# COMP 4321 - Project Report and Documentation

Leung Ka Wa, 20770807, kwleungau@connect.ust.hk

## Program code Structure

### java source code (project_package)

- **Crawler.java -** Crawler class
- **StopStem.java -** StopStem class
- **URLIndex.java -** URLIndex class, use to manipulate the URL.db
- **WordIndex.java -** WordIndex class, use to manipulate the WordDB.db
- **Tester.java -** Tester class, use to test and run the crawler.
- **SearchEngine.java -** SearchEngine class, use to perform IR.

### Library

No extra library from lab is used in this project.

## Design of the jdbm database scheme

### URL.db

It contain of 6 objects. Each of them is a HTree.

- **PageID** - Store the URL and its pageID.

  (**String**)URL = (**UUID**)pageID. Example:

  http://library.hkust.edu.hk/events/staff-workshops/ = b9275b04-58a1-3f4f-ab38-606e30a198a8

  Design decision: A ID mapping.

- **ParentToChilden** - Store the parent to children relationship.

  (**UUID**)parentID = Vector⟨**UUID**⟩(childID). Example:

  5ed456f8-36f3-3ca2-8451-62696e13f7fc = [b9275b04-58a1-3f4f-ab38-606e30a198a8, 9e4a5a31-dbb7-39d0-82ab-8d8b37595564, bd93a542-88ec-3b29-b739-9faf1ffc3bdc, . . . ]

  Design decision: Can easily get the out link of a page for later use, e.g. PageRank, hub weight, authority weight and HITS.

- **ChildToParents** - Store the child to parents relationship.

  (**UUID**)childID = Vector⟨**UUID**⟩(parentID). Example:

  5ed456f8-36f3-3ca2-8451-62696e13f7fc = [b9275b04-58a1-3f4f-ab38-606e30a198a8, 9e4a5a31-dbb7-39d0-82ab-8d8b37595564, bd93a542-88ec-3b29-b739-9faf1ffc3bdc, . . . ]

  Design decision: Can easily get the in link of a page for later use, e.g. PageRank, hub weight, authority weight and HITS.

- **PageToTitle** - Store the pageID and its originial title.

  (**UUID**)pageID = (**String**)title. Example:

  bd93a542-88ec-3b29-b739-9faf1ffc3bdc = "This is the Title"

  Design decision: Store the whole title for display use only.

- **LastModified** - Store the pageID and its last modified date.

  (**UUID**)pageID = (**Date**)lastModifiedDate. Example:

  bd93a542-88ec-3b29-b739-9faf1ffc3bdc = (**Date**)Thu Jun 16 16:47:33 HKT 2022

  Design decision: Store the last modified date of a page to determine whether the page is updated or not.

- **PageSize** - Store the pageID and its size.

  (**UUID**)pageID = (**Integer**)size. Example:

  bd93a542-88ec-3b29-b739-9faf1ffc3bdc = 1024

  Design decision: Store the size of a page to determine how much infomation are in this page.

## WordDB.db

It contain of 4 objects. Each of them is a HTree.

- **WordID** - Store the word and its ID.

  (**String**)word = (**UUID**)wordID. Example:

  intellig = b9275b04-58a1-3f4f-ab38-606e30a198a8

  Design decision: A ID mapping.

- **Inverted** - Store the wordID and posting list.

  (**UUID**)wordID = Map⟨(**UUID**)pageID, Vector⟨**Integer**⟩(position)⟩. Example:

  b9275b04-58a1-3f4f-ab38-606e30a198a8 = {9bfc960c-53e4-3faf-8623-b44c251584c1=[1, 5, 10], 114471e0-e3dd-39d8-aa8a-11f77c85a7fa=[50, 60], 8019de9c-bcf5-3600-814b-53ed90ab33bb=[10], …}

  Design decision: Store the posting list of the word for tfxidf and phase search. Also, finding the document with highest word frequency is easy.

- **Forward** - Store the pageID and its forward word list.

  (**UUID**)pageID = Map⟨(**UUID**)wordID, Vector⟨**Integer**⟩(position)⟩. Example:

  b9275b04-58a1-3f4f-ab38-606e30a198a8 = {9bfc960c-53e4-3faf-8623-b44c251584c1=[1, 5, 10], 114471e0-e3dd-39d8-aa8a-11f77c85a7fa=[50, 60], 8019de9c-bcf5-3600-814b-53ed90ab33bb=[10], …}

  Design decision: Store the forward word list of the page for later algorithm, e.g. tfxidf. Finding the words and their position to get the phase and frequency in a document is easy.

- **TitleInverted** - Store the wordID and posting list of title.

