CS 50 Walkthrough 6

Problem Set 6: Mispellings

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 (or something)

- **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
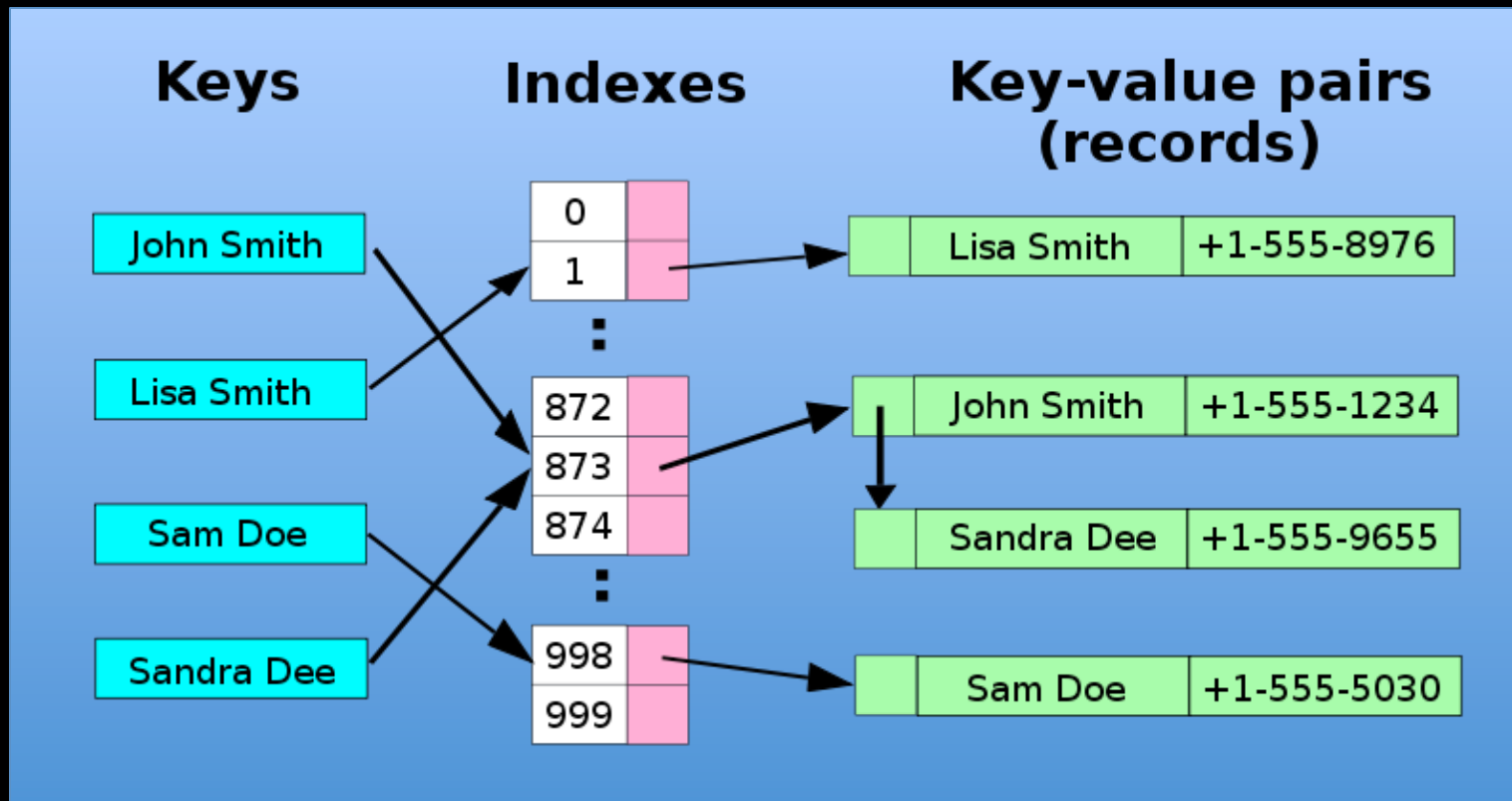- Hash tables
- Tries
- ☺

