Data Structures and Algorithms: Mini Project

Debojit Das, Arpan Dasgupta, Tanish Lad, Animesh Sinha, Divanshi Gupta,
Dixit Garg, Akshat Goyal, Tushar Joshi, Dhruv Sharma, and Aditya Kumar
Data Structures and Algorithms - Teaching Assistants, IIIT Hyderabad.
(Dated: April 14, 2021)

I. 8X8 CHECKERS

Develop a 2-player game of 8x8 Checkers that you can play on the terminal.
Assigned to: ***Debojit Das***

A. Legend

Checkers (also known as Draughts) is a popular group of board games. The most commonly played one is an 8x8 version of American Checkers (also known as English Draughts). Here, you will be developing a terminal version of this game.

B. Tasks

Here are the tasks that you are expected to implement:

1. A fully functional 8x8 American Checkers game. Refer to this link for the rules.

2. A feature to UNDO as many moves as asked by the players (provided both players agree)

3. A feature to REVIEW the game from the start, so the players can see what led to the current game state if they wish to

4. A feature to SHOW ALL POSSIBLE MOVES for the next k turns (if both players make a move each, that is counted as two turns), where k is a number that the player gives

Other than these, feel free to implement any feature that you feel will be useful to the players. Creativity will be rewarded.
Handle edge cases well so your code does not crash. Write clean code and add appropriate comments. Make a readme file providing instructions on how to run the game and how to play it. Make a report specifying the data structures used and the time complexity of the features.

C. Scoring and Evaluations

You will be evaluated on the correctness of your program, the structure and neatness of your code, how smooth the gameplay experience is, the UI, the readme and the report. A viva will be taken during the evaluations for which you are expected to know the specifics of the entire project to a decent extent and the specifics of the part you implemented to a very deep extent.

II. EXAMTEX

We need help with making managing our question banks to prepare different problems of uniform difficulty in exams, and you need to come to the rescue.
Assigned to: ***Animesh Sinha***

A. Legend

We use Moodle to make question banks with slightly different structures and values. We use Moodle again for updating your marks. In all of this, we have a real hard time managing and reusing our question banks. It's not really possible to ensure that there is a lot of different questions and each student gets a different set and yet equal difficulty is maintained. Finally, setting up Moodle is not an option for most schools, it would be a lot better if your problem writing language was decoupled from the website that hosts it. So here we ask you to make a markup language that helps you write questions for exams.

B. Language Specification

The user should be able to access two sets of files, and optionally feed a third file.

- Question Bank: Contains a lot of questions with their type and difficulty.

- Question Paper: Includes the question bank files and samples questions of different difficulties from it, making the final question paper output.

- Marks File (not in the scope of this project, but do plan for it): stores the results for an exam, used to update the difficulty of questions.

The Question bank file has lines in the format as shown below, you can add elements to this formal, but hold this represents the minimum functionality you are required to implement:

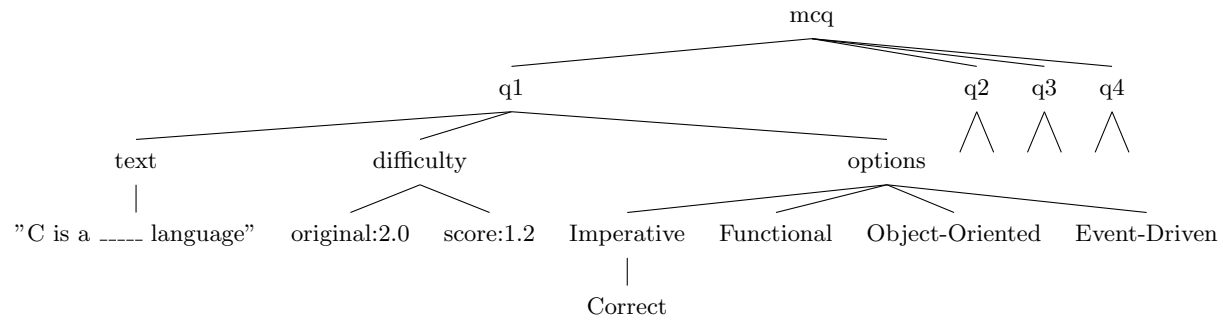

FIG. 1: Sample Abstract Syntax Tree

```
\question{type = mcq}{difficulty = 0.2}
{text = C is a _____ language}
{opt=Functional,Procedural,Objects,Events}
{ans=Functional}
```

