Bank of Exercises, Series 1

- This document includes a collection of various exercises from the previous exams of the Aglo. course.

- Note that this file is not an exam example (in the format that you will see during the actual exam), but a collection of possible exercises.

- Also, note that "some" exercises of this document are different from the types of the exercises in "exam_example" that I have uploaded it on BB. So, be prepared for exercises in the actual exam, similar to any of these types of exercises.

- In this file, for each exercise, I have added some annotation, hints or tips to find the solution, but not a full solution.

- We may dedicate an extra lecture/session where you can ask your questions about these exercises (we will discuss it during the last lecture on Fri. 3/3).

Exercise A (10 points)

Write the pseudocode of the Insertion sort and Quicksort algorithms. Describe how they work on the array $A = \langle 15, 7, 4, 12, 11, 6, 10, 8, 16, 5 \rangle$. State the worst- and best-case computational complexity of the two algorithms in terms of the size $|A|$ of the input array, and explain why.

Same as exam_example, discussed in the class

Exercise B (12 points)

Construct a Binary Search Tree (BST) from the array of integers A in the previous exercise. Do this by inserting each integer into the tree, one by one, starting from the left (element '15'). Draw the resulting tree and state its height. Is the tree balanced?

Remove the third element of A (element '4') from the BST, explaining the operations that this procedure performs. Now, insert the element you removed back into the BST, and draw the resulting tree. Is the resulting BST different from the BST before the element was removed? If so, will this be the case for any element that is removed and then re-inserted?

Write the pseudocode of the algorithm for searching for an element in a BST. What is the computational complexity of the algorithm? Show how the algorithm works, step by step, for the value 12.

Same as exam_example, discussed in the class

Exercise C (12 points)

The *Fibonacci sequence* is the sequence of natural numbers starting from 0 and 1, where each number is the sum of the two preceding numbers. In other words:

$$F(n) = \begin{cases} n & \text{if } n \leq 1 \\ F(n-1) + F(n-2) & \text{otherwise} \end{cases}$$

The first few values in the sequence are

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \ldots.$$

Write a *recursive* algorithm that computes the n-th Fibonacci number $F(n)$. State the time complexity of your algorithm in terms of the input number n.

Consider the following *iterative* algorithm for computing $F(n)$:

```
FIB_IT(n)
1  f = new array of size n + 1
2  f[0] = 0
3  f[1] = 1
4  for i = 2 to n
5        f[i] = f[i − 1] + f[i − 2]
6  return f[n]
```

State the complexity of the iterative algorithm in terms of the input number n. Is the iterative algorithm more or less efficient than your recursive algorithm?

discussed in D.P lecture

Turning a recursive formula into a recursive algorithm should be an easy task

Exercise D (12 points)

The binomial coefficient $\binom{n}{k}$ counts the ways of selecting k elements out of a set of n elements. The binomial coefficient can be defined as

$$\binom{n}{k} = \begin{cases} 1 & \text{if } k = 0 \text{ or } k = n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{otherwise} \end{cases}$$

or, equivalently, as

$$\binom{n}{k} = \begin{cases} 1 & \text{if } k = 0 \\ \left(\binom{n}{k-1}(n-k+1)\right)/k & \text{otherwise} \end{cases}$$

naive

Implement a recursive algorithm for each of the two definitions. Do the two algorithms have the same asymptotic complexity? If not, which one is more efficient? Please explain your answer.

