ORF 201

COMPUTER METHODS FOR PROBLEM SOLVING

Lecture 13

Linked Lists (Jotto)

"The secret to happiness is: *Always get as much RAM as you can possibly afford.*"

http://www.princeton.edu/~orf201

The Game Jotto

Pick a 5-letter word: *teach*

Your opponent queries 5-letter words.
To each, you respond with number of "matches":

Query	Matches
baste	3
ledge	1
abode	2
tepid	2
metro	2
lathe	4

Game ends when number of matches is 5. That is, when query either matches hidden word exactly or is an anagram of it.

Humans Opponents Use Logic

Query	Matches
baste	3
ledge	1
abode	2
tepid	2
metro	2
lathe	4

Let's see, *lathe* has 4 matches. Therefore, *a* and *e* are either both in or only one of them is in. Suppose first that both are in. Then, from *baste*, *b*, *s*, and *t* are out. From *ledge*, *l*, *d*, and *g* are out. From *abode*, *o* is out too. Etc.

Now, suppose that *a* is in but *e* is out. Then from lathe, *l*, *t*, and *h* are out too. Etc.

Computer Algorithm for Jotto

(1) Read a list of all 5-letter words (there are 2415).
(2) Pick a word at random from the list as the first query.
(3) After being told the number of matches, strike from the list every word that fails to produce that response.
(4) Go back to step 2.

Here's a sample output from computer Jotto:

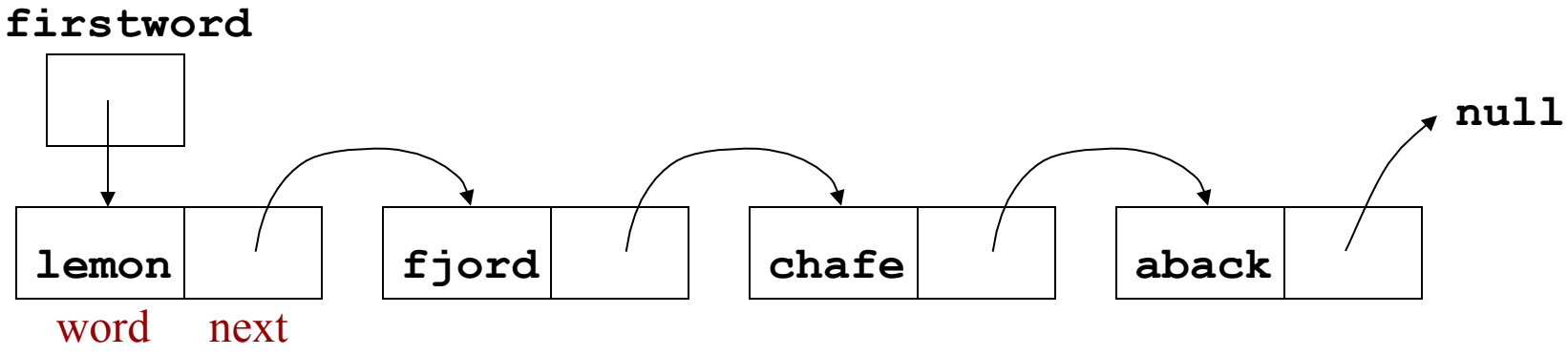
```
Think of a 5-letter word
Are you ready to begin (y/n)? y

There are 2415 words in my dictionary.
My 1-th guess is:  gorge
How many matches?  1

There are 1046 words in my dictionary.
My 2-th guess is:  lathe
How many matches?  4

There are 39 words in my dictionary.
My 3-th guess is:  slate
How many matches?  3

There are 16 words in my dictionary.
My 4-th guess is:  ethyl
How many matches?  3
```