This should get parsed into a tree structure as shown in Fig. 1.

Next we will have the Question paper language, which will take the tree built by the question bank, and use it to generate the question paper. It will have lines like the following example:

```
\sample{type = mcq}
{difficulty <= 0.2}
{number = 5}
```

C. Project Plan

1. Make a few sample input files, both Question Bank and Question Paper, to visualize what is it that you are parsing.
2. Make a list of the rules you will use to implement this transform from text to tree, what will you do when you encounter each symbol (\, {, }, =, etc.), and how will you manage your stack and your tree as you build it.
3. Use these rules to implement the building of your tree, find some basic traversal of the tree to print it and check if everything was correctly built. This is a valuable checkpoint. Try to write some automated tests here (optional).
4. Start parsing your sampler line by line and write output to another text file. This should feel like a complete project for 2-3 types of questions.
5. Add functionality like having more options in MCQs than required (i.e. there are 8 options in the question bank, but when printed each user get 1 correct + 3 wrong options), have multiple correct and other types of questions, etc. This will test if you have written good scalable code.
6. Imagine what the most general format of a marks file is, and how it can be used to get a better estimate of difficulty of questions. You **do not** need to code this, just plan it out.

D. Evaluation and other comments

You will be evaluated on your ability to implement each of these steps laid out in the project plan. Each student in the team is expected to know the details of the overall plans, the rules laid out in point 1 of the project plan, what code is present in each file, etc. The work needs to be equitably distributed, and all students need to answer questions on what they implemented. We will conduct a manual evaluation session for this, run the code and test. There will be associated vivas. The code quality is very important, use `clang-tidy` and `clang-format` to ensure readability of your code from the start. This is a big project, once the code become unmanageable, no TA will be able to help.

III. ROUTING

You have to develop risk averse routing algorithms on simulated city maps.

Assigned to: ***Tanish Lad***

A. Legend

You have an exam to get to in 1 hour. But it's marriage season in your tiny little cities, and all roads that take you to your exam center often get super congested and no traffic can get through for hours. These traffic jams keep appearing suddenly when the marriage processions get on the street, so you cannot just know where they are and plan to avoid them exactly. Google Maps, with all it's shortest path routing algorithm which will get you

to your exam center in 27 minutes. Your dad however, doubts this assessment. In the average case this is the best and the fastest option, but you are told that it's safer to go through the outskirts of the city which will surely get you to your exam center in 45 minutes.

Being a computer scientist, you don't want to keep asking your dad for advice, and want to fix this issue in Google map's shortest path routing algorithm. Take your stab at it!

B. Major Steps

You shall be expected to complete the following tasks:

1. Make a program to read in and manipulate graphs, take input from a file, etc. This should support insertion and deletion of edged, finding adjacency lists, finding the weight of an edge, etc.

2. Implement Dijkstra's algorithm to find the shortest path between two nodes on the graph. Write tests for this algorithm and try it out on some sample city maps (hypotheical maps as graphs).

3. Use simulated traffic data, and estimate the weights of edges for fastest routing. Here you need to either manufacture data from a simulation or use the Google HashCode 2021 traffic data (downloadable from https://codingcompetitions.withgoogle.com/hashcode/archive), or preferrably both.

4. Formulate a definition of congestion and safe routing. Use your new objectives to enhance your dijkstra's algorithm. This formulation is left to your mathematical creativity and understanding of probabilistic modelling of roads. It can be as simple or complex as you want.

