

## Problem Set 4

Due: 1:30pm Tuesday, April 10

This is the final version of PS4. Note that Problem 2 is now posted on the web at <http://cs.wellesley.edu/~cs230/ps4-color-game>. The problem set is now due at 1:30 on Tuesday, April 10.

### Overview:

The purpose of this assignment is to give you practice with (1) using queues, tables, and sets in a simple web-crawling scenario and (2) building a GUI from scratch. As usual, it is recommended that you (1) start early and (2) work with a partner (a different one than you worked with on PS1 – PS3).

### Submission:

Each team should turn in a single hardcopy submission packet for all problems by slipping it under Lyn's office door by 6pm on the due date. The packet should include:

1. a team header sheet (see the end of this assignment for the header sheet) indicating the time that you (and your partner, if you are working with one) spent on the parts of the assignment.
2. your final version of `WebCrawler.java` from Problem 1.
3. a transcript of your `WebCrawler` test cases for Problem 1.
4. your final version of `GuessTheColor.java` from Problem 2.

Each team should also submit a single softcopy (consisting of your entire final `ps4` directory) to the drop directory `~cs230/drop/ps4/username`, where `username` is the username of one of the team members (indicate which drop folder you used on your hardcopy header sheet). To do this, execute the following Linux commands in the team member's account where the code is stored:

```
cd /students/username/cs230
cp -R ps4 ~cs230/drop/ps4/username/
```

### Problem 1 [35]: A Web Crawler

#### Background

Modern web search engines like Google have sophisticated web page indexing mechanisms that make it possible to quickly list web pages containing a given set of keywords. Conceptually, there are two distinct phases to web page indexing:

1. An **indexing** phase in which a very large database (think table) is constructed by processing all the words in each of the web pages that is indexed.
2. A **query** phase in which users submit keyword queries and some subset of the web pages in the database matching these queries (hopefully the most relevant ones) are found and displayed.

Since web pages are constantly changing and new web pages are being published all the time, the indexing phase is an ongoing process that is interleaved with the query phase. The indexing phase is largely carried out by automated agents variously known as **web crawlers**, **web spiders**, and **web robots**. After indexing the contents of a page, these agents follow one or more links mentioned on page and recursively index the pages that are the targets of these links.

In this problem, you will implement an extremely simple web crawler that indexes a subset of the web pages that are reachable from the web pages in a given list. This web crawler is a modified version of the file indexer that you studied in Lab 8. Like the file indexer, the web crawler uses a table to map each word encountered in the indexing process to a set of all the web pages in which it was found. Unlike the file indexer, the web crawler has to follow links from one web page to another.

For example, consider Fig. 1, which shows the links between some sample web pages.<sup>1</sup> From `index.html`, four pages can be reached by one link: `descartes.txt`, `seuss.txt`, `popeye.html`, and `dome.html`. Some of these pages themselves have links that can also be followed.

The diagram in Fig. 1 is an example of a **directed graph**, a structure in which nodes are connected by directed edges. In a directed graph, there may be more than one path from one node to another. For example, in Fig. 1 it is also possible to get from `index.html` to `descartes.txt` by following one link from `index.html` to `popeye.html` and then following another link from `popeye.html` to `descartes.txt`. A directed graph may also contain **cycles** — nonempty paths that go from a node back to itself. For example, the path from `index.html` to `gdome.html` and back to `index.html` is a cycle.

Assume that our web crawler is always visiting a particular page. The basic strategy of the crawler is to index all the words at that page and to add the targets of the links that originate at that page to a **queue of pages to visit**. When the crawler is done indexing a page, it dequeues the next page from the queue and begins to process it similarly.

In order to avoid duplicating work or getting stuck in an infinite loop, we want our web crawler to avoid visiting any page more than once. For this purpose, the crawler will maintain a **set of already-seen pages**. Whenever it encounters a link to another page, the crawler will first check if that page is in the set. If so, it will not add the page to its queue. If not, it will add the page to its queue, and will also add it to the already-seen set.

