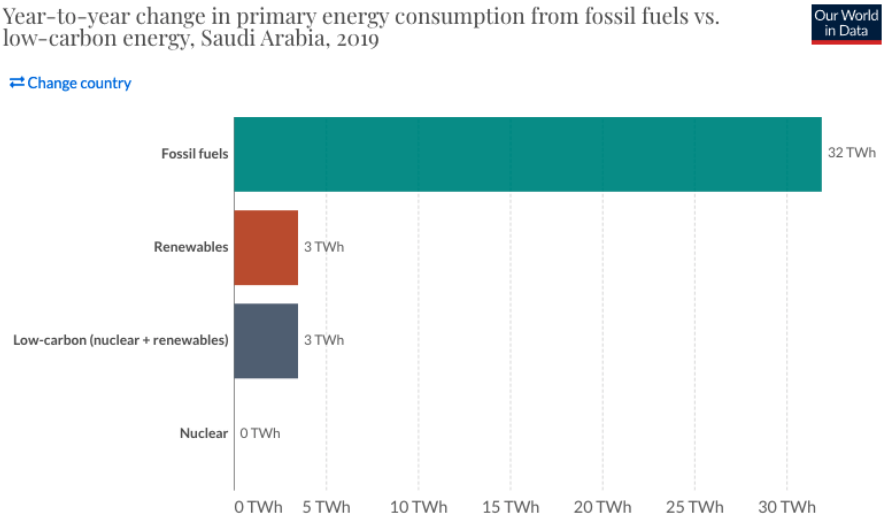
CS210: Term project 2 [211] Weight: 10% Due: End of Monday December 6, 2021 (23:59)

Experimental evaluation & time-analysis of various queries on a choice of Data Structures for United Nations Sustainable Development Goals (SDG) – theme 13 datasets

The United Nations Sustainable Development Goals (SDGs) are targets for global development adopted in September 2015, set to be achieved by 2030. All countries of the world have agreed to work towards achieving these goals. At its heart are the 17 (SDGs), which are an urgent call for action by all countries - developed and developing - in a global partnership. They recognize that ending poverty and other deprivations must go hand-in-hand with strategies that improve health and education, reduce inequality, and spur economic growth – all while tackling climate change and working to preserve our oceans and forests.

One of these 17 goals is the goal number 13: "Take urgent action to combat climate change and its impacts". Atmospheric concentrations of the major greenhouse gases continued to increase despite the temporary reduction in emissions in 2020 related to measures taken in response to the COVID-19 pandemic. The six years from 2015 to 2020 are likely to be the warmest on record. Climate change is driven by our CO2 emissions plus emissions of other greenhouse gases such as methane and nitrous oxide. Energy accounts for around three-quarters of total greenhouse gas emissions – the other quarter coming from agriculture and land-use change. The global energy generation, consumption and the CO2 emissions go hand in hand. Burning fossil fuels – coal, oil and gas – emit carbon dioxide (CO2). To reduce global emissions, it's essential that we shift our energy systems away from fossil fuels towards low-carbon sources of energy. In recent times, Saudi Arabia has taken leadership in reducing its power production reliance on fossil fuels (Oil and gas) and initiating the green energy initiatives. Current it is producing 6 TWh of energy by the use of renewables and low-carbon energy sources.

This is a group-based project that requires you to develop a software using various Data Structures to process the datasets provided. Your team will develop data-structures and run various queries on these datasets to analyze the performance of the data structures and algorithms studied in this class.



1. Datasets

Four datasets are provided:

Dataset	Information	Size
A	Per Capita Electricity consumption	5520 records
B	Global Energy Consumption per capita (kWh)	8962 records
C	Annual CO2 emission per country	23950 records
D	Annual CO2 generation by source	65854 records

All datasets are arranged in rows and columns, saved in a Comma Separated Values file (CSV). Each dataset has at least 4 columns, where the first three columns are always the same. These are: Country Name (String), Country Code (String), year (int) and value(s) (doubles). Only in dataset D, you can see additional columns.

2. Data Structures

This project works around Hash tables. You will build two kinds of hash tables.

Hash table - Chaining
Hash table – Displacement (Linear Probing)

The size of the hash-table is variable based on the experiment (see section 3). The size is determined by the load-factor.

Load-factor: If *n* is the total number of buckets and *k* is the number of buckets that have data; then Load-factor is *k*/*n*. Example; if *n* is 10 and *k* is 7, then load factor is 0.7.

You will write a Node class to store information. Each node will store information provided in a row (record). You will write a Hashtable class to implement the Hash table or various sizes.

3. Use of Hash functions

Each dataset has three columns that are common, i.e. Country Name (S), Country Code (C) and year (Y).

