**CSD301: Advanced Algorithms**

***Instructor: Tran Hong Yen***

**PROGRAMMING PROJECT**

Several programming projects are listed and briefly described below. Students will select one of these topics to work in team (4 members per team). After getting the project, please make appointments with the instructor to discuss the requirements of your topic in more detail.

**1. Project Goals**

- Challenge yourself with solving a practical problem based on the knowledge and skills you get from the course, especially test your abilities to:

+ Design algorithm and data structure, and implement them solving computing problems;

+ Evaluate existing algorithm/data structure within a computer-based system, process, component and conclude about their appropriateness for its solution;

+ Select suitable algorithmic and data structure design for a computer software or a computer-based solution for a problem that demonstrate the tradeoffs between design choices;

**2. Task requirements**

- Study theoretically on the project topic.

- Apply the knowledge of algorithm and datastructure to solve the problem in your project.

- Implement your solution.

**3. Reporting Requirements**

Your project reports have to meet the requirements below:

a. Content requirements: Please to be noted that in your project report you have to show clearly:

+ Task assignment to each member of your team.

+ Background of the problem, related works.

+ Description of your solution to the problem and the architecture of your system:

* Why you choose this algorithmic and data structure design? Answer this question by analyzing and demonstrating the trade-offs between design choices (time and space complexity, efficiency for the problem…)
* How you design data structures and algorithms for your system?

+ Evaluation of your solution (implementation details, experimental setup, dataset, results)

+ Conclusion of the project results, future work

b. Format requirements: Your project report have to follow an academic technical format.

**4. Submission requirements**

- Due date: 12/11/2013

- Create the directory with a name like <class>-< projectnumber>, e.g. CS0701-Proj01 (1)

- The (1) directory contains:

+ Your PDF report project.

+ Your PPTX presentation file.

+ Your source code files and dataset.

+ Your help document for instructing the use of your system.

- Compress the folder (1) to .zip (or .rar) file (with the same name), and send to email: yenth@fpt.edu.vn.

**5. Project assessment**

- Extra credit for new ideas or applying complex techniques from papers rather than textbook.

- Programs submitted late will get a penalty of 1 mark out of 10 per day late.

**Project 1: Basic Web Crawler**

**1. Introduction**

In this project, you will study how search engine collect web documents, implement a basic web crawler with some specific crawling strategy and visualize the structure of the web graph in a particular domain (e.g. <http://www.fpt.edu.vn>)

**2. Your task**

- Build a web crawler:

Your web crawler will haveto perform the following tasks:

**+** Start with a particular URL (e.g. <http://www.fpt.edu.vn>)

**+** Perform a Web traversal using breadth-first strategy and depth-first strategy with some limit of the depth of crawling (i.e. the number of hops (links) to follow and the size of the pages to be fetched). Collect the satisfied web pages and store them in a local repository.

**+** Keep track of the traversed URLs, making sure they are part of the fpt.edu.vn domainand they were not already traversed (i.e. avoid duplicates, avoid cycles)

- Visualize the structure of the web graph:

+ You will have to build a web graph from the crawled links to illustrate the structure of the web domain.

+ Apply some graph algorithms on this web graph.

**3. Optional**

A web crawler designed to crawl the entire web is a sophisticated program that uses advanced programming technology to improve its time and space efficiency. For extra mark, you should study on some problems with large-scale crawlers and choose any technique you want to improve your basic web crawler.

**Project 2: Subject-Specific Web Crawler**

**1. Introduction**

In this project, you will learn how to build a subject-specific web crawler which is utilized in subject-specific search engines. As the smart crawler traverses, sites not relevant to the particular subject are filtered, and only sites found relevant are indexed and stored in a database for later searching.

**2. Your task**

**-** Build a “smart” web crawler

+ Input:

- A particular URL from which to start the crawl (e.g. <http://www.fpt.edu.vn>)

- A query word, or a set of query words

- A maximum number of pages to download.

**+** Crawl: Any of the following variants: (Choose one of these)

- Crawl only from pages whose text includes one of the words in the list.

- Crawl only from pages whose title includes one of the words in the list.

- Crawl only across links where the anchor includes one of the words in the list.

- Combine some above conditions (up to you)

- If no links satisfy any of the above criteria, then the crawler should just do a breadth-first search.)

+ Output: “Search results” page: It should be an HTML file with a link for every page you have downloaded. If the page has a title, then the anchor of the link should be the title; otherwise, the anchor for the link should be the URL.