Main

```
public static void main(String[] args)
{
    int m;
    String guess;
    readDictionary();
    System.out.println("Think of a 5-letter word");

    for (int k=1;  ; k++) {
        System.out.println();
        System.out.println("There are " +n+
                           " words in my dictionary");
        if (n==0) {
            System.out.println("Sorry, your word is "+
                               "not in my dictionary");
            break;
        }
        guess = pickRandom();
        System.out.println("My "+k+"-th guess is: "+guess);
        m = Console.in.readInt("How many matches? ");
        if (m==5) {
            System.out.println("I win, I win!");
            break;
        }
        if (m>5 || m<0) {
           System.out.println("No Way!  Try again.");
           continue;
        }
        removeWords(m, guess);
    }
}
```

Store Dictionary in a Linked List

```
class WordLink
{
     String word;
     WordLink next;
}

public class Jotto
{
     /* Class variables */

     static int n;                    // size of dictionary
     static WordLink firstWord;       // Pointer to first word
                                      // in linked list
      .
      .
      .
}
```



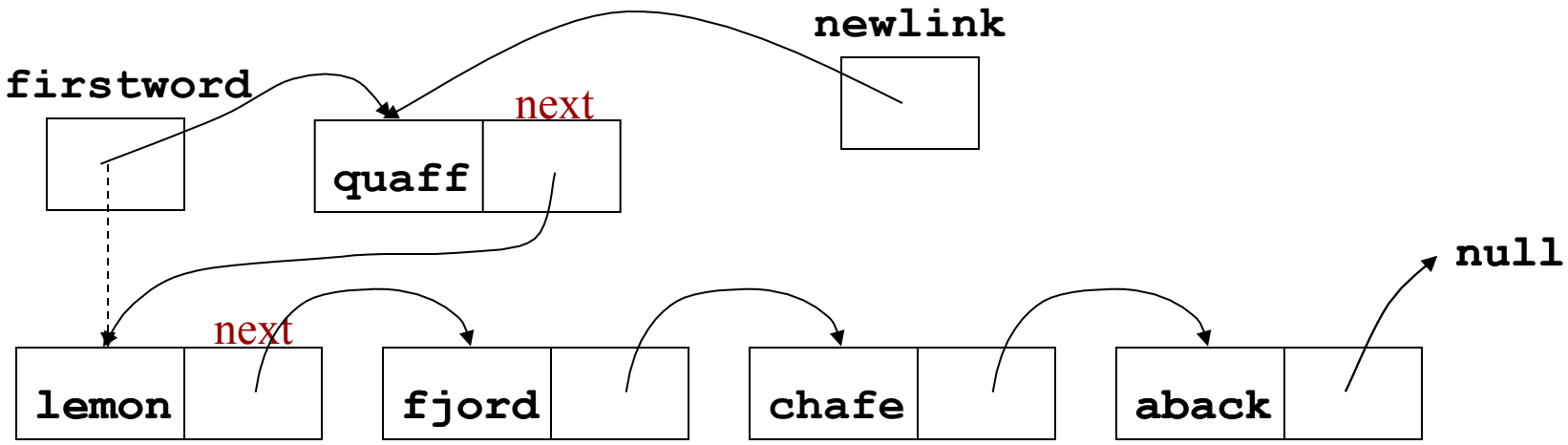
readDictionary ()

```
static void readDictionary()
{
    BufferedReader in;
    try{
        in = new BufferedReader(new FileReader("words5"));
    }
    catch (FileNotFoundException fe) {
        System.err.println("File not found");
        return;
    }

    String word;

    firstWord = null;
    n = 0;
    while (true) {
        word = FileIO.readString(in);
        if (word == null) {break;}    // End of file
        addWord(word);
    }
}
```

null is a Java language word.
It is a pointer to nowhere.

addWord()