The first hash function *h(n)* works with the String S.

$$h(s) = \left[\sum_{i=0}^{n-1} s[i] \,.\, 31^{n-1-i}\right] \; mod \; p$$

Example; Country Name s is string “Mali”, here the string length *n* = 4. Assume that *p* is a prime number 383.
s[0] is ‘M’ with ASCII Code 77
s[1] is ‘a’ with ASCII Code 97
s[2] is ‘l’ with ASCII Code 108
s[3] is ‘i’ with ASCII Code 105

then

$h(s) = 77 * 31^{4-1-0} + 97 * 31^{4-1-1} + 108 * 31^{4-1-2} + 105 * 31^{4-1-3} \% 383$
$h(s) = 77 * 31^3 + 97 * 31^2 + 108 * 31^1 + 105 * 31^0 \% 383$
$h(s) = 77 * 31^3 + 97 * 31^2 + 108 * 31^1 + 105 * 31^0 \% 383$
$h(s) = 2293907 + 93217 + 3348 + 105\% 383$
$h(s) = 2390577 \% 383$
$h(s) = 274$

“Mali” will hash to bucket 274.

The second hash function *f(n)* works with integer values where f(n) = n % p. *p* is a primer number.

4. Writing the program

You will write a java program that read data from the input CSV file, processes information, and write to an output CSV file. Your program will run the three events, multiple number of times.

1. Load/Read data from a CSV input file into the memory.
2. Search and remove selected items from the memory; and
3. Write the result to a CSV output file.

You will write a program to “time” the three events. Use java’s System.nanoTime() or System.currentTimeMillis() as appropriate, to measure the time elapsed.

The program reads information from the command line as follows:

```
java -jar project2 <input file> <output file> <column> <hash table size n> <collision
resolution> <prime number p> <remove keys>
```

Example:

```
java -jar project2 file1.csv output.csv 1 1000 1 13 Mali
```

project2 reads file1.csv and writes to output.csv. It indexes the value based on column 1 (Country Name); it creates a hash table of size 1000 with p size 13. It removes all instances of “Mali” from the Hash table. Here is a sample run:

```
java -jar project2 file1.csv output.csv 1 1 1000 13 Mali
Reading data …
Total time to read data: 1.099812 seconds
Searching and removing Mali
Total time to remove data: 0.001582 seconds
Writing data to output file
Total time to write data: 1.201220 seconds.
```

For column 1 and 2, where the data is a string, hash function *h(n)*. For column 3, use second hash function *f(n)*. Collision resolution 1 is with Displacement (Linear probing method). 2 is with Chaining.
Note: If the dataset index values are empty, insert and replace it with 0.

5. Experimental Evaluation

As we already studied, the run times for Hash-tables are theoretically O(1). It is time to empirically analyze the performance of hash tables, the impact of hash table size on collisions, the choice of a good hash function etc. You will conduct four experiments, each with a dataset.

Tell us about the computer you will use for experimentation.

CPU Cores and Speed (GHz)	
Available RAM (GB) – check Windows Task Manager	

Experiment A:

Read data set A. Index data based on the "year" into hash table using hash function *f(n)*. Remove "ALL" data for year **2012** only.

	Linear probing method			Chaining method		
Time	N = 8000 P = 11	N = 8000 P = 37	N = 8000 P = 919	N = 8000 P = 11	N = 8000 P = 37	N = 8000 P = 919
Read						
Remove						
Write						

Experiment B:

Read data set B. Index data based on the "year" into hash table using hash function *f(n)*. Remove "ALL" data for year **2002** only.

	Linear probing method			Chaining method		
Time	N = 100000 P = 11	N = 100000 P = 37	N = 100000 P = 919	N = 100000 P = 11	N = 100000 P = 37	N = 100000 P = 919
Read						
Remove						
Write						

Experiment C:

Read data set C. Index data based on the "Country Code" into hash table using hash function *h(n)*. Remove "ALL" data for countries: **WLF, JOR, CYP, BMU and ZMB**.

	Linear probing method			Chaining method		
Time	N = 24000 P = 313	N = 24000 P = 1913	N = 24000 P = 7919	N = 24000 P = 313	N = 24000 P = 1913	N = 24000 P = 7919
Read						
Remove						
Write						

Experiment D:

Read data set D. Index data based on the "Country Name" into hash table using hash function *h(n)*. Remove "ALL" data for countries: **Poland, Morocco, Saudi Arabia, Senegal, Vietnam and "Wallis and Futuna**".

