Final Project Sheet

Project Goals

- Summarize all of your course knowledge in one significant coding project.
- Implement research problems on Graph Scheduling algorithms.
- Create and evaluate your own algorithm(s).
- Solve a real-life problem or develop a real-life application.
- Work individually or as a team to implement an interesting system.
- Research a new algorithmic problem that we won't cover during this semester.

Project Details

A significant portion of your final course grade (20%) is the project component. There are two tracks designed for the course project. In track 01, you get an opportunity of implementing algorithms using advanced data structures such as Trees and Graphs. This option is particularly useful to master the course topics discussed in 202 and to experience the real implementation of Algorithms. In track 02, you are required to develop a course project that is to design a real-life application system that uses one or more algorithms. In track 02, you have a significant amount of flexibility with what you can do with your project, so use it wisely! Is there something you wanted to create but never had time? Is there a project from other courses that you always wanted to extend? Juniors: is there an idea you have for your thesis that you want to try out develop a pilot before committing to it? Seniors: did you want to explore a different aspect of your senior thesis? If the answer is "yes" to any of these, and if you can connect your idea to algorithms, then you can do it as your project now! Of course, you cannot claim anything you have previously done as a contribution to this project, but you can certainly use your previous work, knowledge, and experience as a backbone for this project. The final project may be developed using any programming language of your choice.

Project Track 1: Implementation of Algorithms and Data Structures

Explore a topic connected to data structure such as Trees or Graphs. Implement an algorithm that we had covered in the class or that we had not covered but listed in the textbook. An example of JGraph, Graphviz (python and java) sample code is provided in the project repository. This sample code is a starting point and may be used to visualize a Tree or a Graph data structure. To give an example, a few options in this track are listed below:

1. Implement the Heap Sort algorithm using binary heap. In this option, you should implement Heap Sort (both ascending and descending order) and conduct experiments to compare Heap Sort algorithm with other sorting algorithms such as [Insertion, Selection, Quick, and Merge sort].
2. Implement the binary trees using JGraph/Graphviz and Linked List. In this option, you should implement any meaningful algorithm (such as Binary Search Tree) that is connected to the concept of binary trees. In this option, you are expected to implement insert, delete, and search operations.
3. Implement the 2-3 balanced or Red-black balanced BST using JGraph/Graphviz. In this option, you are expected to implement insert, delete, and search operations.
4. Implement graph-based algorithms such as Shortest Path, Minimum spanning tree, and so on using the graph data structure and JGraph. These are just a few examples. Any graph-based algorithm will be acceptable in this category. These are just a few examples of Track 01.

Project Track 2: Implementation of a real-life system.

Explore a topic that is aimed to solve a real-life problem or develop a real-life application. After identifying a topic, start to research the problem selected to get an idea of what has already been done in this area. Include references to existing work in the final report. Here are some examples of projects that were done last year:

1. Develop an application for Trip Recommendation based on user provided inputs such a dollar, start, and end destination.
2. Improving room draw by analyzing numbers generated in previous years.
3. Improving room draw through a points system based on grades and extracurricular activities. – An automated, intelligent Bitcoin buyer/seller.
4. Efficient collision detection using quadtrees. There are many references to all of these problems, and I'm sure as you think of your own project, you will find resources for them as well. If you're completely stumped in coming up with a project idea, you can certainly talk to me and we will set up a brainstorming session. Be creative and choose something that is interesting to you!

General Expectations:

The course project must involve designing new algorithm(s). This requirement may be approached using the following:

1. Creating a brand new algorithm and applying it to some problem. You must do some research to make sure your algorithm has not been proposed before.
2. Designing a significant extension to an algorithm and applying it to some problem.
3. Most likely, you will need to develop or extend a series of several algorithms.
4. In any of these cases, you must provide a discussion (or proof) on the correctness of your algorithm. This discussion can include small examples and references to existing work (especially if you are extending existing algorithms.
5. Some hard requirements are outlined below:
   - You must provide a running time analysis of the algorithm.
   - Your project must have a significant implementation part where you will develop program(s) for your algorithm(s) for your chosen problem. You may write your code from scratch, or reuse and extend some existing code. Obviously, anything you use that is not yours must be documented. You may program in any programming language that you like.
   - Your project must be extensive enough to qualify as a project (think of work for at least 3 one-week lab assignments), but not too extensive so that you cannot finish it in the remainder of the semester.

Timeline and Deliverables

1. Team - Fill the team list document by end of Thursday (04/22) lab time and write out one paragraph with a title and project idea using a file named `project-idea.md`.
2. Proposal – 1-2 pages, Deadline April 26th (5:00 PM EST). Develop the ideas into a proposal document using a file named `proposal.pdf`. Write a 1-2 page technical description of what you propose to do for your project and submit a PDF copy of your proposal through the GitHub link shared. Your proposal does not need to be very detailed at this point. It should describe what project you are going to pursue, what you want to do (the real problem you will tackle, and at least a couple of references to indicate that you have done some research about the problem.

3. Progress Report – 3-4 pages, Deadline May 6th. By this point, you should have made a tremendous amount of progress towards implementing your project and submit the report using a file named `progress-report.pdf`. Were there any unexpected challenges? Did you have to change your initial model/framework or the project skeleton code? You should have also finished your analysis of your proposed methodology by now. Include everything you have done so far in your progress report, even if it is incomplete. No need to include the actual code (unless you want my help with it), just describe what progress you have made with it. Submit a PDF copy of your progress report using the GitHub link shared.

4. Presentation – The deadline for this part is May 12th 5:00 PM EST. All team should record a 10 minutes video (only screen recording) to virtually present their course project. I expect every team member to participate in the presentation. Please present individually, if your project is done individually by you. It is expected that students use Slides during their presentation. The link to upload will be shared at a later point. By the presentation session, you should have finished implementation, run some preliminary experiments, and done some basic analysis. In the presentation, you should describe the motivation, problem definition, challenges, approaches, and results and analysis. Use diagrams and a few bullet points rather than long sentences and equations. The goal of the presentation is to communicate the important high-level ideas and give intuition rather than be a formal specification of everything you did. Design at least 6 to 10 slides, including a slide with the title of your project and your name. Also, you need to show a short demo of your tool at the end of the presentation.

5. Final Report – 6-8 pages, Deadline May 17th EOD. Incorporate any feedback from the progress reports and the presentation session and submit the final report using a file named `final-report.pdf`. Your final report should be clear and well written, which includes no typos or grammatical errors. Your report should be written in a professional and technical manner. Your report should include the following:

    - The motivation for your project. Why is the problem you decided to solve important or useful?
    - Background for the proposed problem. What have others done for it already? Include references.
    - Detailed data structure overview, description of the proposed implementation, and the complexity analysis. Include pseudocode, diagrams, and examples if appropriate. If you are extending existing work, briefly describe previous work and include references to it.
    - Description of your results. Make graphs, tables, and anything else that can help me understand your results.
    - Conclusion. Give a short overview of your project and its results. Describe what you learned, what were the biggest challenges and the biggest rewards.

Grading Rubric

For each deliverable, you need to submit a PDF with your report (or presentation slides). For your final report, you need to submit any supplementary material (code, data, a README file documenting what everything is, and how to run your program) to the git repository using the link shared:

1. Proposal – 15 points
2. Progress report – 15 points
3. Presentation – 35 points
4. Final report and implementation – 35 points
5. Please make sure to sign the honor code statement in all submission folder.