Because the graph of web pages is so vast, we cannot reasonably expect that our simple crawler will be able to visit all reachable pages. So we will limit it to visit only those pages that are reachable by following no more than some given number of links (called the **maximum distance**) from a starting page. For example, if the crawler starts at the page named `index.html` in Fig. 1, then here are the pages it will visit for some values of the maximum distance:

Maximum Distance	Pages Visited
0	<code>index.html</code>
1	<code>index.html</code> , <code>descartes.txt</code> , <code>seuss.txt</code> , <code>popeye.html</code> , <code>gdome.html</code>
2	<code>index.html</code> , <code>descartes.txt</code> , <code>seuss.txt</code> , <code>popeye.html</code> , <code>gdome.html</code> , <code>~gdome</code>
3	<code>index.html</code> , <code>descartes.txt</code> , <code>seuss.txt</code> , <code>popeye.html</code> , <code>gdome.html</code> , <code>~gdome</code> , <code>cs.wellesley.edu</code> , <code>list.html</code> , <code>table.html</code> , <code>cookies.html</code>

### *Finding Links in a Page*

Within an HTML page, links have the form `<A HREF=link-url>link-text</A>`, where *link-text* is the highlighted text that appears in the browser, and *link-url* is the URL (Uniform Resource Locator, i.e., the address) of the web page that is the target of the link. The URL may either be **absolute**, in which case it specifies the entire address of the page or **relative**, in which case it specifies an address of the page relative to the address of the current page. For example, the URL `http://cs.wellesley.edu/~gdome` in `dome.html` is absolute, while the URL `dome.html` in `index.html` and `./` in `dome.html` are relative URLs. A relative URL can be transformed to an absolute URL by merging its information with the absolute URL of the page in which it occurs. For example, suppose the absolute URL of `index.html` is

<sup>1</sup>The page named `index.html` resides at `http://cs.wellesley.edu/~cs230/amness/index.html`, and all the other pages are reachable from it.

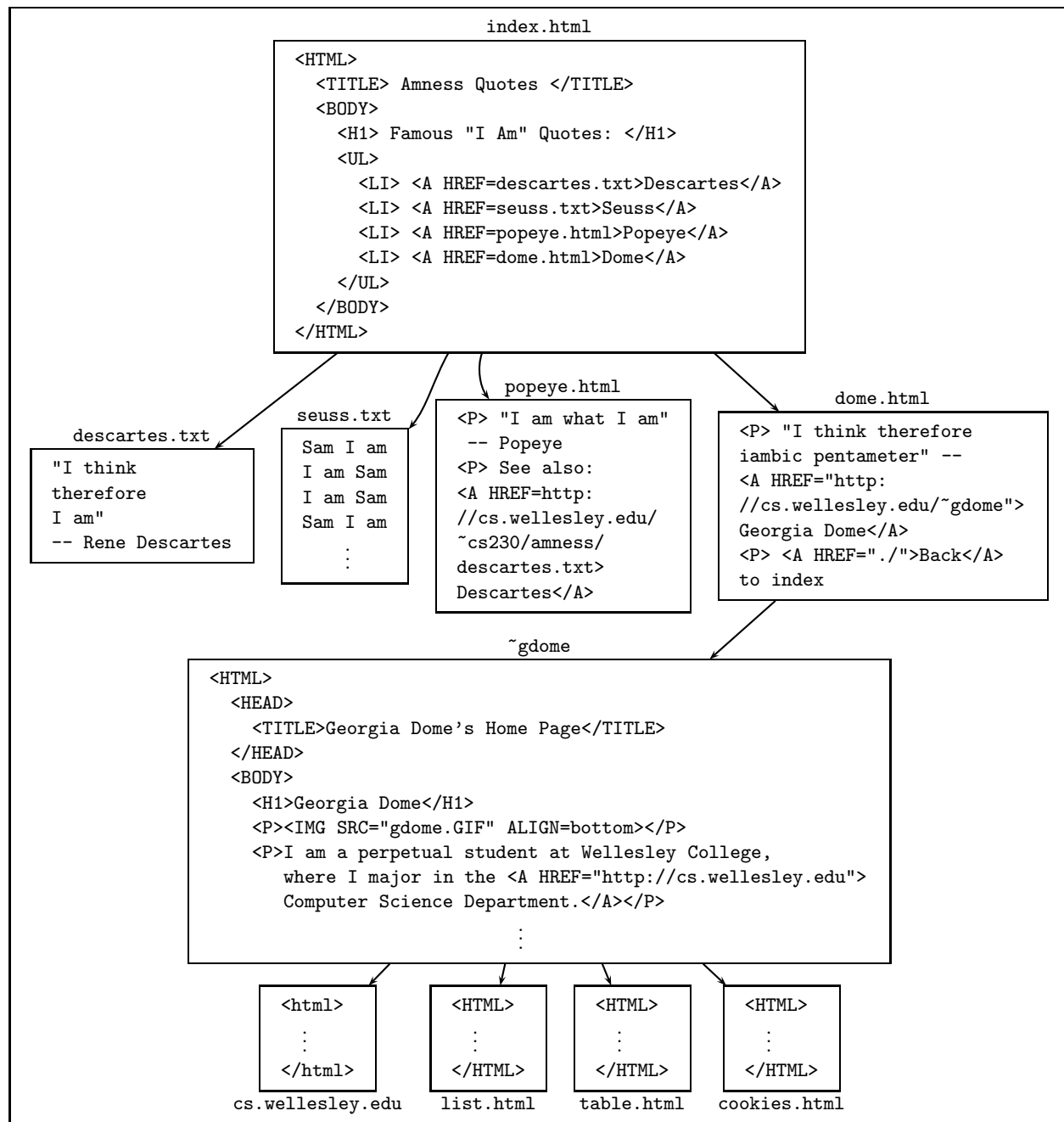


Figure 1: A web page graph.