The following are tasks that you are encouraged to complete, but they will **not** form a significant portion of your evaluation:

1. Try to visualize the graphs you plot and the traffic on them using some graphing library. (C may not be the language of choice for this)

2. See if you can get real world data from Google maps to test out if your heuristics work and research on better heuristics.

IV. TREE SEARCH LIBRARY

Tree searches and their visualizations are key to the widest variety of modern algorithms. Make a general tree search library to contribute to this task.

Assigned to: ***Arpan Dasgupta***

A. Legend

You need to write a generic tree search library with the ability to simply change the exploration strategy (and hence the algorithm) by changing a single function in the code. The library should also provide several statistics about the search algorithm which allows easy plotting and analysis of the strategy. These kinds of libraries (much advanced versions ofc) are used in places like DeepMind to analyse their search strategies like Monte Carlo Tree Search.

B. Project Requirements (Subparts)

1. The tree node should be generic enough to allow arbitrarily complex information to be stored (say game states and other variables). By default, you might want to store information like time seen, depth etc. to perform different searches.

2. The next node to explore in a tree search can be picked via a generic priority queue instead of a specific data structure like stack or queue in DFS or BFS. The comparator function for the priority queue should be something which can be plugged in by the user to allow them to change the algorithm.

3. Default algorithms - **DFS, BFS, A-star and greedy search** need to be provided as sample (basically, write the comparator function with required node structure for these algorithms). [Big bonus if you can do MCTS :P]

4. Some information needs to be placed in a global array at each iteration for analytics. Some of this information will be - max depth, average depth, branching factor. You can add more of this if you want. This information can be plotted/analyzed for different algorithms to show the difference. (DFS max depth increases fast, branching factor less, etc.) [Plotting not necessary but would be nice]

C. Additional Information

1. The file the user needs to change must be a single file which has the node struct, the comparator for the priority queue and any other necessary function which the user needs to modify.

2. The tree structure will be input beforehand from the user (including required details for each node).

3. The data to test your algorithm on can be picked from somewhere or can be generated or we might provide it to you, more info on this later.

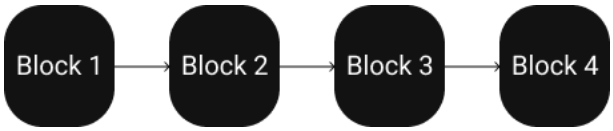

FIG. 2: Block-Chain

Block Number:	5
Transactions:	[{from, to, amount}, ...]
Previous Block Hash:	"xhgjhg36547621"
Nonce:	23

TABLE I: Block Structure

D. Evaluation and other comments

You will be evaluated on your ability to implement each of these steps laid out in the project plan. Each student in the team is expected to know the details of the overall plans, the rules laid out in point 1 of the project plan, what code is present in each file, etc. The work needs to be equitably distributed, and all students need to answer questions on what they implemented. We will conduct a manual evaluation session for this, run the code and test. There will be associated vivas. The code quality is very important, use `clang-tidy` and `clang-format` to ensure readability of your code from the start. This is a big project, once the code becomes unmanageable, no TA will be able to help.

V. BITCOIN

In this project, you will simulate the working of Bitcoin and do various operations.

Assigned to: ***Aditya Kumar***

A. Description

In the Bitcoin system, transactions are stored in data blocks like this. Each Block contains 50 transactions, and it is structured as follows.

Assume *PreviousBlockHash* of the first block is 0. For calculating the *PreviousBlockHash* , you can design your own Hash Function, which accepts *BlockNumber* , Transactions, PreviousBlockHash and Nonce of the Previous block.

You also need to store the user details; you can use any data structure to do so. It must store details like Unique ID, Wallet Balance, Transaction History, *JoinDateTime*.