  (**UUID**)wordID = Map⟨(**UUID**)pageID, Vector⟨**Integer**⟩(position)⟩. Example:

  b9275b04-58a1-3f4f-ab38-606e30a198a8 = {9bfc960c-53e4-3faf-8623-b44c251584c1=[1, 5, 10], 114471e0-e3dd-39d8-aa8a-11f77c85a7fa=[50, 60], 8019de9c-bcf5-3600-814b-53ed90ab33bb=[10], …}

  Design decision: Store the posting list of the word in title to favor matches in title.

# Running the program (crawler part)

## How to run the program

The Tester class is the main calling class of this program. Pass command line argument to it to run the program.

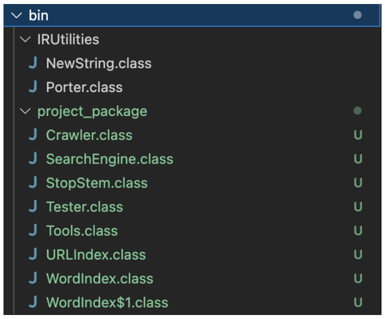
As I am using VS code to develop this project, I was just simply using the java extension and run the program without mannually compile the project. For me the command line is:

```
/usr/bin/env /Users/boscoleung/opt/anaconda3/bin/java @/var/folders/f1
    /6mvnwxt109n9rswystbch0t40000gn/T/cp_dh97avm16bvprxybpew6los8v.
    argfile project_package.Tester <argument>
```

If you want to compile the project mannually, you can run the following command (work on mac):

```
javac -cp ":lib/*" -d bin $(find . -path ./apache-tomcat-10.1.6 -prune -o -name "*.java" -print)
```

A bin folder containing all the classes will be created.



```
bin
  IRUtilities
    NewString.class
    Porter.class
  project_package
    Crawler.class        U
    SearchEngine.class   U
    StopStem.class       U
    Tester.class         U
    Tools.class          U
    URLIndex.class       U
    WordIndex.class      U
    WordIndex$1.class    U
```

