Project Description

The course project consists of applying your knowledge of data structures and algorithms design to solve ~~**two**~~ **out of six** of the following course project choices. You must design and implement an optimal algorithm to the problem and then demonstrating your work and results to the class and course instructors. Be prepared to not only have a solution to each problem, but to also provide an in-class presentation and demonstration of your work in addition to a project report including the algorithms you used (pseudocode), time complexity, and lastly the results of your algorithms on the 6 inputs provided.

K-Means Clustering

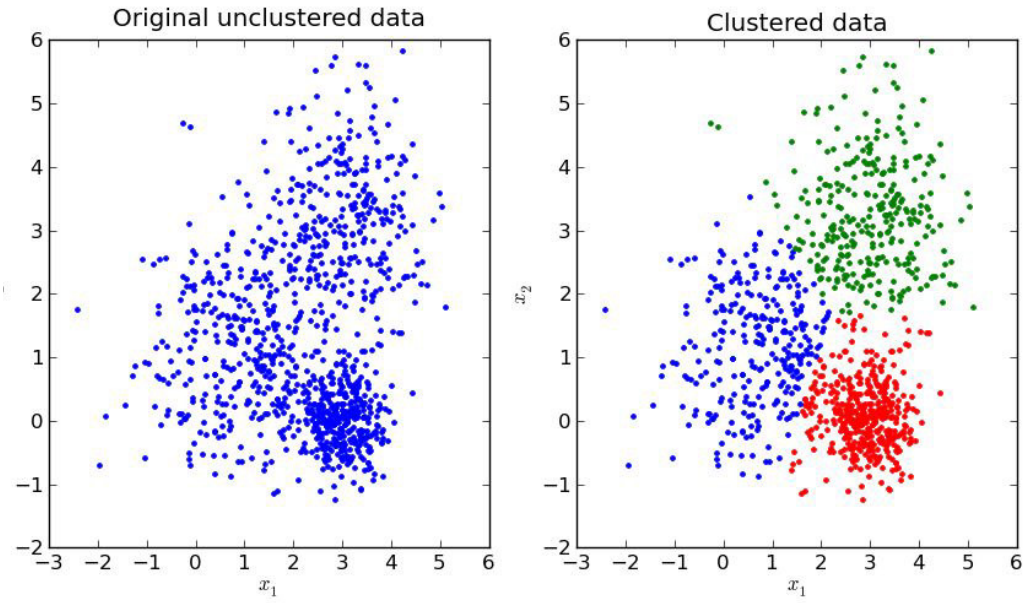
The K-means clustering algorithm is one of the most widely used methods for finding clusters (patterns) within data and is widely used in the field of Machine Learning. Your task is to implement the K-means clustering method algorithm as well as to use data structures to ensure that the algorithm performs well even with large amounts of data. For this project choice you must ensure that you meet all of the following criteria in your submission.

1. The problem will **only be for two-dimensions (2D)**, with a maximum of 4-5 clusters (groups within the data).

2. The K-means algorithm must be implemented by you and you **cannot use a library**, as k-means is widely used there are many libraries that already perform clustering, however you must implement the algorithm yourself.

3. You will be provided with sample data in the form of Comma-Separated Values (CSV) and then must use K-means to find the clusters in the data.

4. For each of the six data sets provided you must perform the K-means clustering algorithm, identify the clusters, and create a plot showing the results. You must plot all of the data and have all of the data in each cluster a different colour so the results can be easily verified. The results of your solution should generate a plot similar to the following, where each cluster is given a different colour.



Localization of Moving Object(s) in Grey-Level Images

Identifying moving objects within video feeds (which are made up of images) is a widely used task, especially for tracking the motion and trajectory of objects. Your task is to implement an algorithm to track moving objects across 10 - 30 images rather than using video data a collection of images will be provided where you must track the movement of objects between the frames. For this project choice you must ensure that you meet all of the following criteria in your submission.

1. The problem will consist only of grey-level images, similar to the example provided below. There will be six separate sets of data, each containing 10 - 30 grey level images that you must track the movement of the objects.

2. You must design and implement your own algorithm, you are allowed to use certain libraries for simple operations (reading / writing files) or drawing the tracking square on objects, however the core algorithm used to track the objects moving **must be written by you.**

3. Your solution must draw a red rectangle around the objects and track their movements across each of the 10 - 30 images provided.

