Indian Institute of Technology Kanpur
CS771 Introduction to Machine Learning

Instructor: Purushottam Kar
Date: October 04, 2024
Total: 100 marks

ASSIGNMENT 1

1 What should I submit, where should I submit and by when?

Your submission for this assignment will be one report PDF (.pdf) file and one code ZIP (.zip) file. Instructions on how to prepare and submit these files are given below.

Assignment Package: `https://www.cse.iitk.ac.in/users/purushot/courses/db/2024-25-a/material/assignments/assn1.zip`
Deadline for all submissions: October 27, 2024, 9:59PM IST
Report Submission: on Gradescope
Code Submission: `https://forms.gle/7hsAyTw4Q7hWPzQdA`
Code Validation Script: `https://colab.research.google.com/drive/1VNaAmgPzJ-fagaRZ_yJ_jLxyG29jQqcj?usp=sharing`
There is no provision for "late submission" for this assignment

2 Alumni Hunt 100 marks

Your Data. Deebo has data of IITK Alumni – for each alumnus, we have three fields available:

1. id (INTEGER, PRIMARY KEY, UNIQUE, NOT NULL): this will always be a 6 digit integer denoting the alumnus' Aadhar number. This number has no relation to the alumnus' name or year of graduation – it is merely a unique identifier. id values need not be contiguous.
2. name (TEXT, NOT NULL): this is the name of the alumnus and will contain only lower-case Latin alphabet characters i.e. a – z. Names contain at least 4 and at most 10 characters.
3. year (INTEGER, NOT NULL): this is the year of graduation of the alumnus is guaranteed to be betweeen 1900 and 2100 (endpoints included). However a specific dataset may only have years in a smaller range say between 1947 and 2047.

Your Task. You need to build an engine to answer simple SQL-like queries on the data. Specifically, you need to do the following four things:

1. Create one or more indices (these could be hash tables, sorted lists, tries, B-trees, B+-trees etc) that speed up query processing time while not taking up too much memory.
2. Calculate statistics on the data that help you decide which index to use and how
3. Decide on the disk layout of the tuples i.e. in which order should the tuples be placed on disk.

4. Given an SQL query, use your stats and indices to figure out which tuples satisfy the query and tell us where those tuples are stored in your disk layout.

The following enumerates 3 parts of the assignment. Parts 1 and 2 need to be answered in the PDF report. Part 2 needs to be answered in the Python files.

1. Describe which indices did you build, which algorithms did you use to build those indices, which statistics did you compute and how did you perform disk layout.
2. Describe how query processing was done, any optimizations e.g. speedy computation for special types of queries.
3. Submit code that implements your above solution. Your submission must contain two Python files named `index.py` and `execute.py` as described above.

Implementation Details. Your submission must contain two Python files named `index.py` and `execute.py`

1. The file `index.py` must contain a method called `my_index()` (and possibly others). Alumni data will be sent to the `my_index()` method as a list of Python tuples with datatypes (int, str, int).
2. `my_index()` must return two objects
   (a) disk: this should be a single list containing id values in some order. We will take this list and store tuples in that order on our virtual disk.
   (b) idx_stat: this should be an object (list, dict, tuple etc) containing all your indices and statistics
3. The file `execute.py` must contain a method called `my_execute()` (and possibly others). We will send SQL queries to the `my_execute()` method as well as the idx_stat object that your `my_index()` returned.
4. `my_execute()` must return the disk location of all tuples that satisfy the query.
5. SQL queries will be sent to the `my_execute()` method in the form of a list of predicates. Each SQL query will be of the form `SELECT id FROM tbl WHERE <clause>`. Specifically, only id will be selected and there will be no ORDER BY or GROUP BY or HAVING clauses. The WHERE clause will be one of three types:
   (a) year-only WHERE clause: this clause will have a single predicate and will be of the form `WHERE year <cmp> <val>` where `cmp` is a comparator that can be either `=` or `<=` or `>=` and `val` will always be a value between 1900 and 2100 (end points included). Such a clause will be sent to the `my_execute()` method as a list of list of strings of the form `[[ 'year', 'cmp', 'val' ]]`
   (b) name-only WHERE clause: this clause will have a single predicate and will be of the form `WHERE name <cmp> <val>` where `cmp` is a comparator that can be either `=` or `LIKE`. If the comparator is `=`, then `val` will be a string of lower-case Latin characters i.e. a – z having between 4 and 10 characters. If the comparator is `LIKE`, then `val` will be a string of lower-case Latin characters i.e. a – z having between 1 and 10 characters followed by a `%` character at the end. Such a clause will be sent to the `my_execute()` method as a list of list of strings of the form `[[ 'name', 'cmp', 'val' ]]`

