建议：

1. OA需要注意的方面
2. 注意test case没有cover到的corner case handling
3. runtime和space 复杂度

Snowflake VO review:

VO1:

Use java to implement the class HashMap of Integer:

Method to implement:

put(key, value) // add a pair to hash map

get(key) // get value by given pair

remove(key) // given a key remove the pair with given key

containsKey(key) // given a key return whether the key is conatined in the HashMap

Solution:

List<List<Pair<Integer, Integer>>> map;

Use key % map.size() to get the index of a given pair.

Use 2D list to handle index collision.

Optimization: (Discussed)

Instead of 2D list, use linkedlist and redblack tree.

Review:

1. 一开始题目的要求是implement一个hashmap for any class， 思路有些混乱，然后更进一步specify说就当是一个hashmap for integer，才把思路理顺写完整个代码，怀疑是一开始的表现不佳。

2. 查过java API后发现Pair<Integer, Integer>的部分用是Entry而不是Pair，这里把java和c++的类搞混了，可能也是一个扣分点。

扣分点：

1. 概念部分：[hashcode，compare](https://www.geeksforgeeks.org/equals-hashcode-methods-java/)这种general的部分不理解。说明java的语言特性掌握不够深。
2. 中小型公司对概念的理解，包括上手能力要求高
3. remove（），怎么删掉？
   1. 时间开销比较大。linkedinlist时间开销小很多。arraylist remove开销大。

VO2:

Implement a service with two method:

1. Store document:

// not specified, assumed

List<String> document;

// document[0] is ID

// store document

Map<String, List<String>> data; // key : value -> ID : Document

// key: ID, Value: List<String> words in the document

面试技巧：

1. record down what you have discussed via comment因为面试官之后还有hiring commitee看你的代码，包括coding pad。如果你什么都没有写，没有办法backtrace。尽量留下你的思路，包括讨论出来的结果。
   1. 比如map的key和value分别代表的语义。

2. Answer questions:

Question: whether the database contains A or/and B or/and C?

- A, B, C are three words

- the query may contains 1-3 words

- return only true or false based on the final expression

- parenthesis may exist

- (A and B) or C

- (A or B) and C

面试技巧：

1. 从input，output的角度来思考。这样 有帮助你发现具体function的purpose。建议一开始就和面试官clarify。
2. 面试的考察一个重点：是否有能力转化ambiguous problem到concrete example。
3. 希望你去问一些clarifying question。找元素：什么type，storage system，有没有可能出现不符合要求的type，data volume，streaming data，batch data。

Priority:

1. Input, output
2. data type
3. storage system
4. corner case handling needed？
5. data volume

Solution:

1. Store documents

List<String> document; // document[0] is ID

// store document

Map<String, List<String>> data;

// key: ID, Value: List<String> words in the document

Whenever need to add new document, add the document by ID

2. Answer questions:

Assume have method that turn question string into boolean expression.

Search in the database for A, B and C separately and turn them into false and true directly. Then, evaluate logic expression with the true false value.

Evaluate the expression in a stack of string, whenever we encounter the situation that another string of word(A/B/C) is an input and the top of stack is an operator(and/or). Pop the top two string out and evaluate the expression(word1 operator word2), push the result back to the result and keep going, untile there is only one element in the stack, which should be the final True or False of the query.

Optimization：

Instead of string use store int into stack and use map to record the corresponding string of each int to speed up evaluation.

Review:

感觉这题整体回答的思路都比较乱，感觉自己的解法可能并不是面试官想问的内容？因为在我的回答里面好像第一问过分简单，而第二问又过分复杂了。感觉另外一个比较大的问题是类型上为了答题简便很多时候直接assume了string，但是同时又导致在evaluate的时候涉及了很多类型转换，很容易错。优化方案也给的不明确。同时在evaluate expression（stack）的这个部分感觉算法感觉讲的不够清晰，可能也是一个扣分点。

Project Description

**# Eulerity Hackathon Challenge**

Congratulations on making it to this stage of Eulerity's interview process! In this folder is a project for a partially built web application whose goal is to crawl a provided URL and pick out the images from it. This README will provide more information about the goals of the project, its structure, and setup and submission instructions.

