CSCI208 Projects Instructions
Spring 2025

- Within the project, you need to **implement** your **algorithms,** rather than using already built-in functions that do everything. Building a project with **only built-in library functions is not allowed**.
- The team must **understand** the **algorithm** dynamics and be able to explain what is happening and why.
- Number of members per team from 3-5 (4 is preferred)
- Team members must be students of the same doctor.
- Each member should clarify his work/role within the project.
- Repeated projects from other subjects such as AI will not be accepted and will get zero grade beside an academic dishonesty case.
- Project Submission: only via Moodle.
- Submission Deadline: details will be announced on Moodle.

  1. Each team should only submit the project deliverables **once by the team leader**. Other team members should upload one document file with the team leader and project name.
- All team members must be there during the discussion according to the announced schedule.
- For students who do not attend the discussion according to the announced schedule:

  1. For those who have an excuse approved by the university, they will be allowed to discuss later without any penalty.

  2. For those who show an excuse after the scheduled discussion time and before final exam date, they will be allowed to discuss later with a penalty of 20%.

  3. Only one date will be allowed for cases 1 and 2, if not committed a zero mark for the project will be recorded.

Deliverables: The following should be uploaded on the course Moodle site before the discussion. Else, you will get zero for all elements related to documentation.

- Executable program + The code
- Slides [+ Report if slides are not enough to recognize the work]
  - The team's work typically should include illustration of the implemented algorithms and their analysis, both theoretical and empirical if possible.

Ideas:

1. Approved idea of your own. The first choice is to work on an idea that interests you, after **getting the approval** from the course **instructor or the TA**. This is to make sure it is related to the topics discussed. This idea could be either:
  - **Your own idea,** or
  - Using the available problems on **online judges** like **CodeForces**, etc. Perfect problems are the medium or hard ones that have multiple approaches to tackle. **Easy problems** are not accepted. Then, each member will need to implement an approach, and your project will need to compare these solutions.
    - In addition to the **problem level**, your problem **must** have *more than one approach*, so each of the team members would solve it with a different approach. Problems that are hard because they ask about a specific single algorithm are not applicable in this case.

- For example, one should solve the problem with DP, the other could try applying different approach like greedy, graph algorithms, etc.
- If the number of members is the maximum, some members might not be able to find approach to use it to solve the same problem; in this case, they could work on either **another suitable problem** or the **correctness proofs** for the implemented algorithms to show they are correct.

2. **Sorting Algorithm Visualizer:** Create a web application that visualizes different sorting algorithms (e.g. bubble sort, selection sort, insertion sort, merge sort, quicksort). The project can be divided into frontend development, backend development, and algorithm implementation. A team of 3-4 members would be ideal.
3. **Graph Algorithms:** Implement some graph algorithms (e.g. Breadth-First Search, Depth-First Search, Dijkstra's algorithm, Bellman-Ford algorithm) and visualize them on a graph. The project can be divided into frontend development, backend development, and algorithm implementation. A team of 3-4 members would be ideal.
4. **Maze Generator and Solver:** Develop a program that generates a maze and solves it using different algorithms (e.g. depth-first search, breadth-first search, A* search). The project can be divided into maze generation, maze solving, and frontend development. A team of 3-4 members would be ideal.
5. **Traveling Salesman Problem:** Develop a program that solves the Traveling Salesman Problem using different algorithms (e.g. brute force, nearest neighbor, genetic algorithm). The project can be divided into algorithm implementation, frontend development, and performance evaluation. A team of 2-3 members would be ideal.
6. **Computational Geometry:** Implement some computational geometry algorithms (e.g. Convex Hull, Line Intersection, Voronoi Diagrams) and visualize them on a graph. The project can be divided into frontend development, backend development, and algorithm implementation. A team of 3-4 members would be ideal.
7. **Cryptography:** Implement some encryption algorithms (e.g. RSA, AES, DES) and compare their performance and security. The project can be divided into algorithm implementation, frontend development, and security analysis. A team of 2-3 members would be ideal.
8. **Knapsack Problem:** Develop a program that solves the Knapsack Problem using different algorithms (e.g. brute force, greedy algorithm, dynamic programming). The project can be divided into algorithm implementation, frontend development, and performance evaluation. A team of 2-3 members would be ideal.
9. **Online Course Scheduler:** Develop a program that helps students create their course schedules based on course availability and their preferences. The project can be divided into frontend development, backend development, and algorithm implementation. A team of 3-4 members would be ideal.
10. **Text Search Engine:** Create a program that searches for keywords in a given text document using different search algorithms (e.g. Brute force, Boyer-Moore, Knuth-Morris-Pratt). The project can be divided into algorithm implementation, frontend development, and performance evaluation. A team of 2-3 members would be ideal.
11. **Image Processing:** Develop a program that applies image processing algorithms (e.g. edge detection, smoothing, image enhancement) to an input image. The project can be divided into algorithm implementation, frontend development, and performance evaluation. A team of 2-3 members would be ideal.
12. **Network Flow Optimization:** Create a program that solves network flow optimization problems (e.g. maximum flow, minimum cut) using different algorithms (e.g. Ford-Fulkerson, Edmonds- Karp). The project can be divided into algorithm implementation, frontend development, and performance evaluation. A team of 3-4 members would be ideal.
13. **Longest Common Subsequence:** Develop a program that finds the longest common subsequence between two input strings using different algorithms (e.g. Brute force, dynamic programming, recursive). The project can be divided into algorithm implementation, frontend development, and performance evaluation. A team of 2-3 members would be ideal.
14. **Parallel Computing:** Implement some parallel algorithms (e.g. parallel sorting, parallel matrix multiplication) using different parallel programming models (e.g. OpenMP, MPI). The project can be divided into algorithm implementation, frontend development, and performance evaluation. A team of 3-4 members would be ideal.

CSCI208 Team
Spring 2025