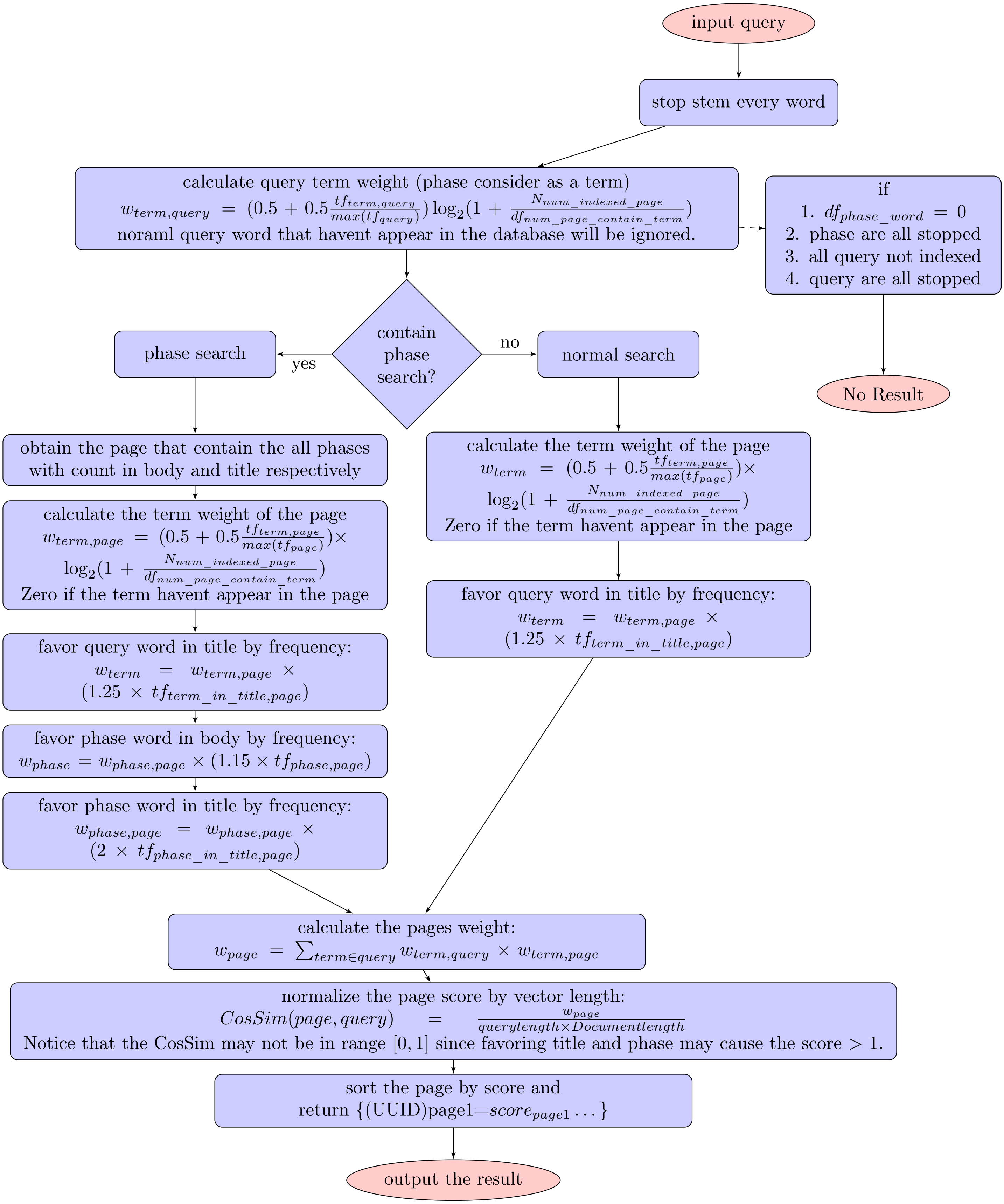
Figure 1: The complied bin folder

Then run the program with the following command:

```
java -cp ".:bin:lib/*" project_package.Tester <argument>
```

- **-runCrawler** - Run the crawler, progress will be printed to the console. The starting URL and the number of pages to crawl can be set by add the url and number. For example, `-runCrawler https://www.google.com 1000`, if no url and number is provided, the default url and number will be used, that is `-runCrawler` = `-runCrawler https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm 300`;
- **-printSpiderResult** - Output the result of the crawler to spider_result.txt. This may take a moment to produce the complete result.
- **-printAllURLdb** - Output all the data in the URL.db to AllURLdb.txt.
- **-printPageTitle** - Output all the data in the URL.db PageToTitle to PageTitle.txt.
- **-printURLPageID** - Output all the data in the URL.db PageID to URLPageID.txt.
- **-printPageMeta** - Output all the data in the URL.db LastModified and PageSize to PageMeta.txt.
- **-printParentToChilden** - Output all the data in the URL.db ParentToChildren to ParentToChildren.txt.
- **-printChildToParents** - Output all the data in the URL.db ChildToParents to ChildToParents.txt.
- **-printAllWordDB** - Output all the data in the WordDB.db to AllWordDB.txt.
- **-printWordID** - Output all the data in the WordDB.db WordID to WordID.txt.
- **-printInverted** - Output all the data in the WordDB.db Inverted to Inverted.txt.
- **-printTitleInverted** - Output all the data in the WordDB.db TitleInverted to TitleInverted.txt.
- **-printForward** - Output all the data in the WordDB.db Forward to Forward.txt.

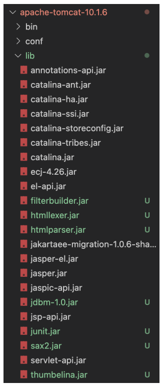
## Search Engine Pipeline


input query
stop stem every word
calculate query term weight (phase consider as a term)
$w_{term,query} = (0.5 + 0.5\frac{tf_{term,query}}{max(tf_{query})})\log_2(1 + \frac{N_{num\_indexed\_page}}{df_{num\_page\_contain\_term}})$
noraml query word that havent appear in the database will be ignored.
if
1. $df_{phase\_word} = 0$
2. phase are all stopped
3. all query not indexed
4. query are all stopped
No Result
contain phase search?
yes
no
phase search
normal search
obtain the page that contain the all phases with count in body and title respectively
calculate the term weight of the page
$w_{term,page} = (0.5 + 0.5\frac{tf_{term,page}}{max(tf_{page})})\times \log_2(1 + \frac{N_{num\_indexed\_page}}{df_{num\_page\_contain\_term}})$
Zero if the term havent appear in the page
favor query word in title by frequency:
$w_{term} = w_{term,page} \times (1.25 \times tf_{term\_in\_title,page})$
favor phase word in body by frequency:
$w_{phase} = w_{phase,page} \times (1.15 \times tf_{phase,page})$
favor phase word in title by frequency:
$w_{phase,page} = w_{phase,page} \times (2 \times tf_{phase\_in\_title,page})$
calculate the term weight of the page
$w_{term} = (0.5 + 0.5\frac{tf_{term,page}}{max(tf_{page})})\times \log_2(1 + \frac{N_{num\_indexed\_page}}{df_{num\_page\_contain\_term}})$
Zero if the term havent appear in the page
favor query word in title by frequency:
$w_{term} = w_{term,page} \times (1.25 \times tf_{term\_in\_title,page})$
calculate the pages weight:
$w_{page} = \sum_{term\in query} w_{term,query} \times w_{term,page}$
normalize the page score by vector length:
$CosSim(page, query) = \frac{w_{page}}{querylength \times Documentlength}$
Notice that the CosSim may not be in range $[0, 1]$ since favoring title and phase may cause the score $> 1$.
sort the page by score and
return $\{$(UUID)page1=$score_{page1} \ldots\}$
output the result