	Linear probing method			Chaining method		
Time	N = 100000 P = 313	N = 100000 P = 1913	N = 100000 P = 7919	N = 8000 P = 313	N = 8000 P = 1913	N = 8000 P = 7919
Read						
Remove						
Write						

6. Observations:

1. What do you observe in Experiment 1 comparing the Displacement method vs. Chaining? Explain why the results appear as they do?

2. Experiment 2 increases the size of the Hash table. The hash table size increased significantly. What is the load factor in your hash tables? Do you see any difference comparing the two methods?

3. Compare results from Experiment 1 and 2. What changed. Explain why the results change?

4. In experiment 3, what is the load factor in your hash tables? What is the difference in results for differing P values.

5. Compare results from Experiment 2 and 3 for the chaining method. Can we compare the results given that the P values are different? Why or why not?

6. What is the load factor in Experiment D for both hash tables? Are the results comparable for P=313?

7. Conclusions

The conclusions should provide a concise but deep analysis of what you have experienced based on this experimentation.

Rubrics for Evaluation:
You are allowed to work in a group of up to 2 students. The score earned would be given to all team members equally. Your work's evaluation would be based on Quality of report, Code inspection and successful execution of N number of test cases. The instructor reserves the right to determine the scores of each test case. Following is the distribution of scores for this project:

Successful execution of program based on requirement specifications	4
Experimental work	4
Quality of Report / Conclusions	2
Total	10

Up to 1 **Bonus point** can be awarded for students using suitable visualization techniques to better present their report/conclusions.

Code Inspection:
The code would be inspected by the instructor. The instructor would determine the score to be given for code inspection. Generally, a readable code (indentation, clear scope definition) is required. For this project, there are no limitations on time and memory usage.

Submission:
You need to submit THREE files, a Jar file, all code in a zip file and your report as PDF. Beware (attach only .java files, .class files cannot be read). The file names must be your id. In the report, clearly identify and write the student ID and Full name for all the team members.

Example:
21001001.jar, 21001001.zip and report.pdf (or word)
All submissions through LMS (http://lms.psu.edu.sa)

Submission Dead-Line:
The submission deadline is final. Late Submissions will be awarded ZERO points.

Plagiarism:
Source code plagiarism is copying or reproducing the same source code without properly acknowledging the original or the genuine source code creator. That includes adapting minimal, moderate reproduction of other's work or including fragments of the original code in your own code [Source: CopyLeaks]. Submitting someone else's code as your code is a classical plagiarism case.

Pair programming is an agile software development technique, where two persons (students in a group), work together on one or more machines (terminals). Code submitted by one of more students within this group is not plagiarism as long as the work is owned by the group members. This involves, discussions about the project, low-level and high-level design of the software, writing of actual code, testing of the program against various input, writing of the report, etc.

