnode->word[i] is where letters should be stored
    - until you reach '\n'
      - newnode->word[j] = '\0' at this point

Put Node in Hash Table

- hash(newnode->word)
  - go to that place in array (array[hashresult])
    - if nothing is there (NULL)
      - put a pointer to your node that you just malloced there
      - set newnode->next to NULL
    - else
      - set newnode->next to the pointer currently there
      - put your pointer there
- when while loop exits, fclose(dict)

Size

Really easy if you've kept a counter that you increment every time you load a word.

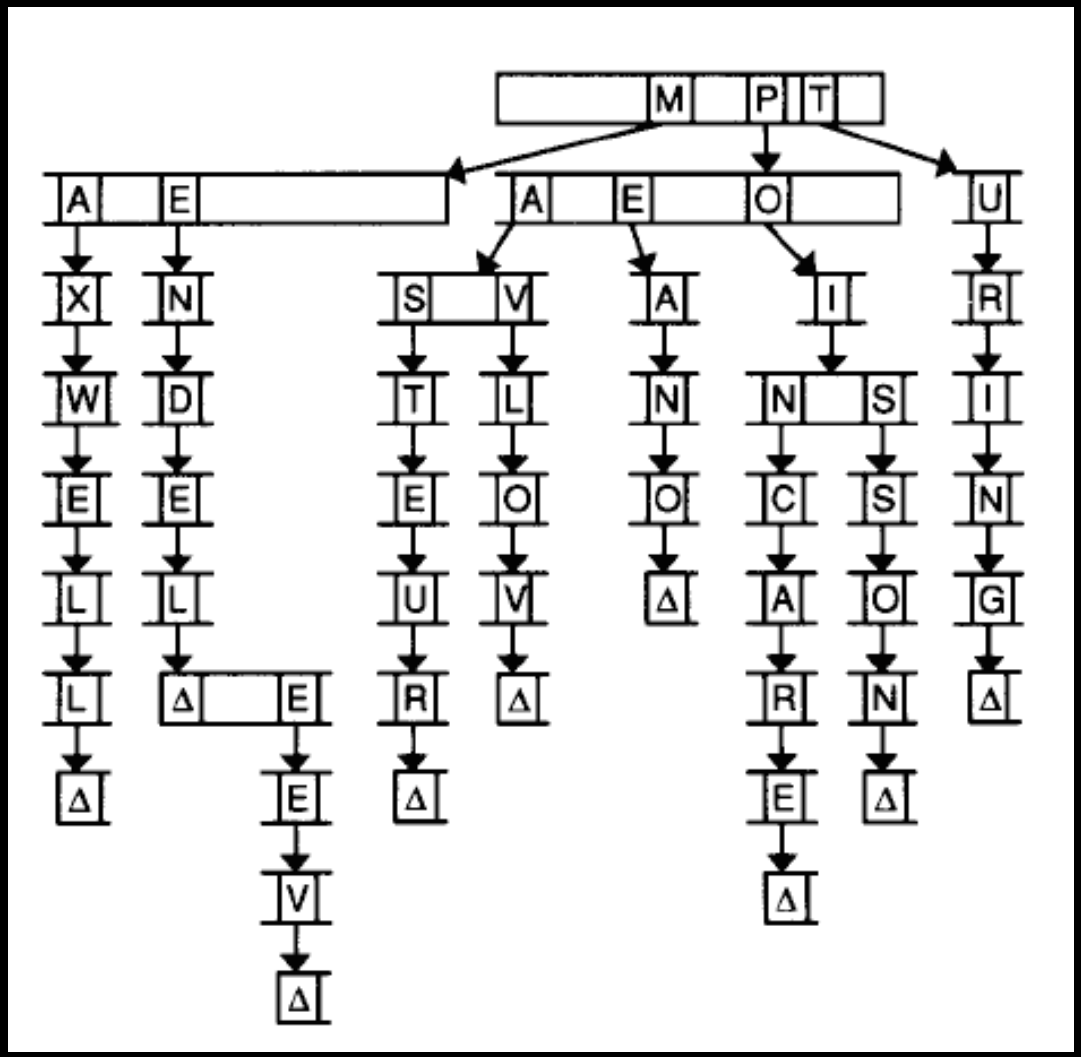
Check

- convert each letter of word tolower
- hash word and go to that place in array
  - temporarily store the ptr you find there (currnode)
    - while (currnode != NULL)
      - compare currnode->word to word
        - how?
        - if match, return true
      - if not, move to currnode->next
    - when you exit this while loop, return false

Unload

- Iterate through each node, like in check
  - free the node
  - free the spot in the array that starts the linked list
  - return true
- run valgrind to ensure no leaks!

Tries


M P T
A E
A E O
U
X N
S V
A
I
R
W D
T L
N
N S
I
E E
E O
O
C S
N
L L
U V
Δ
A O
G
L Δ E
R Δ
R N Δ
Δ E
E Δ
Δ
V
Δ

Tries – A struct

```
//Here's an example of what each node
//in our struct will look like
typedef struct node {
    bool is_word;
    struct node *children[27];
} node;

//We have our root node
node *root;
```

Tries - Operations

- Initializing our root node
- Adding words
- Checking words
- Unloading words

Tips

- Start with a small dictionary and small text file (speller [dict] file)
- Mapping out data structures on paper and on screen
- Using gdb

CS 50 Walkthrough 6

Problem Set 6: Mispellings

Marta Bralic