# Running the user interface (JSP web)

## lib

Make sure those .jar are added the the lib of JSP (green .jar are newly added):



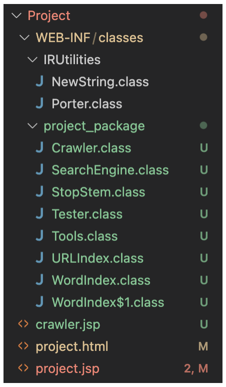
```
apache-tomcat-10.1.6
  bin
  conf
  lib
    annotations-api.jar
    catalina-ant.jar
    catalina-ha.jar
    catalina-ssi.jar
    catalina-storeconfig.jar
    catalina-tribes.jar
    catalina.jar
    ecj-4.26.jar
    el-api.jar
    filterbuilder.jar              U
    htmllexer.jar                  U
    htmlparser.jar                 U
    jakartaee-migration-1.0.6-sha...
    jasper-el.jar
    jasper.jar
    jaspic-api.jar
    jdbm-1.0.jar                   U
    jsp-api.jar
    junit.jar                      U
    sax2.jar                       U
    servlet-api.jar
    thumbelina.jar                 U
```

Figure 2: lib of JSP should contain those .jar

The .jar files can be copied from the lib folder of the project.

## webapps

Make sure the Project folder is in the webapps folder of Tomcat.



```
Project
  WEB-INF / classes
    IRUtilities
      NewString.class
      Porter.class
    project_package
      Crawler.class                U
      SearchEngine.class           U
      StopStem.class               U
      Tester.class                 U
      Tools.class                  U
      URLIndex.class               U
      WordIndex.class              U
      WordIndex$1.class            U
  crawler.jsp                      U
  project.html                     M
  project.jsp                   2, M
```

Figure 3: The Project folder

The package under WEB-INF/classes can be copy from the compiled bin folder from figure 1.

## DataBase

The databases should be placed in the same folder as the project folder (for my case). Even if you don't know where should the database be placed or don't have the databases, you can use the bonus features - Crawler in JSP web interface to index more pages to create databases and crawl pages. Details can be found in the Bonus features beyond requirement section.

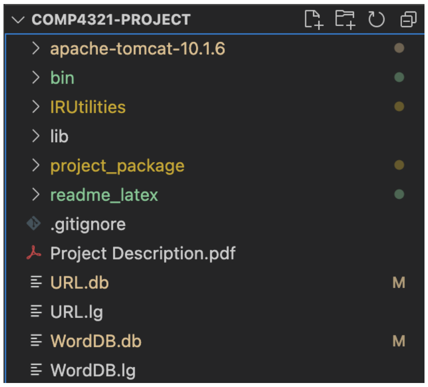


Figure 4: The databases locations

## Run

In the project folder, run the following command:

**apache-tomcat-10.1.6/bin/startup.sh**

Then, open the browser and go to the following url:

