OPEN COMMUNITY

Student Experience Advisory: Issue 1

Minor Project Ideas

Issue Date: 29 August, 2020
Intended for: CSE - III Year

IDEA	Description
Compression using Huffman Coding	The Huffman encoding algorithm is an optimal compression algorithm when only the frequency of individual letters are used to compress the data. (There are better algorithms that can use more structure of the file than just letter frequencies.)
BSTs following the memoization algorithm	In computing, memoization or memoisation is an optimization technique used primarily to speed up computer programs by storing the results of expensive function calls and returning the cached result when the same inputs occur again. Memoization related to dynamic programming. In reduction-memoizing BSTs, each node can memoize a function of its subtrees. Consider the example of a BST of persons ordered by their ages. Now, let the child nodes store the maximum income of each individual. With this structure, you can answer queries like, "What is the maximum income of people aged between 18.3 and 25.3?" It can also handle updates in logarithmic time.
Obscure binary search trees	Items, such as names, numbers, etc. can be stored in memory in a sorted order called binary search trees or BSTs. And some of these data structures can automatically balance their height when arbitrary items are inserted or deleted. Therefore, they are known as self-balancing BSTs. Further, there can be different implementations of this type, like the BTrees, AVL trees, and red-black trees. But there are many other lesser-known executions that you can learn about. Some examples include AA trees, 2-3 trees, splay trees, scapegoat trees, and treaps. You can base your project on these alternatives and explore how they can outperform other widely-used BSTs in different scenarios. For instance, splay trees can prove faster than red-black trees under the conditions of serious temporal locality.
Optimal treaps with priority-changing parameters	Treaps are a combination of BSTs and heaps. These randomized data structures involve assigning specific priorities to the nodes. You can go for a project that optimizes a set of parameters under different settings. For instance, you can set higher preferences for nodes that are accessed more frequently than others. Here, each access will set off a two-fold process: ● Choosing a random number ● Replacing the node's priority with that number if it is found to be higher than the previous priority
Search engine for data structures	The software aims to automate and speed up the choice of data structures for a given API. This project not only demonstrates novel ways of representing different data structures but also optimizes a set of functions to equip inference on them. We have compiled its summary below. The data structure search engine project requires knowledge about data structures and the relationships between different methods. It computes the time taken by each possible composite data structure for all the methods. Finally, it selects the best data structures for a particular case.
Stack-based text editor	Your regular text editor has the functionality of editing and storing text while it is being written or edited. So, there are multiple changes in the cursor position. To achieve high efficiency, we require a fast data structure for insertion and modification. And the ordinary character arrays take time for

	storing strings.
	You can experiment with other data structures like gap buffers and ropes to solve these issues. Your end objective will be to attain faster concatenation than the usual strings by occupying smaller contiguous memory space.
Social Network Simulation using Graph	Social network analysis (SNA) is the process of investigating social structures through the use of networks and graph theory. It characterizes networked structures in terms of nodes (individual actors, people, or things within the network) and the ties, edges, or links (relationships or interactions) that connect them.
Metaheuristics Algorithms for networking algorithms	Metaheuristics are strategies that guide the search process. The goal is to efficiently explore the search space in order to find near–optimal solutions. Techniques which constitute metaheuristic algorithms range from simple local search procedures to complex learning processes.
Graph Based Algorithm Implementation in C	1. Dijkstra's algorithm 2. Prim's algorithm 3. Topological sort 4. Kruskal algorithm
Chess Using Dynamic Programming	In computer chess, dynamic programming is applied in depth-first search with memoization aka using a transposition table and/or other hash tables while traversing a tree of overlapping subproblems aka child positions after making a move by one side. A Chess game will always be a project that every developer should lay their hands (at least every developer who wants to be a software engineer)! A Chess game has lots of moving parts and lots and lots of corner cases that one must consider! It involves and a lot of Graph Theory and even to this date making an unbeatable AI for Chess is no child's play! Even with the aid of the minimax algorithm, it can be very difficult to make an unbeatable AI for Chess (since the number of combinations is very high) and one must apply some performance optimization techniques to reduce the memory overhead!
Visualization of algorithms and data structures using JavaScript	You can make a web interface for visualization of different algorithms on large data and a dynamic view of their running time. Running time will be measured and displayed simultaneously while the program is running. This will highlight the advantages and drawbacks of various algorithms in various operations.
A Text Editor	With their relevance in every programming, sub-culture Text Editors tops the list since text-editor is the basic tool every programmer uses and he must be familiar with how a text-editor works. Companies like Flipkart, Microsoft, etc. constantly asks questions related to text-editors in their interviews-questions related to the two-stack model, undo-redo operation (with unlimited stack size unlike Notepad which allows only single undo/redo operations), finding and replacing text, using the trie data-structure (for scaling the text-editor to a word-processor), pattern-based searching (using regular expressions), stylizing text (as in code-editors) and so on. So a Text-Editor project is worth a shot.
A Graphical Calculator	A calculator is a simple application but a graphical calculator is interesting mainly because of the compact user-interface (not displaying all functionality at once). There are a lot of things that can be expanded to it such as allowing complex expressions in the text-field (as input – which may involve Expression trees), allowing user-defined functions (similar to GNOME Calculator) and so on! The complexity of the project shouldn't be under-estimated and one must try to keep the user-interface as compact as possible. All and all even if interviewers do-not ask that many questions related to expression trees yet making a graphical calculator can be a good GUI exercise and a project every developer should lay their hands on!
Terminal Shell	Shells involve a lot of string-processing and smart use of the two-stack model for storing the history of commands and to locomote between them! Making a Terminal Shell is a great exercise project for learning systems programming since it involves turning on/off a great number of bit flags to have to enter the raw-mode. A lot of GUI applications nowadays have a graphical console in them which works the same way as a shell-commands are entered, parsed and then processed! Ancient games had a console which was often a time the only means of interaction (text-based adventure games which had their window)! At one point terminal shells can also include some kind of meta-programming (talking with other programs dynamically)!

Note: OPEN does not claim ownership of ideas. To better facilitate students the ideas list has been compiled from various sources on the internet.