`http://cs.wellesley.edu/~cs230/amness/index.html`

Then the relative URL `dome.html` in `index.html` stands for the absolute URL

`http://cs.wellesley.edu/~cs230/amness/dome.html`

and the relative URL `./` in `dome.html` stands for the absolute URL

`http://cs.wellesley.edu/~cs230/amness/`

(which itself is a shorthand for `http://cs.wellesley.edu/~cs230/amness/index.html`).

The details of finding absolute URLs for links in a file are handled by the `URLWordsAndLinks` class (which was created for this assignment). This class is an iterator that yields all the words and links in a file. Links are represented as strings of the form `link=url`, where `url` is an absolute URL.

Figs. 3 and 3 show two examples of using `URLWordsAndLinks` to process files. The file `index.html` is found to have four links, while the file `dome.html` is found to have two links.

For more details on the `URLWordsAndLinks` class, see Appendix A.

```
[fturbak@puma web-crawler] java URLWordsAndLinks http://cs.wellesley.edu/~cs230/amness/index.html
0:HTML
1:TITLE
2:Amness
3:Quotes
4:/TITLE
5:BODY
6:H1
7:Famous
8:I
9:Am
10:Quotes
11:/H1
12:UL
13:LI
14:A
15:HREF
16:descartes.txt
17:link=http://cs.wellesley.edu/~cs230/amness/descartes.txt
18:Descartes
19:/A
20:LI
21:A
22:HREF
23:seuss.txt
24:link=http://cs.wellesley.edu/~cs230/amness/seuss.txt
25:Seuss
26:/A
27:LI
28:A
29:HREF
30:popeye.html
31:link=http://cs.wellesley.edu/~cs230/amness/popeye.html
32:Popeye
33:/A
34:LI
35:A
36:HREF
37:dome.html
38:link=http://cs.wellesley.edu/~cs230/amness/dome.html
39:Dome
40:/A
41:/UL
42:/BODY
43:/HTML
[fturbak@puma web-crawler]
```

Figure 2: Using `URLWordsAndLinks` to enumerate the words and links of `index.html`.

### *The Web Crawler in Action*

The web crawler for this assignment is implemented in an application named `WebCrawler`. Invoking this application on a non-negative number  $n$  and a URL indexes all web pages that are at most  $n$  links away from the page at the URL. During the indexing phase, messages are printed whenever a web page is enqueued on the queue of pages to visit and whenever a web page is actually indexed. The indexer indexes lower-case versions of all the words in each page, associating every

```

[fturbak@puma web-crawler] java URLWordsAndLinks http://cs.wellesley.edu/~cs230/amness/dome.html
0:P
1:I
2:think
3:therefore
4:iambic
5:pentameter
6:A
7:HREF
8:http://cs.wellesley.edu/~gdome
9:link=http://cs.wellesley.edu/~gdome
10:Georgia
11:Dome
12:/A
13:P
14:A
15:HREF
16:./
17:link=http://cs.wellesley.edu/~cs230/amness/
18:Back
19:/A
20:to
21:index
[fturbak@puma web-crawler]

```

Figure 3: Using `URLWordsAndLinks` to enumerate the words and links of `dome.html`.

word with a set of all the web pages in which it was found.

After the indexing phase, a read/query/print loop is launched in which the user can type a sequence of words. The program responds with a list of all pages in the index in which all the words were found. The special query `#contents` will display the entire contents of the index.

Examples of the `WebCrawler` application are shown in Appendix B. An executable version of this program can be found in `~cs230/ps4/web-crawler/test`. You are encouraged to experiment with this program before continuing with this problem.

#### *Your Task*

A skeleton of a program implementing the simple web crawler described above is presented in Figs. 4–5. The skeleton contains a complete implementation of the `main` method and the read/query/print loop, as well as a fleshed-out `WebCrawler` constructor method.