Development of a Word Dictionary (Add, Delete, Edit, Search)

Dictionaries are widely used and we often take spellchecking for granted. You will be required to create a dictionary that not only provides the ability to search a word, but also to add new words, delete existing words, and edit the definition of a word in the dictionary. For this project you must ensure that you meet all of the following criteria.

1. You must implement a dictionary that contains not only words but also their definition.

2. You will be provided with a list of six different Comma-Separated Values (CSV) files containing words and their definitions, as well as for each a list of operations to perform.

3. Your solution must be able to read in the data provided and create the dictionary as well as process the accompanying file with the list of operations to perform.

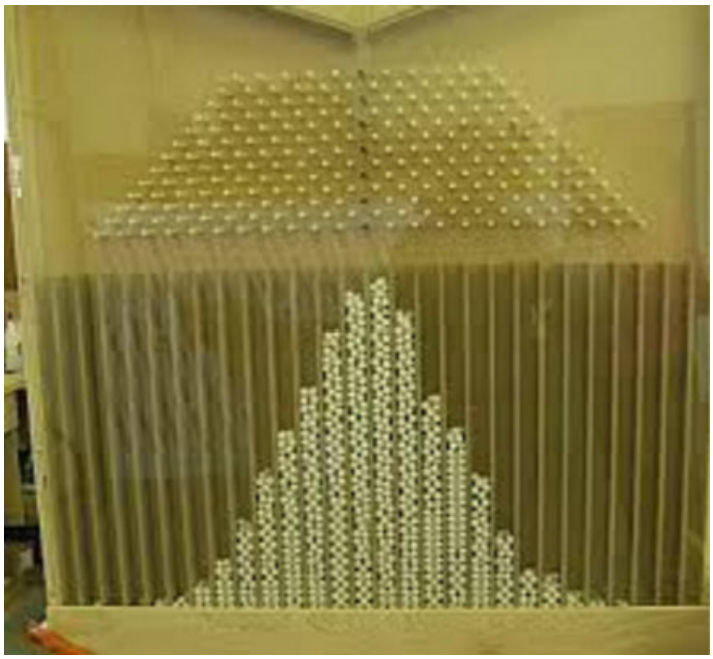
4. Your program should also have a CLI so that a user can perform each of the operations from the terminal such as searching for a word to retrieve the definition.

Implement a Galton Board Simulation

The Galton board simulates Central Limit Theorem (CLT), which roughly states that given enough independent random variables the total aggregate outcome of their results converges to an approximately normal distribution. For this project you must make sure to meet the following criteria.

1. You will be provided with six different numbers for the amount of random balls to drop (e.g. 1000, 10000, etc.).

2. For each random ball dropped (random value generated) you must show an animation of the Galton Board being created with each additional random point being generated. **You do not need to animate balls dropping**, but you must animate showing how the Galton Board statistical distribution is being created with each additional random data generated.