**## ImageFinder Goal**

The goal of this task is to perform a web crawl on a URL string provided by the user. From the crawl, you will need to parse out all of the images on that web page and return a JSON array of strings that represent the URLs of all images on the page. [Jsoup](https://jsoup.org/) is a great basic library for crawling and is already included as a maven dependency in this project, however you are welcome to use whatever library you would like.

**### Required Functionality**

We expect your submission to be able to achieve the following goals:

- Build a web crawler that can find all images on the web page(s) that it crawls.

- Crawl sub-pages to find more images.

- Implement multi-threading so that the crawl can be performed on multiple sub-pages at a time.

- Keep your crawl within the same domain as the input URL.

- Avoid re-crawling any pages that have already been visited.

**### Extra Functionality**

No individual point below is explicitly required, but we recommend trying to achieve some extra goals as well, such as the following:

- Make your crawler "friendly" - try not to get banned from the site by performing too many crawls.

- Try to detect what images might be considered logos.

- Show off your front-end dev skills with Javascript, HTML, and/or CSS to make the site look more engaging.

- Any other way you feel you can show off your strengths as a developer 😊

**\*\*PLEASE do not send us a submission with only a basic JSoup crawl and only a couple lines of code.\*\*** This is your chance to prove what you could contribute to our team.

You have one week to work on the submission from the time when you receive it. To submit you assignment, zip up your project (`imagefinder.zip`) and email it back to me. **\*\*Please include a list of URLs that you used to test in your submissions.\*\*** You should place them in the attached `test-links.txt` file found in the root of this project.

**## Structure**

The ImageFinder servlet is found in `src/main/java/com/eulerity/hackathon/imagefinder/ImageFinder.java`. This is the only provided Java class. Feel free to add more classes or packages as you see fit.

The main landing page for this project can be found in `src/main/webapp/index.html`. This page contains more instructions and serves as the starting page for the web application. You may edit this page as much as it suits you, and/or add other pages.

Finally, in the root directory of this project, you will find the `pom.xml`. This contains the project configuration details used by maven to build the project. If you want/need to use outside dependencies, you should add them to this file.

**## Running the Project**

Here we will detail how to setup and run this project so you may get started, as well as the requirements needed to do so.

**### Requirements**

Before beginning, make sure you have the following installed and ready to use

- Maven 3.5 or higher

- Java 8

- Exact version, **\*\*NOT\*\*** Java 9+ - the build will fail with a newer version of Java

**### Setup**

To start, open a terminal window and navigate to wherever you unzipped to the root directory `imagefinder`. To build the project, run the command:

*>*`mvn package`

If all goes well you should see some lines that end with "BUILD SUCCESS". When you build your project, maven should build it in the `target` directory. To clear this, you may run the command:

*>*`mvn clean`

To run the project, use the following command to start the server:

*>*`mvn clean test package jetty:run`

You should see a line at the bottom that says "Started Jetty Server". Now, if you enter `localhost:8080` into your browser, you should see the `index.html` welcome page! If all has gone well to this point, you're ready to begin!

**## Submission**

When you are finished working on the project, before zipping up and emailing back your submission, **\*\*PLEASE RUN ONE LAST** `mvn clean` **COMMAND TO REMOVE ANY UNNECESSARY FILES FROM YOUR SUBMISSION\*\***. Please also make sure to add the URLs you used to test your project to the `test-links.txt` file. After doing these things, you may zip up the root directory (`imagefinder`) and email it back to us.

**## Final Notes**

- If you feel you need more time to work, you are free to ask for it.

- If you are having any trouble, especially with the setup, please reach out and we will try to answer as soon as we can.

- The ideas listed above on how to expand the project are great starting points, but feel free to add in your own ideas as well.

- Try to follow some good-practice principles when working on your code, such as meaningful and clean variable/method names and other good coding practices.

- The code we have provided is to allow you to hit the ground running. You are free to use whatever web service you would like (as long as you use Java 8 and it is runnable from the command line).