Your task is to complete the skeleton by implementing the code that indexes all web pages within `maxDistance` of the starting page. You should do this by fleshing out the `indexPagesStartingAt()` method. In order to make your code more readable, you are encouraged to write helper methods as well. Your methods may use the four instance variables of the `WebCrawler` class:

- `maxDistance`: the maximum number of links from the starting page for a web page to be indexed.
- `indexTable`: a table that associates each word (i.e., lower-case string) encountered in the indexing phase with a set of the URLs of all the web pages in which it was found.
- `urlsSeen`: a set of all the URLs seen already (i.e., the URLs of those pages that have been indexed or are on the queue to be indexed).
- `queue`: a queue of all the URLs to be indexed. The elements of this queue are instances of the `Edge` class (Fig. 6), which pairs a URL (i.e., a string) for a web page with an integer indicating the distance of that page from the starting page.

```

/** A simple web crawler application invoked via java WebCrawler <maxDistance> <URL>,
    where <maxDistance> is a non-negative number and <URL> is the URL of the web page at which
    to start. Indexes all the words in all web pages at most <maxDistance> links from the
    starting page, and then launches a read/query/print loop. In this loop, when the user enters
    a sequence of words, a list of all pages in the index containing all the words is displayed.
    The special input #contents displays all entries in the index table. Entering an empty line
    exits the program. */

import java.util.*;
import java.io.*;
import java.net.*;

public class WebCrawler {

    // Instance Variables
    Table<String,Set<String>> indexTable; // Word index for processed pages
    Set<String> urlsSeen; // Set of URLs already seen (i.e., indexed or queued to be indexed)
    Queue<Edge> queue; // Queue of edges to web pages to be indexed
    int maxDistance; // Maximum distance of a web page from starting page for it to be indexed.

    // Create a web crawler that will go no further than n edges away from a specified page.
    public WebCrawler (int n) {
        // Initialize the instance variables
        maxDistance = n;
        indexTable = new TableVectorSorted<String,Set<String>>();
        urlsSeen = new SetVectorSorted<String>();
        queue = new QueueTwoEndedMList<Edge>();
    }

    // Index words on all pages <= maxDistance edges from given url
    public void indexPagesStartingAt (String url) { // Flesh out this method for Problem 1 }

    // Launch a read/query/print loop for entering queries about indexTable
    public void queryLoop () {
        try{
            String input = FileOps.keyboardToLine("\nquery> ");
            while (! input.equals("")) {
                if (input.equals("#contents")) { // Special directive that displays contents of table
                    displayTableContents();
                } else { // Read words entered by user and find all indexed web pages containing this words.
                    String [] words = input.split("\\s+");
                    processQuery(words);
                }
                input = FileOps.keyboardToLine("\nquery> ");
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
        // Exit the read/query/print loop
        System.out.println("Thank you for using the indexing query system!\n");
    }

    // Display all word/set-of-filename associations in the index table
    private void displayTableContents() {
        Iterator<String> ks = indexTable.keys();
        while (ks.hasNext()) {
            String key = ks.next();
            System.out.println(key + ": " + indexTable.get(key));
        }
    }
}

```

Figure 4: Skeleton for the WebCrawler class, part 1

```

// Display the names of web pages in indexTable containing all the words.
public void processQuery (String [] words) {
    if (words.length >= 1) {
        // Find set of files s containing first keyword. Must copy this set to avoid mutating set
        // in table with subsequent intersections. If the mutations were performed on the original
        // rather than the copy, the set value associated with the first keyword in the table
        // would be mutated -- a very undesirable behavior!
        Set<String> files = lookup(words[0].toLowerCase()).copy();
        // Iteratively intersect s with sets for remaining keywords
        for (int i = 1; i < words.length; i++) {
            files.intersection(lookup(words[i].toLowerCase()));
        }
        // Now s contains the set of all files with all keywords. Display them:
        Iterator<String> iter = files.iterator();
        while (iter.hasNext()) {
            System.out.println(iter.next());
        }
        // Note: could replace the above by IterableOps.display(files);
    }
}

// Helper method that acts like get() but returns an empty set
// for any word that is not in indexTable
private Set<String> lookup (String x) {
    Set<String> result = indexTable.get(x);
    if (result == null) {
        return new SetVectorSorted<String>();
    } else {
        return result;
    }
}

// Display a usage message for this application
public static void usage() {
    System.out.println("usage: java WebCrawler <maxDistance> <url>");
}

// Entry point for the WebCrawler application
public static void main (String [] args) {
    if (args.length >= 2) {
        try {
            int maxDist = Integer.parseInt(args[0]);
            if (maxDist >= 0) {
                WebCrawler crawler = new WebCrawler(maxDist);
                crawler.indexPagesStartingAt(args[1]);
                crawler.queryLoop();
            } else {
                System.out.println("maxDistance must be nonnegative:" + maxDist);
                usage();
            }
        } catch (NumberFormatException ex) {
            System.out.println("maxDistance must be a number!" + args[0]);
            usage();
        }
    } else {
        usage();
    }
}
}

```

Figure 5: Skeleton for the WebCrawler class, part 2

```

/** An instance of the Edge class represents a link to a URL
    that is a specified distance away from another web page. */

class Edge {

    public String url;
    public int distance;

    public Edge (String url, int distance) {
        this.url = url;
        this.distance = distance;
    }

    public String toString () {
        return "Edge[url=" + url + "; distance=" + distance + "];";
    }
}

```

Figure 6: Implementation of the Edge class.