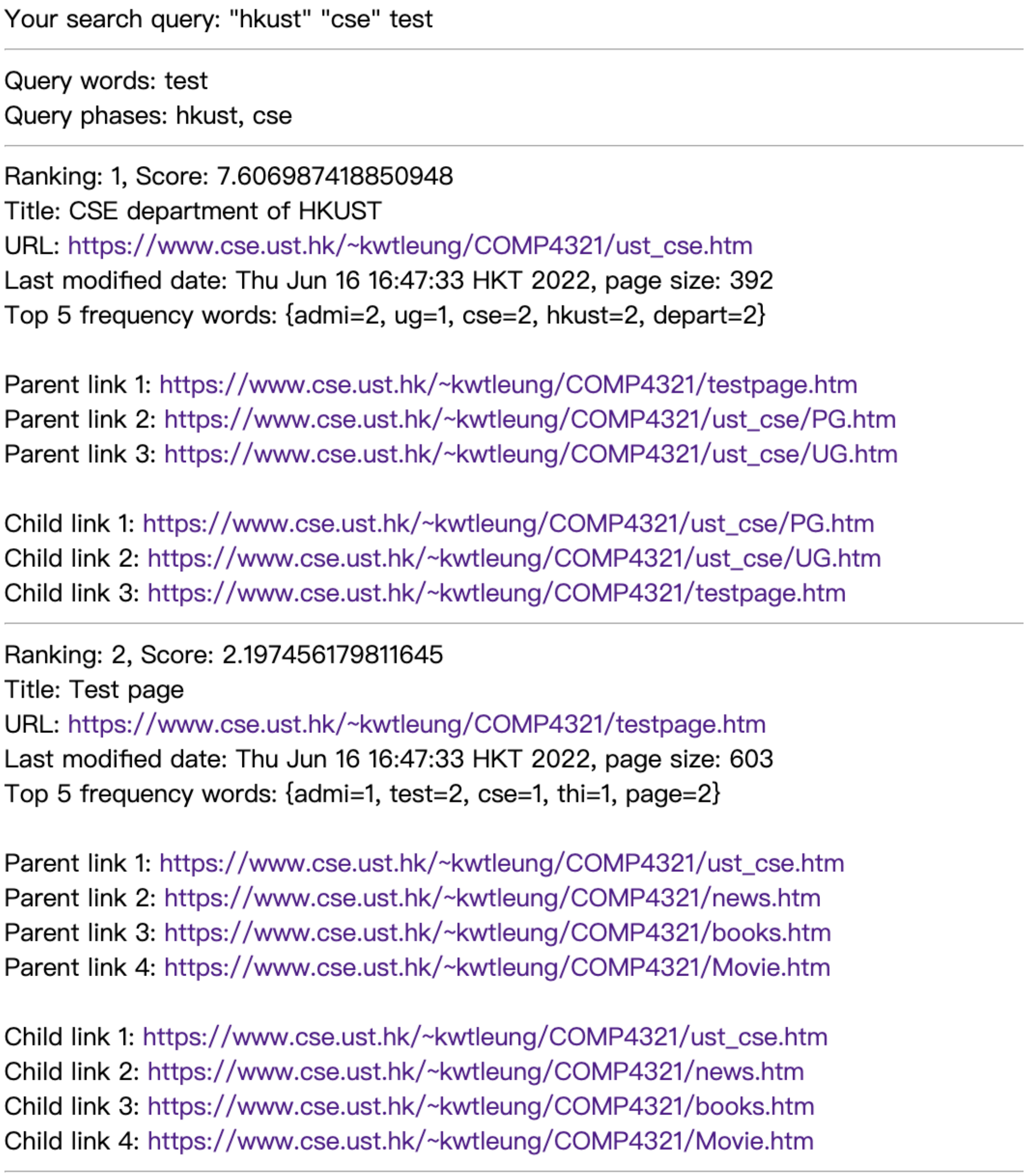
**http://localhost:8080/Project/project.html**

## Usage and show test

Type the query in the search box and click search.

Figure 5: Search box

The result will be shown in the following format with your query words and phase shown:



Your search query: "hkust" "cse" test

Query words: test<br>
Query phases: hkust, cse

Ranking: 1, Score: 7.606987418850948
Title: CSE department of HKUST
URL: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
Last modified date: Thu Jun 16 16:47:33 HKT 2022, page size: 392
Top 5 frequency words: {admi=2, ug=1, cse=2, hkust=2, depart=2}

Parent link 1: https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm
Parent link 2: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/PG.htm
Parent link 3: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/UG.htm

Child link 1: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/PG.htm
Child link 2: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/UG.htm
Child link 3: https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm

Ranking: 2, Score: 2.197456179811645
Title: Test page
URL: https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm
Last modified date: Thu Jun 16 16:47:33 HKT 2022, page size: 603
Top 5 frequency words: {admi=1, test=2, cse=1, thi=1, page=2}

Parent link 1: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
Parent link 2: https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm
Parent link 3: https://www.cse.ust.hk/~kwtleung/COMP4321/books.htm
Parent link 4: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

Child link 1: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
Child link 2: https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm
Child link 3: https://www.cse.ust.hk/~kwtleung/COMP4321/books.htm
Child link 4: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

Figure 6: Result

# Bonus features beyond requirement

## Query length

The query length don't have a maximum. It can be any length.

## Phase length

The phase length is not limited to 2 or 3. It can be any length. Of course, the longer the phase, the less the result contain matches.

