

CIS 121 - Fall 2009

Homework 5 – Monday, October 26

Due Wednesday, November 11, 10AM

Required Problems 4 (100 points)

In this assignment you will implement a relatively simple web-crawler and search-engine based on maps. The web-crawler will extract keywords from each page it visits and store them along with the address of the page in an index. The search-engine takes an array of keywords and returns a map that contains the pages that match the keywords and how many keywords were matched by the page. We've provided a helper class `WebpageParser` that will extract relevant keywords and links from a website. Also we've included an `Integer` comparator `reverseIntegerComparator` that sorts in descending order instead of the default ascending order provided by the class `Integer` that you are free to use if you find useful.

For this assignment you are only allowed to instantiate a `TreeMap` and a `HashMap`. The underlying representation of a `TreeMap` is a Red-Black tree. The `HashMap` is implemented with a hash table. You will use many maps in this assignment. Along with your code you must include a short explanation of why you chose a `TreeMap` or a `HashMap` for each instance of a map that you used.

Problem 1: 25 Points; files to submit `WebpageIndexEntry.java`

Write an implementation for the `WebpageIndexEntryI` interface below. This interface defines a webpage entry in a webpage index.

Important: Include a default constructor for us to use in grading.

```
public interface WebpageIndexEntryI
{
    public URL                getAddress();
    public Set<URL>           getLinks();
    public Map<Integer, Set<String>> getFrequencyMap();
    public void               setAddress(URL url);
    public void               addLinks(Set<URL> links);
    public void               addKeyword(String keyword);
}
```

Problem 2: 25 Points; file to submit `WebpageIndex.java`

Implement the methods `addEntry` and `keywordMap` in the class `WebpageIndex`. `addEntry` adds the webpage entry to this index. `keywordMap` takes a frequency and returns a map from keywords to webpage entries. In order for the website to count a word as a keyword it must contain that word at least the number of times specified by the given frequency.

Important: Include a default constructor for us to use in grading.

```

public class WebpageIndex
{
    public void addEntry(WebpageIndexEntryI entry) {}
    public Map<String, Set<WebpageIndexEntryI>> keywordMap(int frequency) {}
}

```

Problem 3: 25 Points; file to submit `Webcrawler.java`

Implement the method `createWebpageIndex` in the class `Webcrawler`. `createWebpageIndex` takes an initial web-address and the number of pages to crawl. Each time the crawler visits a webpage, if it contains keywords it creates a new `WebpageIndexEntryI` for that site and fills it in, adds it to a `WebpageIndex` and adds all links the site links to to the list of pages to crawl. It stops either when there are no more links to explore or when it has explored the number of pages specified.

```

public static WebpageIndex createWebpageIndex(int pagesToCrawl,
        URL startingURL)
{
    // Your code here.
}

```

Problem 4: 25 Points; file to submit `WebSearch.java`

Write the method `search` in the class `WebSearch`. This method has as input a `webindex`, minimum frequency count and an array of keywords. It returns a map where the keys are the number of keywords matched and the values are the set of webpages that matched that many keywords. A word in a `WebpageIndexEntryI` only counts as a keyword if it appears at least the number of times specified by the minimum frequency.

You are free to do a strict matching, where a webpage matches a keyword only if it has a keyword that matches letter for letter one of the searched for keywords, or a more fuzzy matching where, for example, if you search for apple it also matches on apples.

```

public static Map<Integer, Set<URL>> search(WebpageIndex index,
        int minFrequency, String[] keywords)
{
    // Your code here.
}

```