Instructions

- For the first 3 problems, implement your solution in Python3. Inputs should be given by the user. You may assume that the inputs will be proper. For instance: the inputs of the Question 3 will be integers between [1,2...,9]. Write a driver function to test each of these algorithms. Gather all of the python code in a single .py file. Pay attention to clean coding.
- Write a report explaining the reasoning behind the algorithms you coded and analyze the worst-case time complexity of each of them. This report should also include your answers to Question 4 and Question 5. Write your report by using a program like MS Office and then convert it to a single PDF file. Pictures of handwritten works are not accepted.
- Upload two files only, a .py file and a .pdf file, not a .zip or a .rar file.

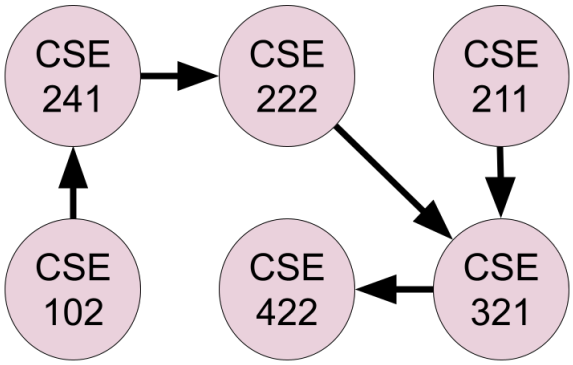
CSE 321 - Homework 3

Due date: 8/12/2022, 23:59

1. A directed acyclic graph (DAG) is a directed graph with no directed cycles. A DAG has at least one node with indegree (number of incoming edges) 0 and one node with outdegree (number of outgoing edges) 0.

   Topological sorting of a DAG is a linear ordering of nodes such that for every edge ab (an edge from node a to node b), a comes before b in the ordering. A DAG may have more than one topological order.

   Example: The following graph presents some of the courses in our department. The directed edges indicate the prerequisites. For instance, a student cannot enroll in $CSE321$ without successfully passing the courses $CSE211$ and $CSE222$. On the other hand, $CSE102$ and $CSE211$ do not have any prerequisites. Topological sortings of this graph should demonstrate the order of these courses such that if a student follows that ordering, they meet all of the prerequisite requirements.

   Such orderings are:

   - $CSE102 \rightarrow CSE241 \rightarrow CSE222 \rightarrow CSE211 \rightarrow CSE321 \rightarrow CSE422$
   - $CSE102 \rightarrow CSE241 \rightarrow CSE211 \rightarrow CSE222 \rightarrow CSE321 \rightarrow CSE422$
   - $CSE102 \rightarrow CSE211 \rightarrow CSE241 \rightarrow CSE222 \rightarrow CSE321 \rightarrow CSE422$
   - $CSE211 \rightarrow CSE102 \rightarrow CSE241 \rightarrow CSE222 \rightarrow CSE321 \rightarrow CSE422$

   (a) ***15 pts.*** Construct a DFS-based algorithm to obtain such an ordering for a given DAG.

   (b) ***15 pts.*** Construct a non-DFS-based algorithm to obtain such an ordering for a given DAG.

PS: The algorithms should find a single solution. You are not asked to find all possible orderings.

2. ***20 pts.*** Design an algorithm to calculate a^n, where $a \in Z$ and $n \in Z^+$, with a worst-case time complexity of $O(logn)$.

3. ***20 pts.*** Design an algorithm to solve $9x9$ sudoku game by using exhaustive search.

4. ***15 pts.*** Sort the following array in ascending order by using insertion sort, quick sort, and bubble sort algorithms. Determine whether they are stable sorting algorithms or not. Justify your answer on the given array.

   $array = \{6, 8, 9, 8, 3, 3, 12\}$

5. (a) ***5 pts.*** Give an explanation of the relation between brute force and exhaustive search.

   (b) ***5 pts.*** Learn about the basic idea of Caesar's Cipher and AES. Are they vulnerable to brute force attacks? Explain your answer.

   (c) ***5 pts.*** Why does the naive solution to primality testing (for input n, checking if $x \in \{2, 3, ..., n-1\}$ divides n) grow exponentially?

Important Notes

- For the first 3 problems, implement your solution in Python3. Write a driver function to test each of these algorithms (inputs should be given by the user). Gather all of the python code in a single .py file. Pay attention to clean coding.
- Write a report explaining the reasoning behind the algorithms you coded and analyze the worst-case time complexity of each of them. This report should also include your answers to Question 4 and Question 5. Write your report by using a program like MS Office and then convert it to a single PDF file. Pictures of handwritten works are **not accepted**.
- Upload two files only, a .py file and a .pdf file, **not a .zip or a .rar file**.