B. Task List

You have to implement the following functionalities

1. **Add user to the system** - Assigns a randomly generated unique ID

2. **Transact** This takes three input Sender UID, Receiver UID, and Amount to be transferred. Decline the transaction if the sender's wallet balance is lower than the Amount entered or invalid UIDs. Also Update the user's transaction history

3. **Create Block** Once you have 50 Transaction, create a block and add it to the Chain of blocks. Value of Nonce can be any random integer between 1 to 500

4. **Attack** Randomly generate a number between 1-50, if a Block with *BlockNumber == Randomly Generated number* exits then modify its Nonce. Otherwise, the attack fails

5. **Validate Block-Chain** Check if Block-chain is valid, i.e. PreviousBlockHash is correct; if invalid, it means that the previous Block has been attacked, adjust its Nonce to match the *PreviousBlockHash* of the next Block

Note - Bitcoin is distributed blockchain network, and It needs a consensus protocol to add new blocks or check the validity of a node. This Project aims to give you a smaller picture of Bitcoin and in no way a complete representation of Bitcoin.

VI. COVID TRACING

You have to develop an application that traces covid people.

Assigned to: ***Akshat Goyal***

A. Problem Details

1. There are N stations, some of the stations are directly connected by the bidirectional roads. There are a total of K people staying in different stations.

2. The people who meet the covid positive people are declared as the primary contacts and those who meet the primary contacts are secondary contacts.

3. As the stations are small, it is assumed that all the people at the station will come in contact with each other and hence if one covid positive person P comes to the station, all the people in that station will be declared as the primary contacts of the P and so on.

4. The danger value of the Station S is defined as: (number of covid positive people) + (number of primary contacts / 5) + (number of secondary contacts / 10). The danger value of the path is the sum of the danger value of the stations in the path. The safety of the path is inversely proportional to the danger value of the path.

B. Major Tasks

You are expected to complete the following tasks:

1. Given the list L of covid positive people on day D, you need to print the list of primary and secondary contacts of the L in the last X ($1 \leq X \leq 15$) days. All the people in the L will be quarantined for 14 days at their current stations.
2. A person P is currently at station S and wants to go to station $S2$ on day D. You need to find the top 3 (if exists) safest and shortest path from S to $S2$. After that, P may choose one of the path to $S2$ and will go through that path, stopping at each intermediate station and reach $S2$ on the same day or may decide to not go. Note that safest and shortest path means that you need to rank paths based on their safety value first and then for the paths with the same safety value you need to rank the shorter path higher.
3. You need to handle other queries like status (negative, positive, primary contact, secondary contact) of a person, location of a person, list of covid positive, primary contacts, secondary contacts at station S.

C. Input

1. In the first line you will be given three integers N, M, K, denoting number of stations, number of roads, number of people respectively.
2. Then M lines follow, the i^{th} of them contains three integers U, V, W, meaning there is a bidirectional road between U^{th} station and V^{th} station of length W.
3. The next line contains K space separated integers where i^{th} integer represents the initial station number of the i^{th} person.

D. Instructions

1. You are allowed to change the above input format and you need to define the input format for the queries so that the user can ask multiple queries. Mention the input format in the Report or Readme.
2. You are allowed to make suitable assumptions wherever you feel. Mention the assumptions in the Report or Readme.
3. Write neat and clean code and add comments.
4. You will be evaluated based on the correctness of your code and, time complexity of each type of queries and memory complexity of the code.

VII. CURRENCY

You have to develop a currency exchange monitor.
Assigned to: ***Dhruv Sharma***

A. Legend

After graduating from your college, your friends decided to settle in different countries. In a few years, you planned to launch a start up together. Obviously, it required transactions of money, and repeated conversions of different currencies. For converting a currency, you'll go to some Trade Bank, which will apply a certain conversion rate.

Since your startup requires frequent conversion of currencies from all over the world, you decided to develop an algorithm that provides you the best way of exchanging currencies.

B. Functionalities

You have to implement the following functionalities in your program.

