**Project Submission Instructions**

You are required to submit, via Moodle, both a program (package) and a single PDF file report.

1. Program

* The program must be written in **Java or C++** and you are allowed to use any of the read-made libraries of Java or C++ (e.g, STL), those that implement data structures and algorithms.
* You are allowed to consult and borrow ideas from any source as long as (a) you **reference** the external sources, in your report, and (b) you **do not copy** and paste external code into your own program: re-write any externally-sourced code yourself, making sure your understand it.
* You are, however, allowed to use external libraries/programs for data **pre-processing** and data **post-processing** purposes, but these cannot form part of the program package you submit.
* Name the program **COEN352-#id-Project#-P**. Example COEN352-#yourID-Project1-P, if you choose Project 1
* Submit the **whole project including all dependencies** required for running the project.
* Please make sure the **project is compiling** before submission.
* Please include a **readme text file with all the necessary information to run** the project.

1. Report

* The Report should be a **PDF** file and should be named as **COEN352-#yourID-Project#-R**.
* The report should have the following sections.

1. **Problem**: Copy or – better – restate the problem statement in your own words.
2. **Methods**: A Pseudocode to illustrate the working of your program. An example is shown below- use the same format. These can be organized in a hierarchical fashion and should cover all important (not trivial) algorithms.

|  |
| --- |
| **Definitions:**  A = an array of unsorted N number.  Small = stores a small value.  Swap(A,i,j) = Swaps elements at index ‘i’ and ‘j’ in the array ‘A’.  **Algorithm:**  Procedure selection\_sort(A): \\ performs selection sort  For all I = 0 to N-1 do:  Small = I  For all J = I + 1 to N-1 do:  If A[J] < A[Small]  Small = J  End-If  End-For  Swap(A,I,Small)  End-For  End-Procedure |

1. **Results**.

For Project 1:

The goal is to achieve accurate recall of all similar images (given some distance measure: e.g., Hamming distance) as fast as possible.

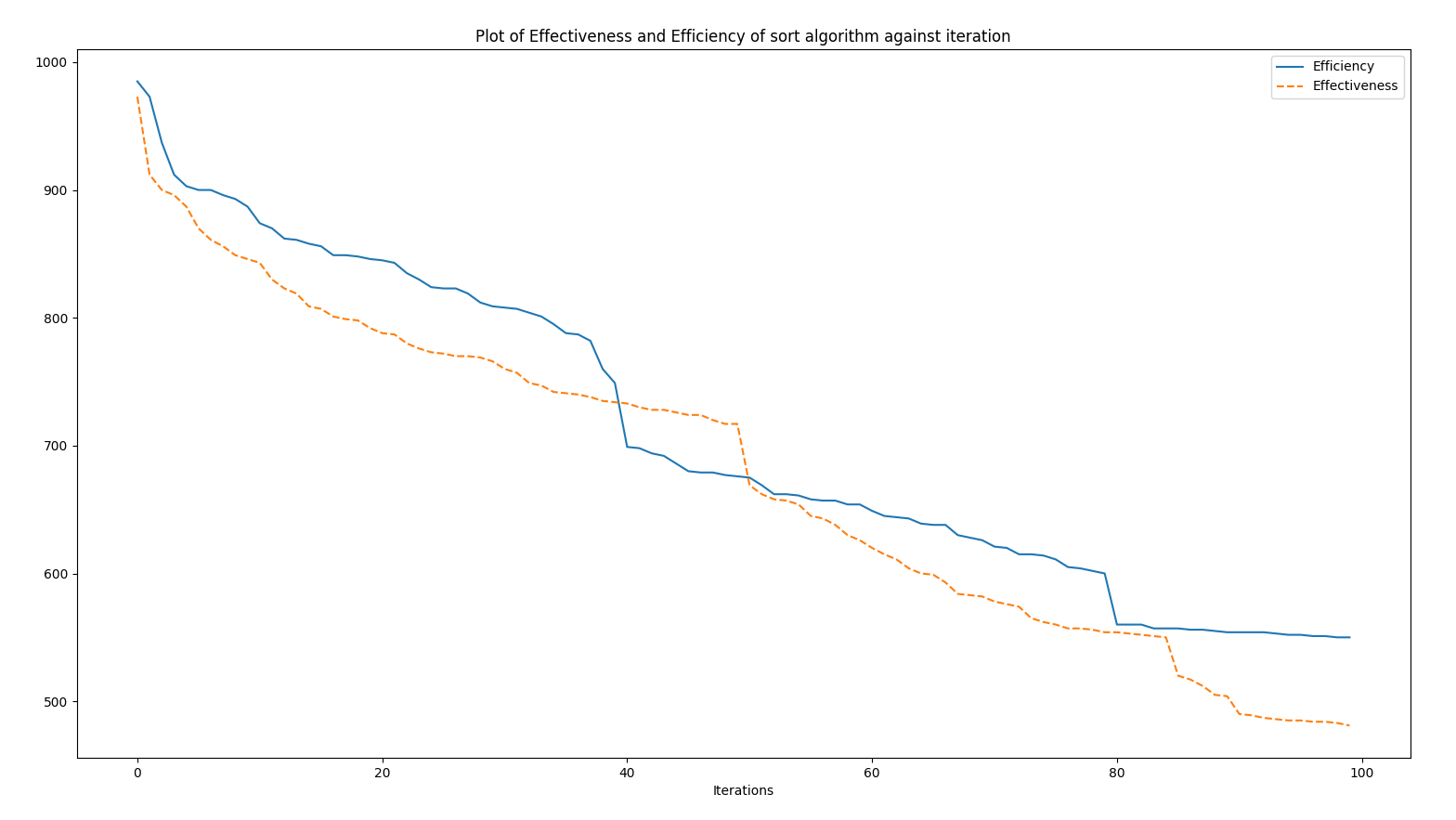
The results should have a table listing the top recalled images for a random selection (e.g., 10) of input image (each with a given threshold distance). An example table format is shown below.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input Image | Distance (threshold) | Top Recalled Images re- | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |

For Project 2:

The Goal is to optimize a random sort algorithm as much as possible.

The results for project 2 should have a plot for the effectiveness and efficiency of the sort algorithm at each iteration of generation. A sample plot is shown below. You need a plot for data category (4 in total).



Efficiency of the algorithm = average number of inversions (in output lists)

Efficiency of the algorithm = number of non-(n,n) swap instructions in the sorting algorithm

1. **Analysis** **of Results** & **Conclusions** + Future **Recommendations** (on how to do things better).

Have both the report and program in a single folder named **COEN352-#yourID-Project**. Submit the compressed folder.