- We look forward to seeing what you can do, so good luck and have fun 😊

package com.eulerity.hackathon.imagefinder;

import org.jsoup.Connection;

import org.jsoup.Jsoup;

import org.jsoup.nodes.Document;

import org.jsoup.nodes.Element;

import org.jsoup.select.Elements;

import java.io.IOException;

import java.net.MalformedURLException;

import java.net.URL;

import java.util.HashSet;

import java.util.List;

import java.util.ArrayList;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.TimeUnit;

public class WebCrawler {

int dfsDepthLeve = 0;

private int failedURLNum = 0;

private int failedImgSrcNum = 0;

private final String userAgent = "Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;%20x64)%20AppleWebKit/537.36%20(KHTML,%20like%20Gecko)%20Chrome/58.0.3029.110%20Safari/537.3";

private final String startURL;

private final String domainName;

public HashSet<String> visitedURLs;

public HashSet<String> imgSrcs;

private List<String> imgSrcsPerson;

private List<String> imgSrcsLogo;

*/\*\**

*\* Constructor for WebCrawler class.*

*\* @param start\_url the start URL for web crawling.*

*\* @throws MalformedURLException if the URL is malformed.*

*\*/*

public WebCrawler(String start\_url) throws MalformedURLException {

this.startURL = start\_url;

URL url = new URL(start\_url);

this.domainName = url.getProtocol()+"://" +url.getHost();

this.visitedURLs = new HashSet<>();

this.imgSrcs = new HashSet<>();

this.imgSrcsPerson = new ArrayList<>();

this.imgSrcsLogo = new ArrayList<>();

}

*/\*\**

*\* Finds all URLs in the same domain with DFS starting from the start URL.*

*\*/*

public void getAllURLs() throws InterruptedException {

System.*out*.println("Start to collect URLS.");

findURLsWithDFS(this.startURL);

System.*out*.println("All URLs in the same domain collected.");

}

*/\*\**

*\* A helper function to recursively find all URLs in the same domain with DFS starting from the input URL.*

*\* @param url the input URL to be visited.*

*\*/*

private void findURLsWithDFS(String url) throws InterruptedException {

this.dfsDepthLeve++;

// stop recursion when the level is too deep

if (this.dfsDepthLeve>100) {

return;

}

if (url.contains("login")) {

return;

}

if (!visitedURLs.contains(url)) {

try {

Connection conn = Jsoup.*connect*(url).userAgent(this.userAgent);

Document document = conn.get();

Elements subUrls = document.select("a[href^=\""+this.domainName+"\"]");

visitedURLs.add(url);

for (Element subUrl : subUrls) {

String address = subUrl.attr("abs:href");

findURLsWithDFS(address);

this.dfsDepthLeve--;

}

} catch (IOException e) {

// uncomment below to print out error message

// System.err.println(e.getMessage());

// System.err.println(url);

this.failedURLNum++;

}

}

// System.out.println("Current dfs level: "+this.dfsLeve);

// Delay request to avoid send too many requests in a short period of time

Thread.*sleep*(100);

}

*/\*\**

*\* Collects all image sources from all the collected URLs sequentially.*

*\* @throws IOException if an I/O error occurs.*

*\*/*

public void getAllImgs() throws IOException, InterruptedException {

System.*out*.println("Start to collect images sources from all of the collected URLs.");

for (String url : visitedURLs) {

getImgs(url);

}

System.*out*.println("All image sources collected.");

}

*/\*\**

*\* Collects all image sources from all the collected URLs with multiple threads.*

*\* @throws IOException if an I/O error occurs.*

*\* @throws InterruptedException if a thread is interrupted while waiting.*

*\*/*

public void getAllImgsMultiThread() throws IOException, InterruptedException {

System.*out*.println("Start to collect images sources from all of the collected URLs.");

int numThreads = Runtime.*getRuntime*().availableProcessors();

ExecutorService executorService = Executors.*newFixedThreadPool*(numThreads);

for (String url : visitedURLs) {

executorService.submit(()-> {

try {

getImgs(url);