(c) conjunctive WHERE clause: this clause will have two predicates and will be of the form `WHERE n_pred AND y_pred` where `n_pred` will be of the form `name <cmp> <val>` as described above and `y_pred` will be of the form `year <cmp> <val>` as described above. Such a clause will be sent to the `my_execute()` method as a list of list of strings of the form `[[ 'name', 'cmp', 'val' ],[ 'year', 'cmp', 'val' ]]`

(d) To be sure, in case of a conjunctive WHERE clause, there will always be two predicates – one dealing with name and one dealing with year. There wont be two name predicates or two year predicates or more than two predicates in the WHERE clause.

Useful things to note.

1. The disk layout should contain only id values, not names or years.
2. It is not necessary that every tuple must be stored just once on disk. You may choose to store tuples in multiple orders on disk – say once sorted according to name and another time sorted according to year. This will increase your disk space but may reduce seek time (see below for evaluation policy).
3. The `my_execute()` method must return the disk locations of tuples, not tuples themselves.
4. Be careful if you are storing tuples multiple times on disk. For a given query, just one of those copies may need to be returned. If you return multiple copies of the same tuple, you will get less marks due to the intersection-over-union metric.

Prohibitions and Internet Resource Usage.

1. The files `index.py` and `execute.py` **must not include any modules**, not even numpy or sklearn. Only basic Python functionality must be used to solve the problem.
2. The files `index.py` and `execute.py` **must not do any file access and must not establish database or internet connections**.
3. Use of tuples, lists, sets is allowed, as are basic Python methods such as `sorted()`, `slice()`, `range()`, zip()+ etc (see `https://www.w3schools.com/python/python_ref_functions.asp` for example). However, use of `compile()`, `open()`, `eval()`, `exec()` methods **is strictly prohibited.**
4. The use of **Python dictionaries is prohibited** since they trivialize certain search problems. The use of direct searching methods such as `dict()`, `hash()`, `filter()` **is also prohibited**.
5. **The use of modules or prohibited basic Python features or methods will result in the submission getting rejected and may have to be resubmitted with penalties.**
6. In case you are confused if a certain basic Python method is allowed or not, please clarify with the instructor.
7. You are allowed to refer to textbooks, internet sources, research papers to find out more about indexing techniques, statistics calculation and algorithms therefor. However, if you do use any such resource, cite it in your PDF report. There is no penalty for using external resources but claiming someone else's work (e.g. a book or a research paper) as one's own work without crediting the original author will attract penalties.