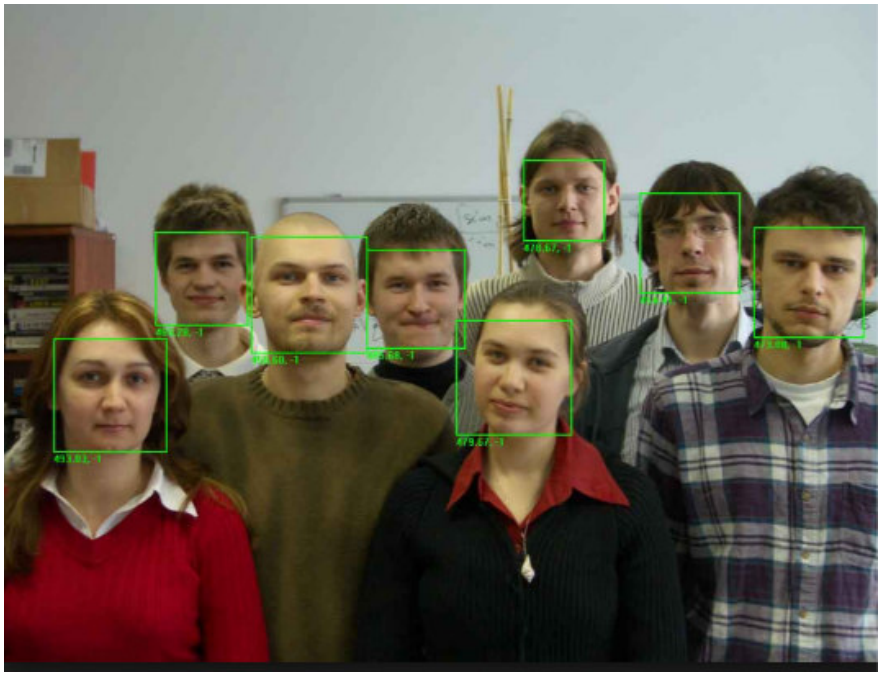
Face Localization in Colour Images

Detecting faces within images is a feature provided in almost all modern cameras and smartphones. For this project you must design and implement the algorithm for detecting in colour images all of the faces (if possible). Ensure that for this project that you meet all of the following criteria.

1. You will be given colour images with faces, you must write the algorithms to find all of the faces (or as many as possible), you can use libraries such as OpenCV to handle simple operations such as reading / writing image files, however **you must write your own algorithms** for the face detection.

2. There will be six pictures provided, each of the pictures provided will vary from lower resolution to HD images, your system should still be able to process high resolution photos and detect as many of the faces as possible.

3. Your algorithm must save the results as an image with a rectangle drawn around each of the faces that were detected.



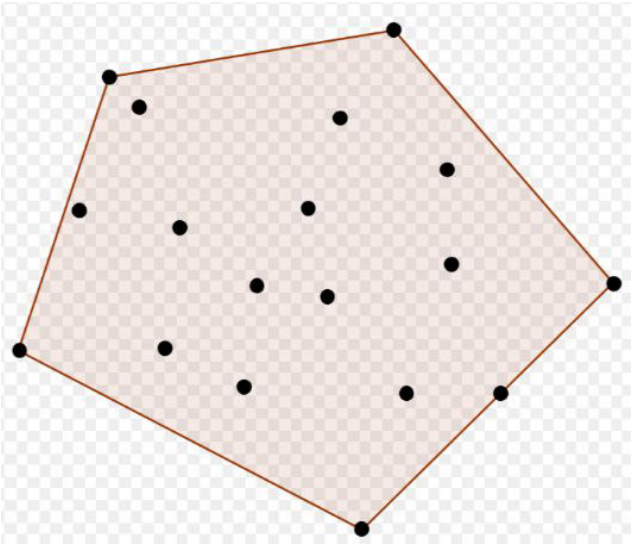
Finding the Convex Hull

The convex hull consists of finding the set of outermost points that encompass within them all of the other points. The Convex Hull problem can be thought of visually as a set of nails partially hammered into a board, where a rubber band has been wrapped around the outside edge of nails. For this project you must implement a solution to the convex hull problem that meets all of the following criteria.

1. You will be given six different sets of points (x and y coordinates) in Comma-Separated Values (CSV) files.

2. While you may base your solution on any of the well-known algorithms for solving the Convex Hull problem, you must design and **implement the algorithms in your own code** and cannot use existing libraries or tools to solve the problem.

3. You must plot an image showing the collection of points provided as well as the Convex Hull found that encloses all of the points within it. Your results should look similar to the following image.



Deliverables

For your project, you must submit a zip file containing your report in addition to all of the source code and related data for your project. You **must also ensure that you provide a README file** that briefly explains how to run each of your projects so that the instructors are able to easily evaluate your results.

1. You must provide your solution (all source codes) for two of the six project problems, your solution must meet all of the criteria specified for each project choice and must have results for each of the six data inputs provided. Ensure that you also include all of the results from each project (solutions, images, results, etc.) in addition to your source code. **[total 50 marks]**

2. You will give during class a presentation of each of your two projects that demonstrates your solution to the problems and shows the performance. **[total 20 marks]**
    - Demonstration of a working solution to both project problems **[10 marks]**
    - Demonstration of the performance (time complexity, plots) **[10 marks]**

3. You must also provide a detailed report that has an introduction and explains each of the problems, shows the time complexity analysis of your algorithms, and compare the performance results of your solution for the six datasets provided **[total 30 marks]**.
    - Introduction and explanation of each project problem as well as the challenges and your solution **[5 marks]**
    - Algorithms you used (pseudocode) and time complexity analysis for your algorithms **[10 marks]**
    - Performance plots for each project solution comparing the execution time for each of the six datasets **[10 marks]**
    - Conclusion and summary of your solutions and their results **[5 marks]**