## Query weight

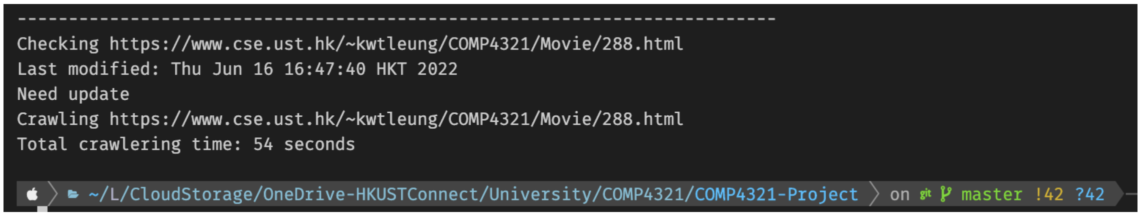
The query weight is not 1. It is calculated by the following formula:

$$w_{term/phase,query} = \frac{tf_{term,query}}{max(tf_{query})} \times \log_2(1 + \frac{N_{num\_indexed\_page}}{df_{num\_page\_contain\_term}})$$

It can show the importance of the term in the query. For example, a search query 'definition of quantum in quantum mechanics', the term 'quantum' will be larger than others in the $tf$ part. Also, the $idf$ part also indicate the likeliness of this term appear in a doc, the less it appear, the higher the weighting. Therefore, we can obtain the query weight which showing the level of importance of term in the query thus make the result more ranking more reasonable(favor the terms that are more specific and meaningful but not general).

## Crawling time is quite fast

It take less than 1 minute to crawl 300 pages. You can test about it. But please use CMD to run the crawler as JSP web interface will slow down the crawling speed :).



```
---------------------------------------------------------------------------
Checking https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/288.html
Last modified: Thu Jun 16 16:47:40 HKT 2022
Need update
Crawling https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/288.html
Total crawlering time: 54 seconds

 ~/L/CloudStorage/OneDrive-HKUSTConnect/University/COMP4321/COMP4321-Project  on git master !42 ?42
```

Figure 7: Result of crawler in the command line

## Special query are handled

1. No match of the phases in the query, e.g. "BoscoLeung"

2. One of the phase are all stop words, e.g. "the of", it will consider as no match.

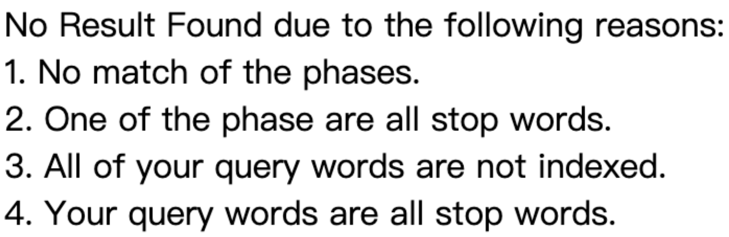
3. All of your query words are not indexed, e.g. BoscoLeung from hkustCSDepartment

4. Query words are all stop words, e.g. there is a

## Crawler in JSP web interface to index more pages

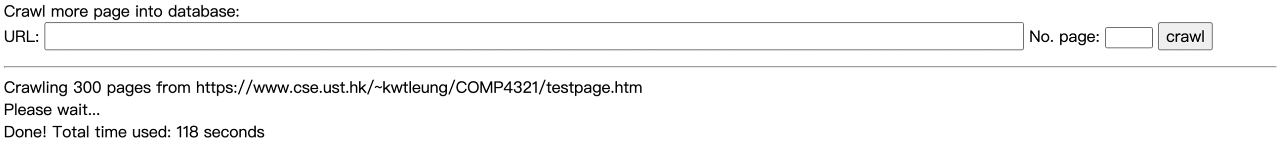
You can now use the crawler in the JSP web interface to index more pages. Even you don't have the database at first. You can now index pages from the interface.

But noticed that the crawling speed will be slower compare to that of crawling in the command line because of the JSP stuff. And the process will not be shown in the web interface. So better not to crawl too many pages at once :). Or you need to panient to wait for the result. (The page crawled will in the datebase even hard stopped.)



No Result Found due to the following reasons:
1. No match of the phases.
2. One of the phase are all stop words.
3. All of your query words are not indexed.
4. Your query words are all stop words.

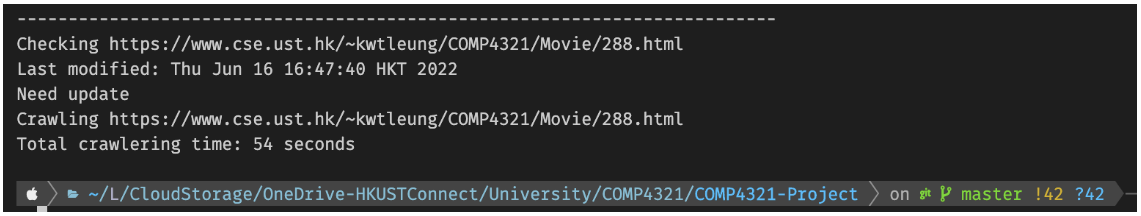
Figure 8: The web interface showing no result



Crawl more page into database:
URL: No. page:

Crawling 300 pages from https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm
Please wait...
Done! Total time used: 118 seconds