Evaluation Your submission will be evaluated as follows. **Please take a close look at the Google Colab evaluation file as well as the dummy files provided with the assignment package. Inspecting those should clarify any doubts you may have about input/output formats, evaluation technique etc.**

1. Report (20 marks): description of indexing, statistics, layout (10 marks) and query optimization techniques (10 marks) used in the submission.
2. Code Evaluation (80 marks): this has several components
   (a) Index build time (15 marks): this is the amount of wall-clock time your `my_index()` method spends computing the indices, statistics and memory layout.
   (b) Disk size (10 marks): this is the number of tuples you have requested to be stored on the virtual disk. Recall that you may store a given tuple multiple times on disk, say to reduce seek time.
   (c) Index size (10 marks): we will take your idx_stat object, pickle it and store it on an actual disk and see how large is it.
   (d) Query indexing time (20 marks): this is the amount of wall-clock time your `my_execute()` method spends processing the query and returning a list of virtual disk locations.
   (e) Disk seek time (20 marks): this is the time spent seeking the virtual disk locations in the order in which they were returned by the `my_execute()` method. Please see below on how is seek time calculated.
   (f) Disk read time (5 marks): this is the time spent reading data from the virtual disk locations specified by the `my_execute()` method. This will simply be the number of disk locations returned by the method.
   (g) Intersection-over-union score (multiplier): The list of tuples returned by the `my_execute()` method and the gold answer will both be computed. Next, the intersection and union of these sets would be computed. The size of the intersection divided by the maximum of the size of the union and number of tuples returned will give us a score which will be a number between 0 and 1.
   (h) The above scores, times, sizes will be calculated over several queries and multiple runs and averages would be taken. Once the average score from the first 6 components have been added to get a number out of 80, it will be multiplied with the average IoU score to get the final score. This policy has been put in place to discourage trivial solutions that, for example, return all or nothing (see dummy solutions) and achieve extremely small build/indexing time and index size (to get high scores in the first 6 components) but at cost of poor retrieval quality.

Seek time calculation. We assume that each block in the virtual disk can store a single tuple. Suppose there are 5 tuples in the dataset namely (1,deebo,2008), (2,deeba,2013), (4,inam,1990), (9,agus,1991), (10,ahbra,2004) and suppose your `my_index()` method chooses to lay them out on disk once sorted by name then sorted by year. Then the `disk` list returned by the `my_index()` method should look like [10,2,1,4,9,4,9,10,1,2] and it would take up 10 blocks to store. Note that the virtual disk now has 10 blocks. On the query `SELECT id FROM tbl WHERE name LIKE 'a%'`, the ids 9 and 10 have to be returned. This can be done either by returning the disk locations [0,4] or else [6,7]. Both will yeild the correct answer but the seek times would be very different.

1. Let the virtual disk have n locations ($n = 10$ in the above example).

2. Upon receiving a list of disk locations, the virtual disk head will teleport itself to the first location in the list (no seek time).

3. It takes unit time to read a disk location. Moreover, in the time it takes for a block to be read, the disk head has already moved on to the next location.

4. The virtual disk reader reads disk locations from left to right and loops back to the 0th location upon reaching the end of the virtual disk. Thus, in the runnning example with 10 locations in the disk, if the head starts reading from location 6 then it would read locations in the order [6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, . . .].

Thus, if the locations [6, 7] have to be read, then the seek time would be 0 units and read time would be 2 units. However, if the locations [0, 4] have to be read, then the seek time would be 3 units and read time would be 2 units. Notice however, that if the locations [7, 6] have to be read, then the seek time would increase to 8 units (read time would still be 2 units). This is because after reading disk location 7, the virtual disk head would have to continue reading in the right direction and loop back to be able to reach location 6. **See the Google Colab evaluation file for the exact seek time calculation algorithm.**

Dummy Submission File and Dummy Tuples. The files `index.py` and `execute.py` are template files that show how your submission files must look like. In order to help you understand what the data tuples look like, what the clauses look like, and how we will evaluate your submission using the evaluation script, we have included a `dummy_index.py` and `dummy_execute.py` files in the assignment package itself (see the directory called `dummy`). We have also included an example database in CSV and Sqlite formats as well as a list of list of SQL queries in the form of clauses. **To see how to read the data tuples and clauses into memory so that you can start processing them, please look at the Google Colab evaluation script.** The `make_predicates()` and `make_sqlite_query()` methods may be of help in this regard. However, note that your submission will be evaluated on a different set of tuples and different queries. The tuples and queries will look similar to the ones we have provided with the assignment package but not be identical. Also note that the dummy index and execute files mostly implement very bad strategies which will score poorly. However, this is okay since their purpose is only to show you the code format.

