King Saud University
College of Computer and Information Sciences
Department of Computer Science
Semester 471 - Fall 2025

Design and Analysis of Algorithms (CSC 311) – Fall 2025

Instructors: Prof. Mohamed Menai, Prof. Abdelmoniem Artoli, Mr. Mishal Al-Dekhayel

Course Project

Submission due date: Sunday, November 30, 2025

Oral Presentations: Monday, December 1st – Thursday, December 4th, 2025

1. Guidelines

A team of four students (at max) select and work on one of the given two projects.

2. What to Submit

1. A hard copy of a report with a cover sheet with your name(s) and a signed pledge indicating that the project write-up and the coding are your own.
2. Write-up of the project (a brief description of your implementation of each algorithm, i.e. what kind of ADT used; cost analysis; comparison of the time complexity of the algorithms, comparison of the running times of the programs, sample runs, and a conclusion).
3. For a sample run, execute your programs on the points given in the file datapoints1.txt for the first project and datapoints2.txt for the second project.
4. Include the list of points with their coordinates in your report, and a plot showing the convex hull found by each algorithm (first project) and the selected route for the second project.

5. CD/ flash media with the source files of the programs (in java/C++/Python), the executables, and a PDF of your report.

3 Important notes

- Oral presentations and demos will be scheduled after November 30, 2025.
- You may consult books and/or the web for ideas only, and it must be duly acknowledged.
- The project write-up and the coding must be your own. In case of plagiarism, the entire group of students will get ZERO.

4. Grading

(a) (3 points) Report.
(b) (3 points) Implementation.
(c) (2 points) Oral presentation and demo.
(d) (2 points) Answers to questions.

5. Projects

Project I

Project Name: Convex Hull

1) Introduction

The convex hull (a.k.a. convex envelope) of a set P of points in the Euclidean space is the smallest convex set that includes P. Figure 1 shows an example of the convex hull of a set of points.

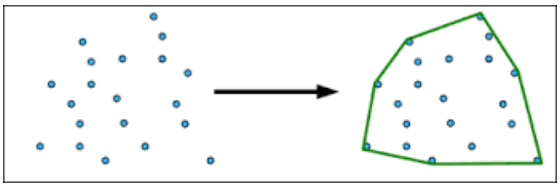

Figure 1: Example of a convex hull

Your task is to design, implement and compare the following three algorithms to determine the convex hull of n points (xi, yi) in the Euclidian plane:

1. Brute force algorithm.

2. Divide-and-Conquer (Quickhull already studied in class).

3. Graham Scan algorithm (see Chap. 33, Cormen et al.'s book, 3rd Edition).

The input points are not supposed to be sorted by their x-coordinate values. You have to work out the details. The output of each algorithm must be graphically displayed. In addition to displaying the input points, each algorithm must display the intermediary details of computing the convex hull at each iteration or recursion level.

In other words, your algorithms must display graphically their progression in computing the convex hull. For example, at each recursion level, the Quickhull algorithm must show:

- The extreme points P1 and P2.

- The details of computing the upper hull (displaying the intermediate points).

- The details of computing the lower hull (similarly).

++++++++++++++++++++++++

Project II

Project Title: Efficient Route Planning for Self-Driving Cars using Brute Force and Divide and Conquer Algorithms

Objective:

The goal of this project is to design and implement a route planning system for self-driving cars that leverages brute force and divide-and-conquer algorithms to optimize routes and reduce computational complexity.

Problem Statement:

Given a graph representing a road network, find the shortest path between two points (source and destination) while considering factors like traffic, road conditions, and vehicle constraints.

Brute Force Approach:

- Use a brute force algorithm to find all possible routes between the source and destination.
- Calculate the cost (distance, time, or a combination of both) for each route.
- Select the route with the minimum cost.

Divide and Conquer Approach:

- Divide the graph into smaller subgraphs using techniques like graph partitioning or clustering.
- Apply a shortest path algorithm (like Dijkstra's or A* search) to each subgraph.
- Combine the results from each subgraph to find the overall shortest path.

Implementation:

- Use a programming language like Python or C++ to implement the brute force and divide and conquer algorithms.
- Utilize libraries like NetworkX or Boost Graph Library to work with graphs and implement graph algorithms.

Evaluation:

1. Compare the performance of brute force and divide and conquer algorithms in terms of computational time and solution quality.
2. Analyze the trade-offs between the two approaches and identify scenarios where one might be more suitable than the other.

Test file:

datapoints2.txt

This test file contains four test cases:

1. A simple route with 5 waypoints
2. A more complex route with 10 waypoints
3. A route with obstacles
4. A large route with 20 waypoints

Each test case specifies the number of waypoints and their coordinates.

The algorithm should read this input and output the shortest path or route that the self-driving car should take.

You can modify this test file to include more test cases or adjust the existing ones to better suit your specific requirements.