**3. Optional**

For extra mark, you may implement any of the following features: stemming, synonymy…

**Project 3: Link-based focussed Web Crawler**

**1. Introduction**

In this project, you will learn how to build a linked-based focussed web crawler. When traversing, the smart crawler will utilizes an evaluation function to make an informed graph search algorithm. As an independent indicator of web page importance, the evaluation function can enhance page navigation.

**2. Your task**

**-** Build a “smart” web crawler

+ Input:

- A particular URL from which to start the crawl (e.g. <http://www.fpt.edu.vn>)

- A maximum number of pages to download.

**+** Crawl: Give priority to page with max number of in-links from pages already seen. In case of ties, prefer page with max number of out-links to pages already seen.

+ Output: “Search results” page: It should be an HTML file with a link for every page you have downloaded. If the page has a title, then the anchor of the link should be the title; otherwise, the anchor for the link should be the URL.

**3. Optional**

For extra mark, you may implement PageRank, HITS or other algorithms as an evaluation function

**Project 4: Indexer for Phrase Query Search**

**1. Introduction**

In this project, you will learn how to construct an indexer for query search in search engine.

**2. Your task**

**-** Construct an indexer

+ Input:

- A collection of text documents (should be as large as possible).

- A phrase query search.

**+** Index: Design and implement an indexer for a simple query search.

+ Output: “Search results” page: It should be an HTML file with a link for each text document.

**3. Optional**

For extra mark, you may implement any of the following features: stemming, synonymy…Note that the more complex phrase query your system can search, the higher mark you can get.

**Project 5: Indexer for Wild-card Query Search**

**1. Introduction**

In this project, you will learn how to construct an indexer for wild-card query search in search engine.

**2. Your task**

**-** Construct an indexer

+ Input:

- A collection of text documents (should be as large as possible).

- A wild-card query search.

**+** Index: Design and implement an indexer for a wild-card query search.

+ Output: “Search results” page: It should be an HTML file with a link for each text document.

**3. Optional**

For extra mark, you may implement any of the following features: stemming, synonymy…Note that the more complex phrase query your system can search, the higher mark you can get.

**Project 6: Spell Checker**

**1. Introduction**

In this project, you will learn the process called spell checking. This process verify that a particular word is spelled properly according to some dictionary. Spell checkers are used in many applications, including word processors (such as Microsoft Word), electronic dictionaries…

Spell checking itself is trivial, requiring only a simple lookup in a dictionary. However, most applications of spell checking also require that the spell checker provide a list of potentially

correct spellings (“near matches”) when the word was spelled improperly. For instance, if I type

“speling” into an online dictionary, it will provide suggestions of similar words that I may have

meant to type, including “spelling”, “spoiling”, “sapling”, and “splendid”.

**2. Your task**

- Build a spell checker that:

+ Determines if a given word is spelled correctly based on a dictionary lookup.

+ If the word is not spelled correctly, it will provide a list of similar words, and ordered based on their edit distance from the string that the user typed.

**3. Optional**

For extra mark, you may implement more complex algorithm to correct spell (e.g. Metaphone algorithm) or build spell checker for Vietnamese language.

**Project 7: Incremental Search**

**1. Introduction**

In this project, you will learn how to design and implement an incremental search feature like Google Instant. Incremental search or real-time suggestions is a [user interface](http://en.wikipedia.org/wiki/User_interface) interaction method to progressively search for and filter through text. As the user type text, one or more possible matches for the text are found and immediately presented to the user. This immediate feedback often allows the user to stop short of typing the entire word or phrase they were looking for. The user may also choose a closely related option from the presented list.

**2. Your task**

- Build an application with incremental search feature (e.g. web browsers, search instant, code completion)

**3. Optional**

For extra mark, you may consider some other effective datastructure and algorithm design support this feature for searching in large dataset.

**Project 8: Map Application**

**1. Introduction**

In this project, you will learn how to implement some queries in a map application (GIS system).

**2. Your tasks**

- Get a HaNoi shapefile (e.g. from OSM (OpenStreetMap <http://www.openstreetmap.org/>)

- Read data from this shapefile to implement some simple queries (e.g. find the k nearest roads to a location, find the shortest path from point A to point B…) by your own code (not use supported API to solve these queries)

- Output: The result of queries

**3. Optional**

The more queries your system can answer correctly, the higher mark you can get