Validation on Google Colab. Before making a submission, you must validate your submission on Google Colab using the script linked below.
Link: `https://colab.research.google.com/drive/1VNaAmgPzJ-fagaRZ_yJ_jLxyG29jQqcj?usp=sharing`

Validation ensures that your submitted files will work with the automatic judge and will not give errors. Please use the Google Colab notebook and the CSV files `public.csv` and `clauses.csv` and the DB file `public.db` to validate your submission. You will have to rename these files first since the validation script uses different names to refer to these files (`secret.csv`, `secret_clauses.csv`, `secret.db`). **Please note that there will be penalties for submissions which were not validated on Google Colab and which subsequently give errors with our automated judge.**

3 How to prepare your code submission

The assignment package contains skeleton files `index.py` and `execute.py` which you should fill in with the code of your proposed method. You must use these skeleton files to prepare your code submission (i.e. do not start writing code from scratch). This is because we will autograde your submitted code and so your code must have its input-output behavior in a fixed format. Be careful not to change the way the skeleton file accepts input and returns output.

1. The skeleton code has comments placed to indicate non-editable regions. **Do not remove those comments**. We know they look ugly but we need them to remain in the code to keep demarcating non-editable regions.
2. The code files you submit should be self contained and should not rely on any other files (e.g. other .py files or else pickled files etc) to work properly. Remember, your ZIP archive should contain only two Python (.py) files.
3. You are allowed to freely define new methods, new variables, new classes in inside your submission Python files while not changing the non-editable code.
4. Do not import any modules or external files in your Python files. Do not perform any prohibited action (see above for list e.g. file I/O, using dictionaries etc).
5. Before submitting your code, make sure you validate on Google Colab to confirm that there are no errors etc.
6. You do not have to submit the evaluation script to us – we already have it with us. We have given you access to the Google Colab evaluation script just to show you how we would be evaluating your code and to also allow you to validate your code.

4 How to make your code submission

1. Your ZIP file should contain two Python (i.e .py) files called index.py and execute.py. We do not care what you name your ZIP file but the Python files sitting inside the ZIP file must be named index.py and execute.py. There should be no sub-directories inside the ZIP file – just two files. We will look for two Python (.py) files called index.py and execute.py inside the ZIP file and delete everything else present inside the ZIP file.
2. **Do not submit Jupyter notebooks or files in other languages such as C, C++, R, Julia, Matlab, Java**. We will use an automated judge to evaluate your code which will not run code in other formats or other languages (submissions in other languages will get a zero score).
3. There will be a penalty for not adhering to the prescribed format and you may have to resubmit your code.
4. Password protect your ZIP file using a password with 8-10 characters. Use only alphanumeric characters (a-z A-Z 0-9) in your password. Do not use special characters, punctuation marks, whitespaces etc in your password. Specify the file name properly in the Google form.
5. Remember, your file is not under attack from hackers with access to supercomputers. This is just an added security measure so that even if someone guesses your submission URL, they cannot see your code immediately. A length 10 alphanumeric password (that does not use dictionary phrases and is generated randomly e.g. 2x4kPh02V9) provides you with more than 55 bits of security. It would take more than 1 million years to go through all $> 2^{55}$ combinations at 1K combinations per second.

6. Make sure that the ZIP file does indeed unzip when used with that password (try `unzip -P your-password file.zip` on Linux platforms). There will be a penalty for submitting ZIP files that do not unzip and you will have to resubmit your code.

7. Upload the password protected ZIP file to your IITK (CC or CSE) website (for CC, log on to `webhome.cc.iitk.ac.in`, for CSE, log on to `turing.cse.iitk.ac.in`).