### Notes

1. All the code for this problem can be found in the directory `~/cs230/ps4/web-crawler`. As usual, you will need to execute `cvs update -d` to grab this directory.
2. As mentioned earlier, the subdirectory `~/cs230/ps4/web-crawler/test` contains an executable version of the program you can use for testing purposes.
3. The `isLink()` and `linkURL()` class methods from `URLWordsAndLinks` (see Appendix A) are handy for manipulating the links produced by a `URLWordsAndLinks` iterator.
4. When you compile `WebCrawler`, you may get the following warning, which is safe to ignore:
 

```
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```
5. If you try `java WebCrawler 5 http://cs.wellesley.edu/~cs230/amness/`, the crawler will encounter many links that do not exist or cannot be read. Your web crawler should display a reasonable message for non-existent links and continue indexing.
6. This implementation of `WebCrawler` uses the `SetVectorSorted` implementation of sets and the `TableVectorSorted` implementation of tables. These work fine for a moderate number of elements (on the order of  $10^5$ ), but begin to noticeably slow down at around  $10^6$  elements. For larger number of elements, more efficient versions of these data structures should be use. However, you will not encounter such large number of elements on this problem if you keep `maxDistance` small.
7. Your should submit a testing transcript for this problem. Your transcript should contain *at least* the test cases in Appendix B, but may contain additional test cases as well.

### Problem 2 [65]: The Color Guessing Game

The description for this problem can be found on the web at <http://cs.wellesley.edu/~cs230/ps4-color-game>. For this problem, you should implement the `GuessTheColor.java` game described there in the directory `~/cs230/ps4/color-game`.

## Appendix A: The URLWordsAndLinks class

An instance of `URLWordsAndLinks` is an `Iterator<String>` that enumerates all words and links in the web page at a given URL. In this context, a “word” is defined in a way similar to its definition in the `FileWords` class you use on Exam 1 and in Lab 8. In particular, a word is any contiguous sequence of alphanumeric characters (i.e., letters and digits) and slashes, as well as certain special characters (namely, `'.'`, `'-'`, `'_'`, `'\''`, and `':'`) that occur between two alphanumeric symbols. Character sequences beginning `"/` and `"/` are also considered to be words.

Links are represented as strings of the form `link=url`, where `url` is an absolute URL. For example: `link=http://cs.wellesley.edu/~cs230/amness/descartes.txt`.

```
public URLWordsAndLinks (String url) throws IOException;
```

Creates an iterator for the words and links in the web page at the URL `url`. Throws an `IOException` if there is no web page at `url`.

```
public boolean hasNext ();
```

Returns `true` if there is another word or link to yield from this iterator, and `false` otherwise.

```
public String next ();
```

Returns the next word or link from this iterator. Throws a `RuntimeException` if there are no more words/links in this iterator.

```
public static boolean isLink (String s);
```

Returns `true` if `s` is a link (i.e., a string beginning `link=`) and `false` if `s` is a word.

```
public static String linkURL (String link);
```

If `link` is a link (i.e., a string beginning `link=`), returns the URL embedded in it. Otherwise, throws a `RuntimeException`.

## Appendix B: A WebCrawler Transcript

```
[fturbak@puma web-crawler] java WebCrawler 0 http://cs.wellesley.edu/~cs230/amness/  
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/; distance=0]
```

```
query> descartes  
http://cs.wellesley.edu/~cs230/amness/
```

```
query> #contents  
/a: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
/body: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
/h1: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
/html: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
/title: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
/ul: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
a: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
am: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
amness: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
body: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
descartes: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
descartes.txt: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
dome: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
dome.html: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
famous: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
h1: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
href: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
html: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]  
i: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
```

```
li: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
popeye: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
popeye.html: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
quotes: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
seuss: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
seuss.txt: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
title: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
ul: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
```

query>

Thank you for using the indexing query system!