Figure 9: Demo of the crawler in JSP web interface

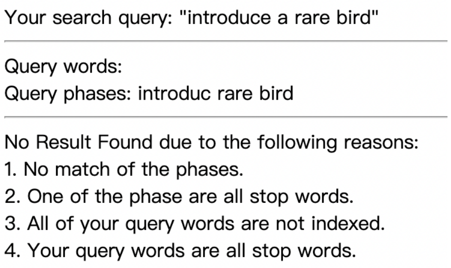


```
---------------------------------------------------------------------------
Checking https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/288.html
Last modified: Thu Jun 16 16:47:40 HKT 2022
Need update
Crawling https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/288.html
Total crawlering time: 54 seconds

 ~/L/CloudStorage/OneDrive-HKUSTConnect/University/COMP4321/COMP4321-Project  on  master !42 ?42
```

Figure 10: Demo of the crawler in the command line

Example:

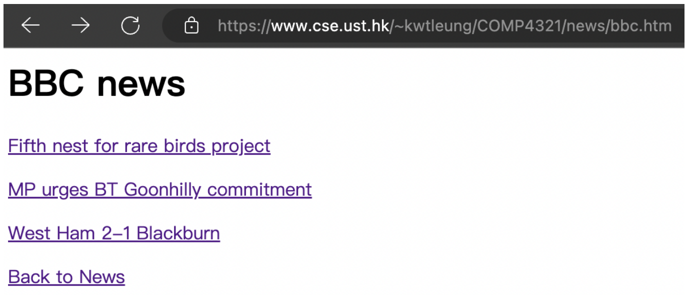


Your search query: "introduce a rare bird"

Query words:
Query phases: introduc rare bird

No Result Found due to the following reasons:
1. No match of the phases.
2. One of the phase are all stop words.
3. All of your query words are not indexed.
4. Your query words are all stop words.

Figure 11: A page that is not indexed



https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc.htm

BBC news

Fifth nest for rare birds project

MP urges BT Goonhilly commitment

West Ham 2–1 Blackburn

Back to News

Figure 12: The starting page

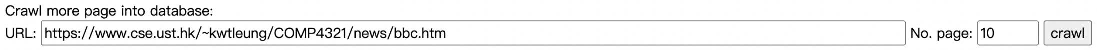
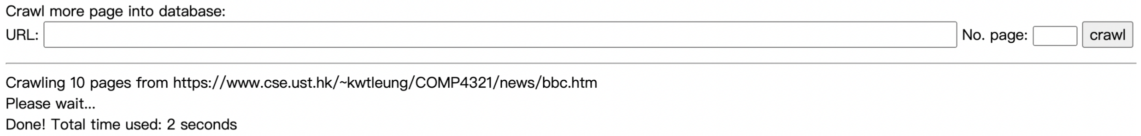
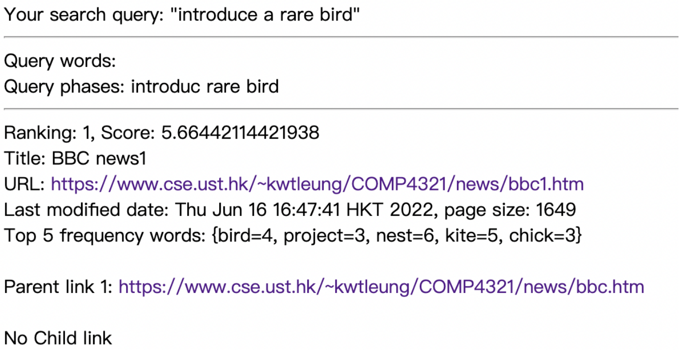

Crawl more page into database:
URL: https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc.htm No. page: 10

Figure 13: Enter the crawling information


Crawl more page into database:
URL: No. page:

Crawling 10 pages from https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc.htm
Please wait...
Done! Total time used: 2 seconds

Figure 14: After crawling


Your search query: "introduce a rare bird"

Query words:
Query phases: introduc rare bird

Ranking: 1, Score: 5.66442114421938
Title: BBC news1
URL: https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc1.htm
Last modified date: Thu Jun 16 16:47:41 HKT 2022, page size: 1649
Top 5 frequency words: {bird=4, project=3, nest=6, kite=5, chick=3}

Parent link 1: https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc.htm

No Child link

Figure 15: The page is now indexed and searchable

End of example.

# Conclusion

## Strengths

The searching time is fast even there will be 50 results to output.
The searching result is shown in a good format.
The web interface is easy to use and responsive.

## Weaknesses

### Phase search minor defect

Some time the phase search result is not good because of the linking postition of the title and the body.