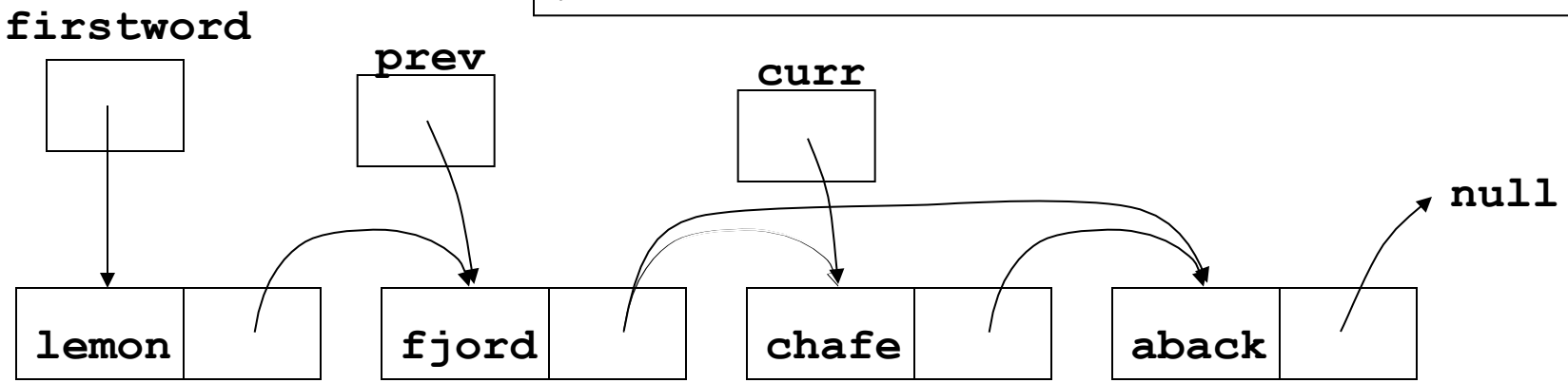
```
static void addWord(String word)
{
     WordLink newlink;

     newlink = new WordLink();
     newlink.word = word;
     newlink.next = firstWord;
     firstWord = newlink;
     n++;
}
```



removeWords() and deleteNextWord()

```
static void removeWords(int m, String guess)
{
    WordLink curr, prev;

    prev = null;
    for (curr = firstWord; curr != null; curr = curr.next) {
        if nmatches(guess, curr.word) != m) {
            deleteNextWord(prev);
        } else {
            prev = curr;
        }
    }
}
```

```
static void deleteNextWord(WordLink prev)
{
    if (prev == null) {
        firstWord = firstWord.next;
    } else {
        prev.next = prev.next.next;
    }
    n--;
}
```



pickRandom()

```
static String pickRandom()
{
    WordLink curr;
    int j = (int) (n*Math.random());

    for (curr=firstWord; curr!=null; curr=curr.next) {
        if (j<=0) break;
        j--;
    }

    return curr.word;
}
```

Note the **`for`** loop paradigm for linked lists.

nmatches()

```
static int nmatches(String s1, String s2)
{
    int i, j;
    int len1 = s1.length();
    int len2 = s2.length();
    int cnt = 0;
    char[] c1 = new char[len1];
    char[] c2 = new char[len2];
    int[]   f = new int[len2];

    s1.getChars(0, len1, c1, 0);
    s2.getChars(0, len2, c2, 0);

    for (j=0; j<len2; j++) { f[j]=0; }

    for (i=0; i<len1; i++) {
        for (j=0; j<len2; j++) {
            if (c2[i] == c1[j] && f[j] == 0) {
                f[j] = 1;
                break;  // out of the for(j...) loop
            }
        }
    }
    for (j=0; j<len2; j+) {cnt += f[j]; }
    return cnt;
}
```

Imports:

```
import java.util.*;
import java.io.*;
import java.lang.*;
import myutil.*;
import ccj.*;
```

Complexity

	Searching	*Adding*	*Deleting*	*Access*
Unsorted Array	n	1*	1	1
Sorted Array	$\log(n)$	n*	n	1
Linked List	n	1	1	n

* Assuming the array is big enough.