

# Hints for lab assignment 1

CS572, Fall2002, Iowa State University  
Jie Bao

<http://www.cs.iastate.edu/~cs572/labs/lab1/lab1.html>

## Algorithm to be implemented

- ? BFS
- ? DFS
- ? Best First
- ? Beam (you can view it as a special kind of DFS)

## What you need to do:

- 1) Implement your search algorithm in  
static void WebSearch::addNewChildrenToOPEN(SearchNode parent, String contents, String searchStrategy)  
    if (searchStrategy.equalsIgnoreCase("breadth"))  
        {  
            OPEN.addElement(new SearchNode(parent,hyperlink));  
        }  
    else if (searchStrategy.equalsIgnoreCase("depth"))  
        {  
        }  
    else if (searchStrategy.equalsIgnoreCase("best"))  
        {  
        }  
    else if (searchStrategy.equalsIgnoreCase("beam"))  
        {  
        }  
    }
- 2) Define search node:  
class SearchNode  
    ? Constructor SearchNode()  
    ? getHvalue(): returns (as a double) the heuristic value associated with a search node,  
    ? getNodeName() that returns (as a String) the name of the file (eg, "page7.html") associated with  
    this node, and  
    ? reportSolutionPath() that prints the path from the start node to the current node represented by the  
    SearchNode instance.
- 3) Design heuristic function SearchNode:: getHvalue()

Since an intranet imitates the web, your surfing experience on the web might help you create good heuristics for your heuristic search. One obvious feature that could be incorporated into a heuristic function for this domain is to count how many special words appear on a given page. If special words show up a lot on a page, there is a high probability that the page is close to a goal page. That is, you can believe you are closer to a goal if you choose to go to a page with more special words on it.

The intuition about the special words on a page also applies to the hyperlinks on a page. That is, if a number of hyperlinks are sprinkled throughout a page, you probably will hesitate for a while in order to decide which one to click on first, given that you want to find the goal as quickly as possible. We often examine the words themselves in a hyperlink in order to guess what kind of page this link will lead to. Because the hypertext associated with the link is often a title, summary, or highlight of the page that the link is pointing to, it is reasonable to guess that the hypertext is similar to the page it leads to in terms of the content of special words. We prefer to first expand those links containing QUERY words because they are more similar to the goal than those links containing only plain words. This suggests a second feature that could be used in defining a heuristic function.

A third possible feature for a heuristic function is also based on the notion of similarity. Notice that the information we are searching for is the sequence of consecutive words QUERY1 QUERY2 QUERY3 QUERY4. Naturally, we think that the order and the consecutiveness is important. Thus we prefer to continue searching at links that have partial sequences of special words that are as similar as possible to the goal sequence. For example, Page4 contains four links. Because the second link contains QUERY2 QUERY3, which other links do not, the page associated with the second link is the best successor page. To implement this feature you must define similarity by quantifying a notion of distance from the goal sequence of words. Look at other pages to get ideas about how to formulate your similarity measure.

All of the above knowledge about web pages concentrates on the content of a page in terms of special words. Considering the positions of those special words on a page is also a good idea. That is, words at the beginning of a page include titles and main ideas, and thus are suggestive of the content of the page. Applying this idea to our intranet, assume that the earlier a special word appears on a page, the more promising the page is. This is a fourth possible feature to use in defining your heuristics.

Some or all of the above four features, and possibly others that you invent, can be combined to define heuristic functions for heuristic searches.

Use the above information to devise a heuristic function for use in best-first search. Describe your motivation for your heuristic. Notice that unlike the standard approach where the heuristic is applied to the next state, here we want to use our heuristic to decide which hyperlink to 'click on' (fetch) next. This means that your heuristic function scores each arc (hyperlink) coming out of the current node and not each child node. Is your heuristic admissible? Explain why or why not. (You're not required to write an admissible heuristic.)

eg.

```
// our goal is QUERY1 QUERY2 QUERY3 QUERY4
h = 0;
If (contains QUERY1 ) h ++;
If (contains QUERY2 ) h ++;
If (contains QUERY3 ) h ++;
If (contains QUERY4 ) h ++;
If (contains QUERY1 QUERY2) h ++;
If (contains QUERY1 QUERY2 QUERY3) h ++;
```

## Turn In

Run all of the search strategies implemented on intranets 1, 5, and 7, and turn in electronically the source code (.java) with a README file explaining how to run the program after compilation. Also turn in a report of the results obtained (nodes visited and solution path found). DON'T TURN IN THE RESULTS OF RUNNING THE CODE WITH `debugging = true`. Also turn in a hardcopy of your commented source code and a neatly written lab report that includes the material and answers for the questions listed above. Use the turnin script to turn in an electronic version of your implementation.

You can't submit the lab without a COMS account.

Suppose your files are stored in subdirectory lab1, within lab1 execute the following: `/home/course/cs572/public/Utilities/turnin lab1`

You can also type the following into your `.cshrc` file (without the quote): `"alias turnin /home/course/cs572/public/Utilities/turnin"`. In order to make the alias become effective, you need to logout and login again. Then, go to the directory where your files are stored (for example, lab1), and execute the following: `turnin lab1`  
Also note that you have to turn in your program from a ComS system with `/usr/bin/Mail` program, such as popeye. You will receive a mail confirming successful turn in.