1. Add or delete a Trade bank. (String)
2. Add or delete a currency. (String)
3. Add or delete a currency conversion from a specified Trade Bank. (Conversion rate - Integer)
4. The best path of converting from currency A to B along with it's cost and the chosen Trade Bank. You can only choose a single Trade Bank for all the exchanges regarding this query.

C. Major Steps

You are expected to complete the following tasks:

1. Make an input program which can generate inputs for the specified functionalities in an input file. Your main program should be able to take the input from the generated input file. The program should not terminate unless specified.

2. Implement Addition/Deletion of banks, currencies and currency exchanges. Report if an operation performed is invalid.

3. Implement an algorithm to find the best way of currency conversion. The answer should be consistent with the changes in the data.

4. Write tests for your algorithm, and try it out on sample test cases.

5. Make a simple README.md file specifying the Input and Output format for all the functions, the complexity of you algorithm and specifying it's limitations.

The following are tasks that you are encouraged to complete, but they will **not** form a significant portion of your evaluation:

1. Clearly specified inputs and outputs.

2. Find the second best currency conversion, from A to B. (The total currency conversion price should be strictly greater than the best conversion price.)

3. Find any cycle, if it exists, for a specified Trade bank.

VIII. FRIENDS RECOMMENDATION SYSTEM

Develop a friends recommendation system like Facebook.

Assigned to: ***Dixit Kumar Garg***

A. Legend

You are fascinated about how Facebook manages to recommend friends which are in most of the cases related to us in some ways. So you decided to come up with your own system to recommend friends. Of-course, it won't be some complex ML stuff!!

B. Specifications

1. User

1. Users can register in your system. Each user will have some parameters like name,age,city,etc. Also there must be some unique ID for every user. The ID should be a minimum positive integer which is not currently in use.

2. Users can unregister from your system. After a user unregisters from the system, its ID can be used by another user.

3. Each user has a collection of users which are called its friends. Note that it is possible that user 'A' is a friend of user 'B' but not vice versa. If a friend unregisters from the system, then that friend must be removed from his friends list.

2. Friends Recommendation

1. **For already registered users**, show top 'K' recommendations. Recommendation should be done as follows

2. A friend should not be recommended. '

3. All second friends (friends of friends) should be given more preference than all third friends. All third friends should be given more preference than all fourth friends and so on.

4. For two friends which are both i^{th} friend, they will be preferred in random order.

5. **When a new user is registering in the system**, show top 10 recommendations based on some common parameters. The more parameters a user has in common, the more preferred it will be. If two users have the same number of common parameters, then choose randomly.

6. If the number of already registered users are less than or equal to 10, then show all the users.

7. If the number of already registered users are greater than but the number of users with some common parameters are less than 10, then show all those users with common parameters and then show other users randomly such that they are all 10 in total.

8. Let, the number of recommended users be K, now the new registering user would add any X ($\leq k$) users to friends. 'K' along with the list of ID of friends would be given as an input.

3. Check Friendship status

1. You will be given an ID of two users A and B and you have to tell whether A is a friend of B or not.

2. You have to do this operation **efficiently**, the more efficient this operation will be, the more marks would be given for this part.

C. Evaluation and other comments

1. You will be evaluated on how well you have implemented the features and with what complexity (both space and time would be considered).

2. You can come up with your own user interface. For user Interface we are not expecting anything beyond command line, but you are welcome to create other exciting interfaces.

IX. ASSIGNMENT SHELL

Making a command line app which helps you manage and maintain the file structure on your assignment files, as well as readily accept updates to the problem structure.

Assigned to: ***Tushar Joshi***

A. Legend

After the assignment is released, TAs make changes to the problems, update the PDFs and the distribution packages, their file system checker code does not work and we have to send them screenshots for debugging, it's all so annoying. Let's go fix this process.

B. Commands Spec

Your shell should work like normal bash, show a prompt like: `animesh/dsa>` , where you can type your commands. The following commands need to be supported. The abbreviated name of the subject you are currently working on should be shown in the shell. This should persist across runs.