```
[fturbak@puma web-crawler] java WebCrawler 1 http://cs.wellesley.edu/~cs230/amness/
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/; distance=0]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/descartes.txt; distance=1]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/seuss.txt; distance=1]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/popeye.html; distance=1]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/dome.html; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/descartes.txt; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/seuss.txt; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/popeye.html; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/dome.html; distance=1]
```

query> am

```
http://cs.wellesley.edu/~cs230/amness/
http://cs.wellesley.edu/~cs230/amness/descartes.txt
http://cs.wellesley.edu/~cs230/amness/popeye.html
http://cs.wellesley.edu/~cs230/amness/seuss.txt
```

query> am green

```
http://cs.wellesley.edu/~cs230/amness/seuss.txt
```

query> #contents // Note: the following results have been reformatted to fit the page

```
./: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/dome.html]
/a: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/,
  http://cs.wellesley.edu/~cs230/amness/dome.html,
  http://cs.wellesley.edu/~cs230/amness/popeye.html]
/body: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
/h1: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
/html: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
/title: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
/ul: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
a: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/,
  http://cs.wellesley.edu/~cs230/amness/dome.html,
  http://cs.wellesley.edu/~cs230/amness/popeye.html,
  http://cs.wellesley.edu/~cs230/amness/seuss.txt]
also: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/popeye.html]
am: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/,
  http://cs.wellesley.edu/~cs230/amness/descartes.txt,
  http://cs.wellesley.edu/~cs230/amness/popeye.html,
  http://cs.wellesley.edu/~cs230/amness/seuss.txt]
amness: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
and: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
anywhere: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
are: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
back: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/dome.html]
be: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
boat: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
body: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
box: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
car: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
could: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
```

dark: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
descartes: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/  
  http://cs.wellesley.edu/~cs230/amness/descartes.txt,  
  http://cs.wellesley.edu/~cs230/amness/popeye.html]  
descartes.txt: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/  
do: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
dome: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/  
  http://cs.wellesley.edu/~cs230/amness/dome.html]  
dome.html: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/  
eat: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
eggs: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
famous: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/  
fox: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
georgia: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/dome.html]  
goat: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
good: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
green: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
h1: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/  
ham: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
here: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
house: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
href: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/  
  http://cs.wellesley.edu/~cs230/amness/dome.html,  
  http://cs.wellesley.edu/~cs230/amness/popeye.html]  
html: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/  
http://cs.wellesley.edu/~cs230/amness/descartes.txt:  
  SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/popeye.html]  
http://cs.wellesley.edu/~gdome: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/dome.html]  
i: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/  
  http://cs.wellesley.edu/~cs230/amness/descartes.txt,  
  http://cs.wellesley.edu/~cs230/amness/dome.html,  
  http://cs.wellesley.edu/~cs230/amness/popeye.html,  
  http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
iambic: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/dome.html]  
if: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
in: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
index: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/dome.html]  
let: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
li: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/  
like: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
may: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
me: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
mouse: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
not: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
on: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
or: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
p: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/dome.html,  
  http://cs.wellesley.edu/~cs230/amness/popeye.html]  
pentameter: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/dome.html]  
popeye: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/  
  http://cs.wellesley.edu/~cs230/amness/popeye.html]  
popeye.html: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/  
quotes: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/  
rain: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
rene: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/descartes.txt]  
sam: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
sam-i-am: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
say: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
see: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/popeye.html,  
  http://cs.wellesley.edu/~cs230/amness/seuss.txt]  
seuss: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/  
seuss.txt: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]

```
so: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
thank: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
that: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
the: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
them: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
there: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
therefore: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/descartes.txt,
  http://cs.wellesley.edu/~cs230/amness/dome.html]
they: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
think: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/descartes.txt,
  http://cs.wellesley.edu/~cs230/amness/dome.html]
title: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
to: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/dome.html]
train: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
tree: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
try: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
ul: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/]
what: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/popeye.html]
will: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
with: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
would: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
you: SetVectorSorted[http://cs.wellesley.edu/~cs230/amness/seuss.txt]
```

query>

Thank you for using the indexing query system!

```
[fturbak@puma web-crawler] java WebCrawler 2 http://cs.wellesley.edu/~cs230/amness/
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/; distance=0]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/descartes.txt; distance=1]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/seuss.txt; distance=1]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/popeye.html; distance=1]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/dome.html; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/descartes.txt; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/seuss.txt; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/popeye.html; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/dome.html; distance=1]
Enqueueing Edge[url=http://cs.wellesley.edu/~gdome; distance=2]
Indexing Edge[url=http://cs.wellesley.edu/~gdome; distance=2]
```

query> computer

http://cs.wellesley.edu/~gdome

query> am

```
http://cs.wellesley.edu/~cs230/amness/
http://cs.wellesley.edu/~cs230/amness/descartes.txt
http://cs.wellesley.edu/~cs230/amness/popeye.html
http://cs.wellesley.edu/~cs230/amness/seuss.txt
http://cs.wellesley.edu/~gdome
```

query>

Thank you for using the indexing query system!