8. Fill in the following Google form to tell us the exact path to the file as well as the password `https://forms.gle/7hsAyTw4Q7hWPzQdA`

9. **Do not host your ZIP submission file on file-sharing services like Dropbox or Google drive. Host it on IITK servers only**. We will autodownload your submissions and GitHub, Dropbox and Google Drive servers often send us an HTML page (instead of your submission) when we try to download your file. Thus, it is best to host your code submission file locally on IITK servers.

10. While filling in the form, you have to provide us with the password to your ZIP file in a designated area. Write just the password in that area. For example, do not write "Password: helloworld" in that area if your password is "helloworld". Instead, simply write "helloworld" (without the quotes) in that area. Remember that your password should contain only alphabets and numerals, no spaces, special or punctuation characters.

11. While filling the form, give the complete URL to the file, not just to the directory that contains that file. The URL should contain the filename as well.
    (a) Example of a proper URL:
    `https://web.cse.iitk.ac.in/users/purushot/cs315/my_submit_assn1.zip`
    `https://home.iitk.ac.in/~purushot/cs315/my_submit_assn1.zip`
    (b) Example of a proper URL:
    `https://web.cse.iitk.ac.in/users/purushot/cs315/assn1/group12.zip`
    `https://home.iitk.ac.in/~purushot/cs315/assn1/group12.zip`
    (c) Example of an improper URL (file name missing):
    `https://web.cse.iitk.ac.in/users/purushot/cs315/`
    (d) Example of an improper URL (incomplete path):
    `https://web.cse.iitk.ac.in/users/purushot/`

12. We will use an automated script to download all your files. If your URL is malformed or incomplete, or if you have hosted the file outside IITK in a manner that is difficult to download, then your group may lose marks.

13. Make sure you fill-in the Google form with your file link before the deadline. We will close the form at the deadline.

14. Make sure that your ZIP file is actually available at the link at the time of the deadline. We will run a script to automatically download these files after the deadline is over. If your file is missing, we will treat this as a blank submission.

15. There will be a penalty for submitting malformed URLs or if your submission cannot be found at the URL and you will have to resubmit your code.

16. We will entertain no submissions over email, Piazza etc. All submissions must take place before the stipulated deadline. Your ZIP file must be available at the link specified on the Google form at or before the deadline.

5 How to prepare your report submission

Use the following style file to prepare your report.
`https://media.neurips.cc/Conferences/NeurIPS2023/Styles/neurips_2023.sty`

For an example file and instructions, please refer to the following files
`https://media.neurips.cc/Conferences/NeurIPS2023/Styles/neurips_2023.tex`
`https://media.neurips.cc/Conferences/NeurIPS2023/Styles/neurips_2023.pdf`

You must use the following command in the preamble

```
\usepackage[preprint]{neurips_2023}
```

instead of `\usepackage{neurips_2023}` as the example file currently uses. Use proper LaTeX commands to neatly typeset your responses to the various parts of the problem. Use neat math expressions to typeset your derivations. Remember that all parts of the question need to be answered in the PDF file. All plots must be generated electronically - hand-drawn plots are unacceptable. All plots must have axes titles and a legend indicating what are the plotted quantities. Insert the plot into the PDF file using LaTeX `\includegraphics` commands.

6 How to make your report submission

1. The report must be submitted as a single PDF file using Gradescope in *group submission mode.*
2. Unregistered auditors cannot make submissions to this assignment.
3. Make only one submission per assignment group on Gradescope, not one submission per student. Gradescope allows you to submit in groups - please use this feature to make a group submission.
4. Link all group members in your group submission. If you miss out on a group member while submitting, Gradescope will think that person never submitted anything.
5. You may overwrite your group's submission as many times as you want before the deadline (submitting again on Gradescope simply overwrites the old submission).
6. Do not submit Microsoft Word or text files. Prepare your report in PDF using the style file we have provided (instructions on formatting given below).