1. `switch <subject>`: Change the name of the subject in the prompt and change folders appropriately.

2. `create <assignment>`: Creates a new folder for the assignment, downloads (or copies locally) the contents of the dist folder and the problem statement into the current directory.

3. `update <assignment>`: Downloads (or copies locally) the new assignment files, deletes the old ones, and replaces them.

4. `setup <assignment>` (optional): Reads an indented text file which has the folder structure to be followed. Creates that folder structure.

5. `test <assignment>`: Runs the submitter.py file in the dist folder, if it exists. Store the logs, be it compilation error or others in a file that can be sent to the TA for debugging.

6. `submit <assignment>` Makes a zip file of the current directory. Uploads the file to our servers (or copies to local directory)

7. `compare <assignment> <zipfile>` Compares the file tree to check if any of the files in the submitted zip are different from any of the files in the current directory. Prints list of those files. Uses MD5 hash (command MD5 sum) to compare.

8. `use <assignment>` Changes the prompt to `animesh/dsa/<assignment>` and for all commands now, if the `<assignment>` argument is not passed, it will default to this assignment.

C. Tasks List

1. Make a basic interactive shell (Shameless plug of pretty code for interactive shell: https://github.com/AnimeshSinha1309/os-assignments/tree/master/AniShell - only see small parts of processor/prompt.c, processor/parser.c, `processor/tokenizer.c`). You don't need to implement any of the features here, just taking input, splitting by spaces, and making cases for each command.

2. Make the commands as listed in the spec above. Try to write basic tests to see if all the commands are working (optional). Each command will carry some marks, and this is the bulk of the project, so build incrementally.

3. Propose what commands need to be added to make the download and running of these in evals (the way we did for assignment 1) as well as running MOSS becomes very easy and fast. Suggest how you will implement these features. Can you ensure that there are almost no cases where the students say "It was working, but now it's not?", if we give many of the test cases. You don't need to implement anything here.

D. Comments on Implementation

You might need to implement a stack to store and parse your inputs. You may also need to maintain a tree to represent the current directory structure, find changes, updates files, etc. Try to take your best call on these. This project is implementation heavy. We would appreciate well handled error cases, and nice pretty code.

X. DIRECTORY MANAGER

You need to develop a directory manager that allows to maintain and search directories efficiently.

Assigned to: ***Divanshi Gupta***

A. Tasks

Make a directory manager program that implements the functionalities specified below. The current directory when the program is initialised should be "root" and it should be empty. The program should not terminate unless specified. The given functionalities are to be implemented -

- **ADD**: Implement addition of file/directory inside current directory. Take user input for type(file/directory) and name.
- **MOVE**: Change current directory to any directory by taking input the complete path to that directory. Error handling should be done for incorrect paths.
- **ALIAS**: Implement saving directory with an alias. Take input the complete path to that directory and the alias. Error handling should be done for incorrect path and for already-exiting aliases.
- **TELEPORT**: Change the current directory to any directory by taking input the alias of that directory. Error handling should be done for non-existing aliases.
- **FIND**: Implement searching all the files/directories with given prefix inside current directory. Take a prefix-string input for search. Output all the files/directories with given prefix.
- **QUIT**: Terminates the program.

NOTE: All the functionalities specified above are to be implemented inside the program. Ex- For adding file to a current directory you just have to add the corresponding file name to whatever data structure you are storing your files in (You don't have to actually create a new file inside your current directory. This is not a shell assignment :P). Similarly ,for moving and teleportation, you don't have to actually change your shell directory but only the current directory variable in your program should be changed and shown as output.

B. Evaluation

You will be evaluated based on the correctness of functionalities and usage of the most efficient data structures. Error handling should be done and will be tested. A viva will be taken during the evaluations for which you are expected to know the specifics of the entire project to a decent extent and the specifics of the part you implemented to a very deep extent.