```
[fturbak@puma web-crawler] java WebCrawler 3 http://cs.wellesley.edu/~cs230/amness/
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/; distance=0]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/descartes.txt; distance=1]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/seuss.txt; distance=1]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/popeye.html; distance=1]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/dome.html; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/descartes.txt; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/seuss.txt; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/popeye.html; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/dome.html; distance=1]
```

```
Enqueueing Edge[url=http://cs.wellesley.edu/~gdome; distance=2]
Indexing Edge[url=http://cs.wellesley.edu/~gdome; distance=2]
Enqueueing Edge[url=http://cs.wellesley.edu; distance=3]
Enqueueing Edge[url=http://cs.wellesley.edu/~gdome/list.html; distance=3]
Enqueueing Edge[url=http://cs.wellesley.edu/~gdome/table.html; distance=3]
Enqueueing Edge[url=http://cs.wellesley.edu/~gdome/cookies.html; distance=3]
Indexing Edge[url=http://cs.wellesley.edu; distance=3]
Indexing Edge[url=http://cs.wellesley.edu/~gdome/list.html; distance=3]
Indexing Edge[url=http://cs.wellesley.edu/~gdome/table.html; distance=3]
Indexing Edge[url=http://cs.wellesley.edu/~gdome/cookies.html; distance=3]
query> chocolate chip
http://cs.wellesley.edu/~gdome/cookies.html
```

```
query> img
http://cs.wellesley.edu
http://cs.wellesley.edu/~gdome
http://cs.wellesley.edu/~gdome/list.html
http://cs.wellesley.edu/~gdome/table.html
```

```
query>
Thank you for using the indexing query system!
```

```
[fturbak@puma web-crawler] java WebCrawler 4 http://cs.wellesley.edu/~cs230/amness/
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/; distance=0]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/descartes.txt; distance=1]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/seuss.txt; distance=1]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/popeye.html; distance=1]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs230/amness/dome.html; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/descartes.txt; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/seuss.txt; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/popeye.html; distance=1]
Indexing Edge[url=http://cs.wellesley.edu/~cs230/amness/dome.html; distance=1]
Enqueueing Edge[url=http://cs.wellesley.edu/~gdome; distance=2]
Indexing Edge[url=http://cs.wellesley.edu/~gdome; distance=2]
Enqueueing Edge[url=http://cs.wellesley.edu; distance=3]
Enqueueing Edge[url=http://cs.wellesley.edu/~gdome/list.html; distance=3]
Enqueueing Edge[url=http://cs.wellesley.edu/~gdome/table.html; distance=3]
Enqueueing Edge[url=http://cs.wellesley.edu/~gdome/cookies.html; distance=3]
Indexing Edge[url=http://cs.wellesley.edu; distance=3]
Enqueueing Edge[url=http://cs.wellesley.edu/cirque/invite.html; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/acm03; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/talks/jette; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/HessFellowship.html; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/HessFellowship.pdf; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/About/welcome.html; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/news; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/handbook.pdf; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/About/history.html; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/About/facilities.html; distance=4]
Enqueueing Edge[url=http://www.wellesley.edu/Admission/admission/visiting.html; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/People/faculty.html; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/People/alumnae.html; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Resources/tutoring.html; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Resources/unix.html; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Resources/work.html; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Resources/internships.html; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Resources/gradprog.html; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Resources/cswomen.html; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Resources/csnews.html; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Curriculum/Major.html; distance=4]
Enqueueing Edge[url=http://www.wellesley.edu/MAS; distance=4]
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Curriculum/whichCS1xx.html; distance=4]
```

Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Curriculum/Courselistings.html; distance=4]  
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Curriculum/OnlineCourses.html; distance=4]  
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Curriculum/MITcourses.html; distance=4]  
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/diagram.pdf; distance=4]  
Enqueueing Edge[url=http://cs.wellesley.edu/%7Ecs/Fall2006schedule.pdf; distance=4]  
Enqueueing Edge[url=http://cs.wellesley.edu/%7Ecs/Spring2007schedule.pdf; distance=4]  
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Research/research.html; distance=4]  
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Research/independ.html; distance=4]  
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Research/honors.html; distance=4]  
Enqueueing Edge[url=http://cs.wellesley.edu/cirque/; distance=4]  
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Activities/studentsem.html; distance=4]  
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Activities/cspanels.html; distance=4]  
Enqueueing Edge[url=http://cs.wellesley.edu/~cs/Activities/studentfun.html; distance=4]  
Enqueueing Edge[url=http://cs.wellesley.edu/%7Ecs/Activities/calendar07.html; distance=4]  
Enqueueing Edge[url=http://cs.wellesley.edu/~WAAM/; distance=4]  
Enqueueing Edge[url=http://cs.wellesley.edu/~wac/; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~gdome/list.html; distance=3]  
Indexing Edge[url=http://cs.wellesley.edu/~gdome/table.html; distance=3]  
Enqueueing Edge[url=http://cs.wellesley.edu/~gdome/index.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~gdome/cookies.html; distance=3]  
Enqueueing Edge[url=http://www.snopes2.com/business/consumer/cookie.htm; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/cirque/invite.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/acm03; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/talks/jette; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/HessFellowship.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/HessFellowship.pdf; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/About/welcome.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/news; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/handbook.pdf; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/About/history.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/About/facilities.html; distance=4]  
Indexing Edge[url=http://www.wellesley.edu/Admission/admission/visiting.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/People/faculty.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/People/alumnae.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Resources/tutoring.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Resources/unix.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Resources/work.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/%7Ecs/Resources/internships.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Resources/gradprog.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Resources/cswomen.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Resources/csnews.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Curriculum/Major.html; distance=4]  
Indexing Edge[url=http://www.wellesley.edu/MAS; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Curriculum/whichCS1xx.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Curriculum/Courselistings.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Curriculum/OnlineCourses.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Curriculum/MITcourses.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/diagram.pdf; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/%7Ecs/Fall2006schedule.pdf; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/%7Ecs/Spring2007schedule.pdf; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Research/research.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Research/independ.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Research/honors.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/cirque/; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Activities/studentsem.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Activities/cspanels.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~cs/Activities/studentfun.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/%7Ecs/Activities/calendar07.html; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~WAAM/; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~wac/; distance=4]  
Indexing Edge[url=http://cs.wellesley.edu/~gdome/index.html; distance=4]  
Indexing Edge[url=http://www.snopes2.com/business/consumer/cookie.htm; distance=4]

