Introduction to AI - Assignment I

School of Information Technology & Engineering
Addis Ababa University

Objectives of the assignment (30 points):

- Be comfortable with graphs and their operations
- Being able to implement graph search algorithms
- Benchmark and compare different search algorithms
- Analyze the results and deduce important findings

❖ Make sure to write reusable code so that you don't write the same algorithm multiple times
❖ You can use **jupyter-notebook** to write your algorithms
❖ Use **matplotlib.pyplot** library to plot the analysis

1. Develop a graph library. It should have the following functionalities
    a. Create a node
    b. Insert and delete edges and nodes
    c. Using your self-made graph library, try loading the graph data presented on page 83rd of the textbook. The file containing the cities will be given to you.

2. Implement BFS, DFS, UCS, Iterative Deepening, Bidirectional Search, Greedy, and A* Search algorithms. Using the graph from Question 2, evaluate each of your algorithms and benchmark them. Remember to run each experiment to run 10 times and report the average of the 5 trials.

    - The benchmark should be finding the path between randomly picked 10 cities from the graphs you generate. Find the path between them.

    - Create random graphs with a number of nodes n = 10, 20, 30, 40. Randomly connect nodes with the probability of edges p = 0.2, 0.4, 0.6, 0.8. In total, you will have 16 graphs. The generated node will have randomly generated **x**, and **y** value that represents their locations. Use those coordinates to compute the heuristic functions.

        i. Randomly select 10 nodes and apply the above algorithms to find paths between them in all 16 graph settings.

ii. Register the time taken to find a solution for each algorithm and graph. Run each experiment 5 times and have the average of the time taken in the five experiments.
iii. Use **matplotlib.pyplot** to plot their average time and solution (found path) length on each graph size.

- For each algorithm
  iv. What is the average time taken for each path search?
  v. What is the path length?

3. In graph theory and network analysis, indicators of centrality assign numbers or rankings to nodes within a graph corresponding to their network position. For example, in a given social network, questions like *who is an influencer? Who is the significant person? Who is the linkage between societies/groups?* can easily be answered by calculating node centrality rankings.

   There are several centralities types. Compute the Degree, Closeness, Eigenvector, Katz, PageRank, and Betweenness centralities on the graph from Question 2. (You have to read online how to calculate these centralities).

   1. Compute these centralities for each node
   2. Report a table containing top-ranked cities in each centrality category
   3. Summarize your observations

4. Compare the performance of different graph representations – Adjacency List, Adjacency Matrix, and Edge List – by evaluating their efficiency on various graph operations. Your task is to generate two tables: one for time comparison and the other for space comparison.

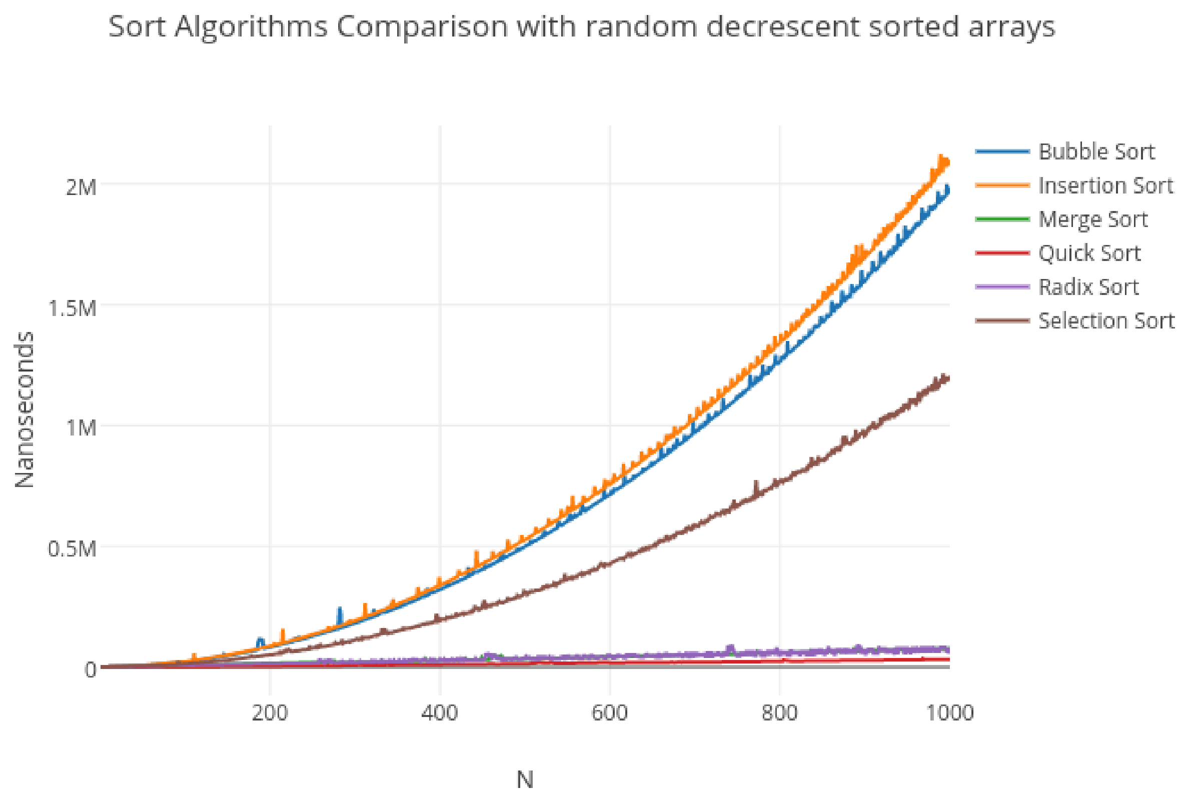
   Graph Representations: Adjacency List, Adjacency Matrix, Edge List
   Operations to be Evaluated: Insertion of vertices and edges, Deletion of vertices and edges, Checking for the existence of an edge, Finding neighbors of a vertex
   Evaluation Criteria: For each operation and representation, measure both time complexity and space complexity using algorithmic analysis.