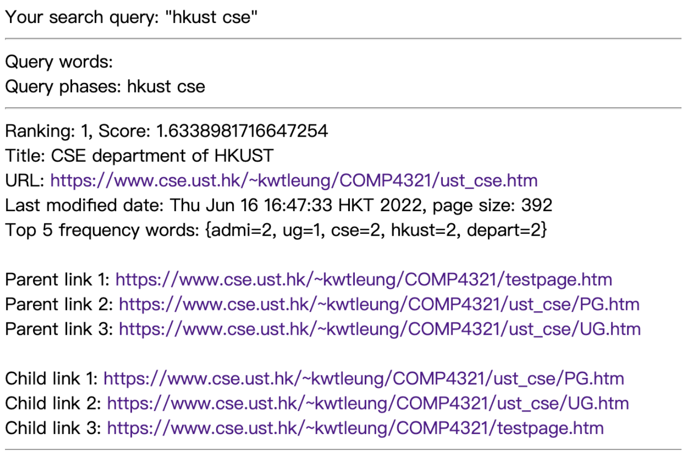
The search result may first give a seemmingly not accurate result, as no phase "hkust cse" is found. But when look into the html source code of the page. The word hkust in the last of the title and cse in the first of the body are consider as a phase as their position are in oder.

This is because in my index, there are two type of index, one is for the title and one is for the title and body. So this kind of unusual case will happen in rare cases but the effect is small as those two word is not usually to make a meaning phase.

Therefore, if I could re-implement the whold system, I will seperate the title and body into two different index. So that the phase search result will be more accurate.

### Links not utilized in the ranking

The links relationships are not used for the searching. It maybe useful to perform PageRank, HITS or other ranking algorithm to rank the result and make the result more accurate.



Your search query: "hkust cse"

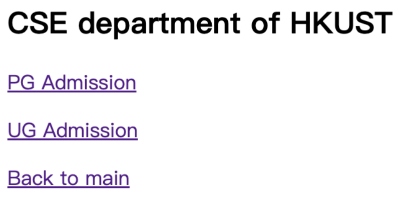
Query words:
Query phases: hkust cse

Ranking: 1, Score: 1.6338981716647254
Title: CSE department of HKUST
URL: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
Last modified date: Thu Jun 16 16:47:33 HKT 2022, page size: 392
Top 5 frequency words: {admi=2, ug=1, cse=2, hkust=2, depart=2}

Parent link 1: https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm
Parent link 2: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/PG.htm
Parent link 3: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/UG.htm

Child link 1: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/PG.htm
Child link 2: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/UG.htm
Child link 3: https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm

Figure 16: Example of bad phase search result



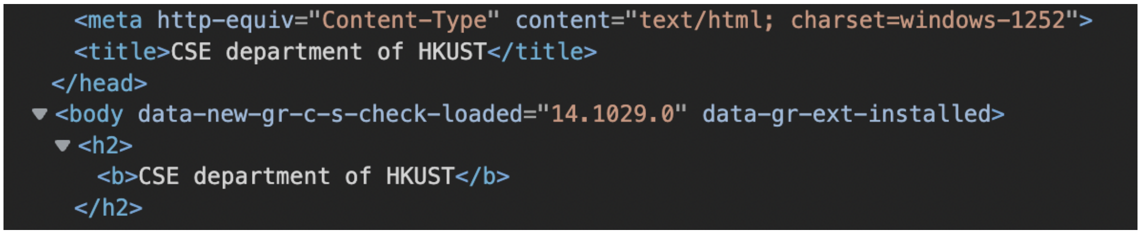
CSE department of HKUST

PG Admission

UG Admission

Back to main

Figure 17: website of the bad phase search result



```
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
    <title>CSE department of HKUST</title>
  </head>
▼ <body data-new-gr-c-s-check-loaded="14.1029.0" data-gr-ext-installed>
  ▼ <h2>
      <b>CSE department of HKUST</b>
    </h2>
```

Figure 18: Reason of bad phase search result

# Special Notice

## Word Extraction

I have set

```
sb.setLinks(false)
```

in the word extraction part, which is different from the lab.

Since if it is set to true, it will create some keywords like *httpslibraryhkusteduhkaboutushoursservicepointshoursservic* and *httpslibraryhkusteduhkhelpforalumnialumni*, it doesn't seem to make sense.

## Crawler Strategy

I have implemented the BFS strategy in this phase. And the crawler will pick the next URL according the occurrence order in the webpage.

## Page Last Modified Date and Page Size

Currently I am using the following method to get the last modified date of the page:

```
url.openConnection().getLastModified();
```

But it seems that the last modified date may missing. In this case, the last modified date will be set to

```
url.openConnection().getDate();
```

which is the date of accessing.

For the page size, I am using the following method to get the page size:

```
url.openConnection().getContentLength();
```

If the page size is missing, I will use the page size value method obtain in lab 2 instead.