Hash Tables

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

Hash Tables – A struct

```c
//Here's how our node is defined
typedef struct node {
    char word[46];
    struct node *next;
} node;

//We have our main directory of nodes
node *myarray[ARRAYSIZE];
```
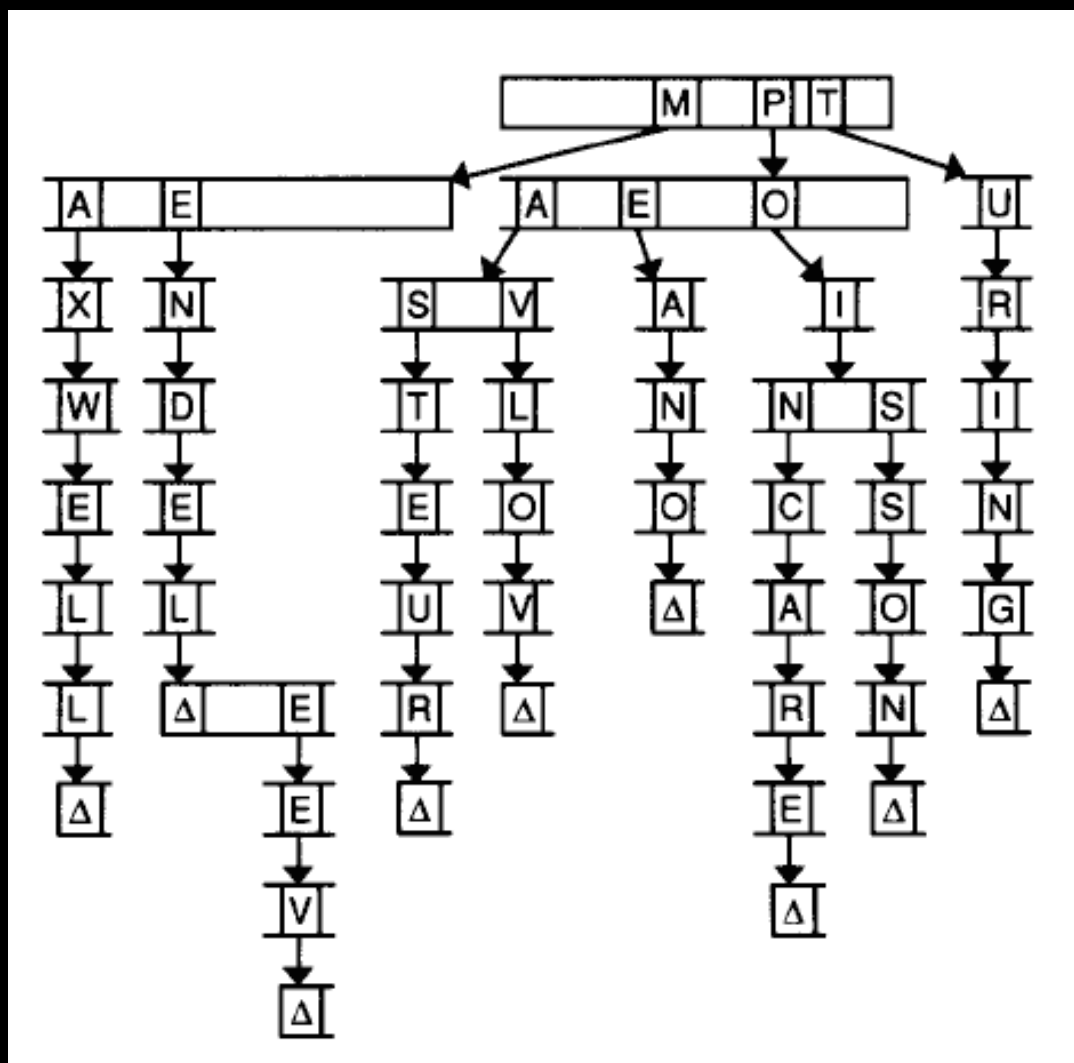
Hash Tables – a Hash Function

```
function myHashFunction(string):
    int hashresult

    foreach character in string:
        hashresult += character - 65

    return hashresult % ARRAYSIZE
```

Hash Tables – Operations

- Initializing our hash table

- Adding words

- Checking words

- Unloading words

Tries

Tries — A struct

```c
//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

Teh Computer Science 50 learnz you about intertubes in teh cloudz. David J. Malan will pwn u ftw so watch out!

CS 50 Walkthrough 6

Problem Set 6: Mispellings

Keito Uchiyama