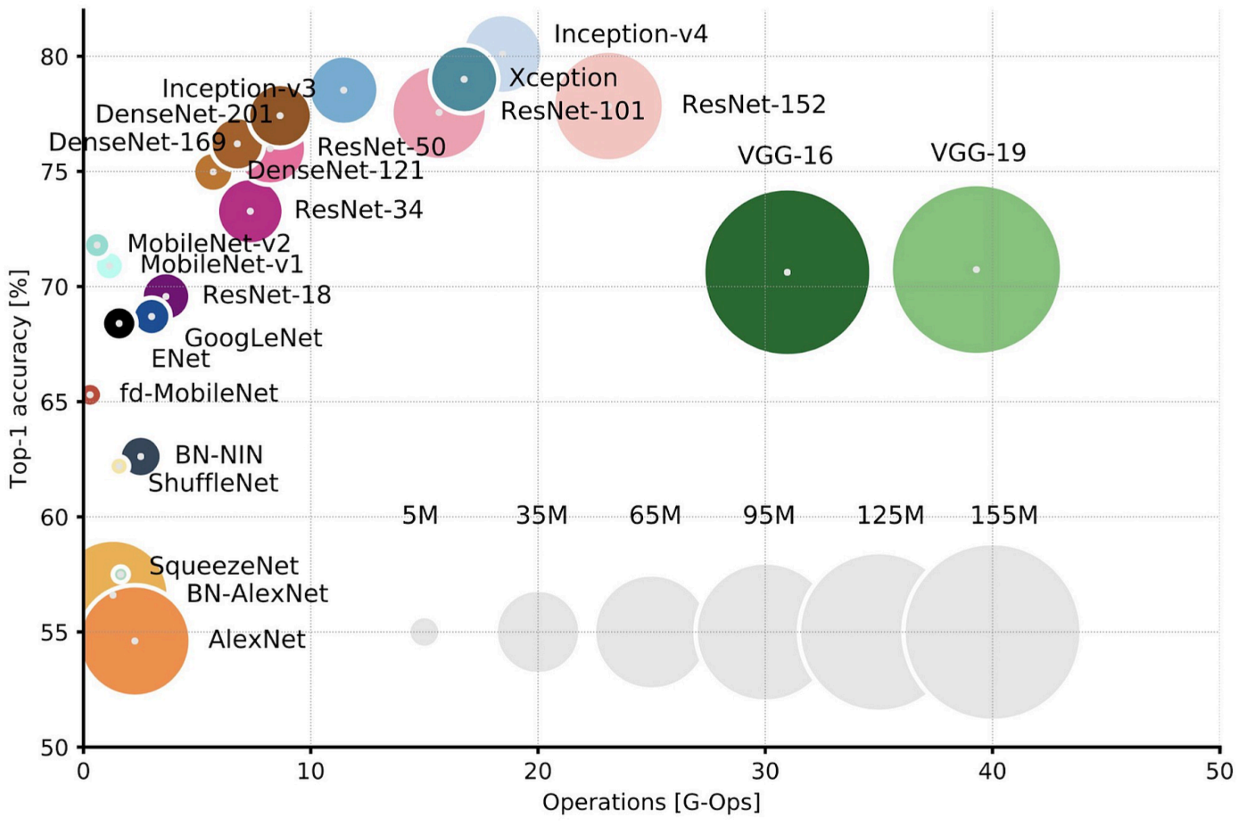
Deliverable:

- A PDF file showing the results of your experiments and your detailed observations. Include tables, graphs, and plots highlighting the results. In your analysis, compare the results and explain the differences and similarities you observed.
- Code organized with proper naming
- Have a readme.txt file you list the IDs of the group members
- Zip the code, the PDF file, and the readme.txt file in one folder and name it **Group-x,** where **x** is your group ID.

  Don't write definitions. Just the results.

Sample performance graph




Inception-v4
Xception
ResNet-101
ResNet-152
Inception-v3
DenseNet-201
DenseNet-169
ResNet-50
DenseNet-121
ResNet-34
VGG-16
VGG-19
MobileNet-v2
MobileNet-v1
ResNet-18
GoogLeNet
ENet
fd-MobileNet
BN-NIN
ShuffleNet
SqueezeNet
BN-AlexNet
AlexNet
5M
35M
65M
95M
125M
155M
Top-1 accuracy [%]
Operations [G-Ops]
50
55
60
65
70
75
80
0
10
20
30
40
50