```
query> cs110
http://cs.wellesley.edu/~cs/About/history.html
http://cs.wellesley.edu/~cs/About/welcome.html
http://cs.wellesley.edu/~cs/Curriculum/Courselistings.html
http://cs.wellesley.edu/~cs/Curriculum/OnlineCourses.html
http://cs.wellesley.edu/~cs/Curriculum/whichCS1xx.html
```

```
query> cs11
```

```
query> cs111
http://cs.wellesley.edu/~cs/About/welcome.html
http://cs.wellesley.edu/~cs/Curriculum/Courselistings.html
http://cs.wellesley.edu/~cs/Curriculum/Major.html
http://cs.wellesley.edu/~cs/Curriculum/OnlineCourses.html
http://cs.wellesley.edu/~cs/Curriculum/whichCS1xx.html
```

```
query> cs230
http://cs.wellesley.edu/~cs/Curriculum/Courselistings.html
http://cs.wellesley.edu/~cs/Curriculum/Major.html
http://cs.wellesley.edu/~cs/Curriculum/OnlineCourses.html
```

```
query> cs110 cs111
http://cs.wellesley.edu/~cs/About/welcome.html
http://cs.wellesley.edu/~cs/Curriculum/Courselistings.html
http://cs.wellesley.edu/~cs/Curriculum/OnlineCourses.html
http://cs.wellesley.edu/~cs/Curriculum/whichCS1xx.html
```

```
query> cs110 cs111 cs230
http://cs.wellesley.edu/~cs/Curriculum/Courselistings.html
http://cs.wellesley.edu/~cs/Curriculum/OnlineCourses.html
```

```
query>
Thank you for using the indexing query system!
```

```
[fturbak@puma web-crawler]
```

*Problem Set Header Page*  
*Please make this the first page of your hardcopy submission.*

## **CS230 Problem Set 4**

### **Due 6pm Friday April 6**

Names of Team Members:

Date & Time Submitted:

Collaborators (*anyone you or your team collaborated with*):

*By signing below, I/we attest that I/we have followed the collaboration policy as specified in the Course Information handout.*

Signature(s):

*In the **Time** column, please estimate the time you or your team spent on the parts of this problem set. Team members should be working closely together, so it will be assumed that the time reported is the time for each team member. Please try to be as accurate as possible; this information will help me design future problem sets. I will fill out the **Score** column when grading your problem set.*

<b>Part</b>	<b>Time</b>	<b>Score</b>
General Reading		
Problem 1 [35]		
Problem 2 [65]		
<b>Total</b>		