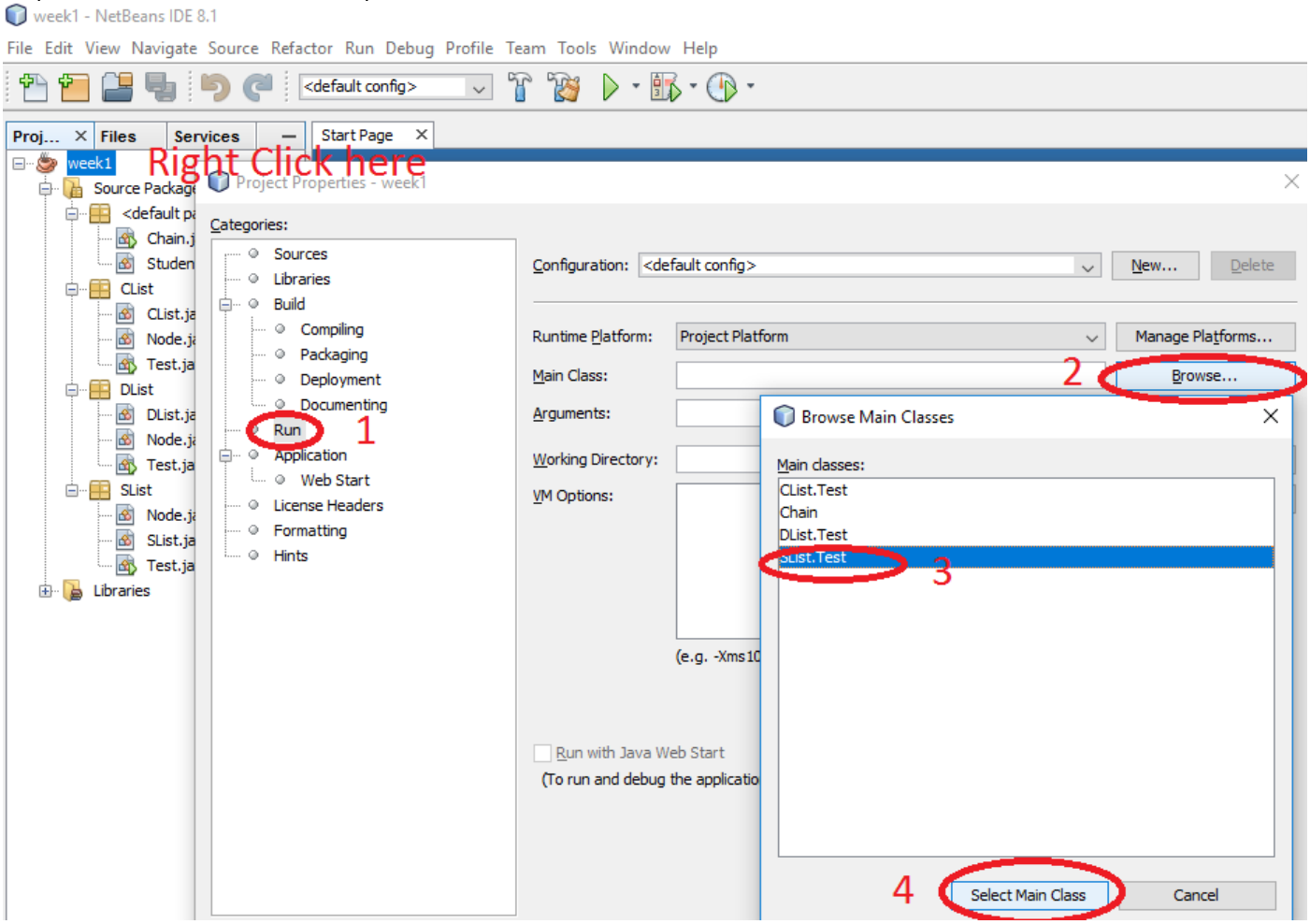
Sharing your code with the intent of helping other students is Plagiarism. As the department policy, the students who shared the code, and the one who re-used the code, will both receive ZERO score in the project. All code submitted by the students would be compared using advanced AI based tools. In the event of suspicion, the instructors reserve the right to interview the students.

First offence: Students caught in plagiarism case will fail the assessment. An academic warning is given by the instructor.

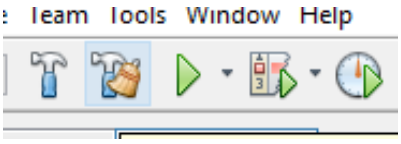
Second offence: Students involved, Fail the course. Students are reported to the Dean of the College. An academic dishonesty and professional misconduct case are filed against the student in the Student Affairs office. This can result in suspension and permanent expulsion of the student from the university.

Note on creating jar files and running programs with command-line arguments:

In Netbeans project do a clean build of your project. With default settings, jar files are created in your project folder under **dist** folder.
Step 0. In Netbeans project, open project properties dialog box by right clicking on the project.
Step 1. Select Run from the options
Step 2. Select the Main Class. Click browse, a small popup window will list all class containing main method.
Step 3. Select the class you want the jar file to execute first.
Step 4. Click Select Main Class and press OK.



Step 5: Do a clean build of your project. With default settings, jar file(s) are created in your project folder under dist folder.



You can watch a video on How to setup your environment variables to test your programs from terminal.
You can watch a video on How to create a jar file in your Netbeans project.
You can watch a video on How to work with command line arguments from Terminal.
You can watch a video on How to work with command line arguments in Netbeans.
You can also watch useful videos and tutorials on installation etc.

Note on verifying jar files.

1. Open terminal window by clicking on start button and typing **cmd**.
2. Relocate to project folder. If your project is in folder **C:>project** then type **cd \project\dist**
3. Type **java –jar <jar file>**

If you get an error on "**main class not found**" this means you need to setup paths in your environment. Follow directions here:

1. On Windows Explorer, right click on "My Computer" or "This PC"
2. Select Properties
3. Select Advanced System properties

4. In “Advanced Tab” locate button “Environment Variables” and click it.
5. In system variables, select path. Click edit
6. Add the path to your jdk\bin folder
7. Add a path to your jre\rt.jar file
8. Press ok, and you are done!

Important Notes:

- It is the student’s responsibility to check/test/verify/debug the code before submission.
- It is the student’s responsibility to check/test/verify all submitted work (including jar files)
- It is the student’s responsibility to verify that all files have been uploaded to the LMS.
- After an assignment/project has been graded, re-submission with an intention to improve an assignments scores will not be allowed.
- After the assignment/project has been graded, the instructor will post test-cases used for grading on the website.
- The Instructor has the right to share project execution reports that may have been auto-generated on the course website.