} catch (IOException e) {

System.*err*.println(e.getMessage());

System.*err*.println(url);

} catch (InterruptedException e) {

throw new RuntimeException(e);

}

});

}

executorService.shutdown();

executorService.awaitTermination(Long.*MAX\_VALUE*, TimeUnit.*NANOSECONDS*);

System.*out*.println("All image sources collected.");

}

*/\*\**

*\* Helper method that scrapes image sources from a given URL and adds them to the instance's imgSrcs HashSet.*

*\* @param url The URL to scrape image sources from.*

*\* @throws IOException if an I/O error occurs.*

*\*/*

private void getImgs(String url) throws IOException, InterruptedException {

String postfix = url.substring(url.length() - 4).toLowerCase();

// if the url itself is an image address, we store it as source and return

if (postfix.equals(".jpg")||postfix.equals(".png")) {

this.imgSrcs.add(url);

return;

}

try {

Connection conn = Jsoup.*connect*(url).userAgent(this.userAgent);

Document document = conn.get();

Elements images = document.select("img");

for (Element image : images) {

String imgSrc = image.attr("abs:src");

addToImgSrcs(imgSrc);

}

} catch (IOException e) {

// uncomment below to print out error message

// System.err.println(e.getMessage());

// System.err.println(url);

this.failedImgSrcNum++;

}

// Delay request to avoid send too many requests in a short period of time

Thread.*sleep*(100);

}

*/\*\**

*\* Helper method that adds a given URL to the instance's imgSrcs list in a thread-safe manner.*

*\* @param url The URL to add to the HashSet.*

*\*/*

private synchronized void addToImgSrcs(String url){

this.imgSrcs.add(url);

}

*/\*\**

*\* Reports the total number of visited URLs and scraped image sources to the console.*

*\*/*

public void reportResult() {

System.*out*.println("Find "+this.visitedURLs.size()+" web pages and "+this.imgSrcs.size()+" images in total.");

System.*out*.println("Fail to scrape "+this.failedURLNum+" URLs and fail to scrape "+this.failedImgSrcNum+" images.");

}

*/\*\**

*\* Main method that initializes a WebCrawler instance with a starting URL, test methods above*

*\* @param args command-line arguments (not used)*

*\* @throws IOException if an I/O error occurs.*

*\* @throws InterruptedException if the main thread is interrupted while waiting for the completion of the image scraping task.*

*\*/*

public static void main(String[] args) throws IOException, InterruptedException {

// get all URLs in the same domain

long startTime = System.*currentTimeMillis*();

String url\_string = "https://jacksonparklic.com/\n";

WebCrawler wc = new WebCrawler(url\_string);

wc.getAllURLs();

long endTime = System.*currentTimeMillis*();

long executionTime = endTime - startTime;

System.*out*.println("DFS\n\tExecution time: " + executionTime + " milliseconds.");

/\* check visitedURLs result

for (String url : wc.visitedURLs) {

System.out.println(url);

}\*/

startTime = System.*currentTimeMillis*();

if (wc.visitedURLs.size()>50) {

try {

wc.getAllImgsMultiThread();

} catch (InterruptedException e) {

throw new RuntimeException(e);

}

} else {

wc.getAllImgs();

}

endTime = System.*currentTimeMillis*();

/\* check imgSrcs result

for (String imgSrc : wc.imgSrcs) {

System.out.println(imgSrc);

}\*/

executionTime = endTime - startTime;

System.*out*.println("Get ImgSrcs\n\tExecution time: " + executionTime + " milliseconds.");

wc.reportResult();

}

}

建议：

1. 功能倒是不用那么担心。general test case能够handle。
2. corner case，前端后端能够handle。包括exception handling。
3. 格式：
   1. const需要define。
   2. 用过超过一次的，用variable存储起来
   3. system out这种command不要放在comment里面
   4. comment写一些白话。正常的英语，不是编程语言。帮助面试官reviewer理解
4. 程序的运行时间。你测试一下。

Googlecloud computer vision api:

<https://cloud.google.com/vision>

1. freelancer工作。