This part is challenging!

hints:
1- one is called bad recursion, one good recursion
2- for each, draw the tree of recursive calls and see how they look like

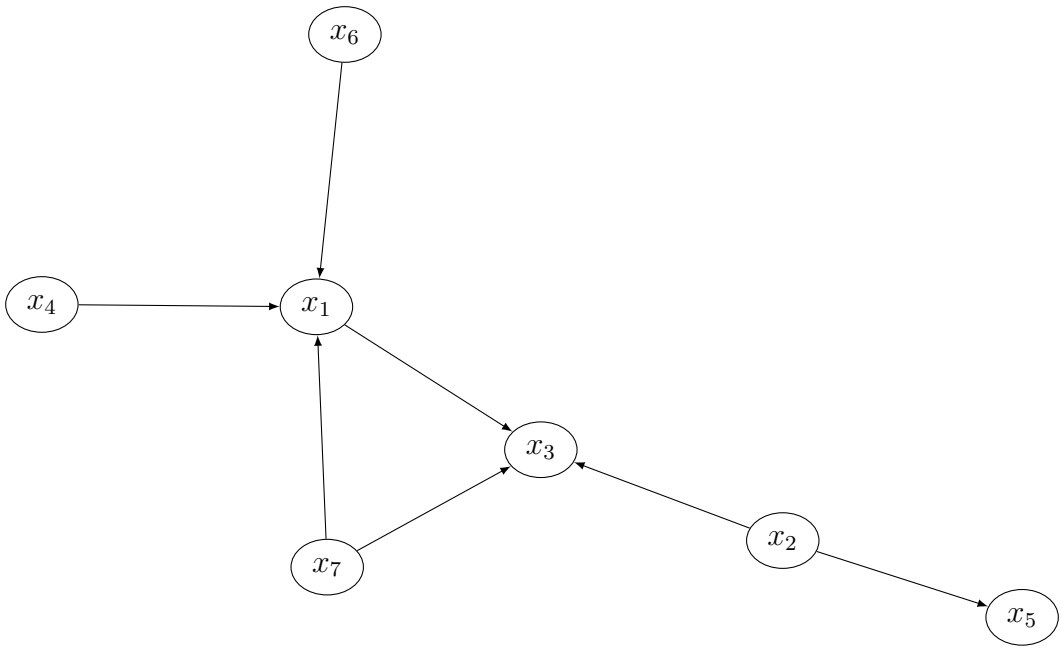
– other way of showing second recursive formula is: $\binom{n}{k} = \binom{n-1}{k-1} \cdot \frac{n}{k}$

– note that $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ which means $\binom{n}{k} = \binom{n}{n-k}$

– think of e.g., $\binom{10}{7} = \binom{10}{3}$

Exercise E (12 points)

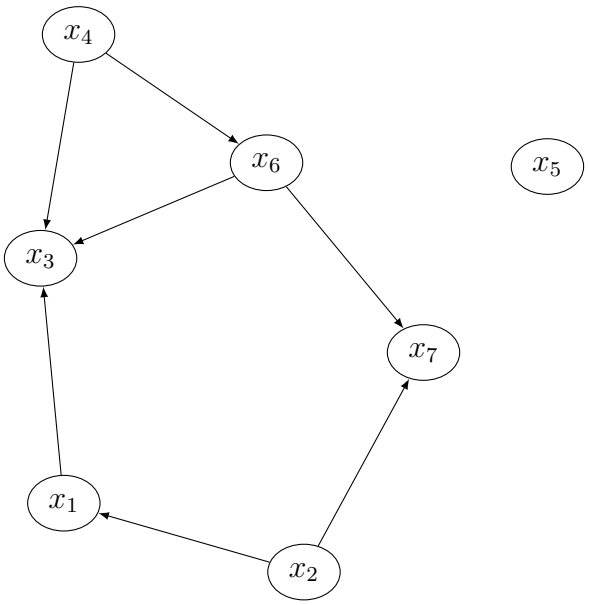
You are given a graph $G = (V, E)$, where eacn $x_i \in V$ represents a task, and each edge $(x_i, x_j) \in E$ represents the dependency "x_i must be completed before x_j". Write the pseudocode of an algorithm that finds an execution sequence for the tasks such that all dependencies are satisified. Illustrate how the algorithm works on the graph below. State the complexity of this algorithm in terms of the number of vertices $|V|$ and the number of edges $|E|$ of the input graph.



see notes on next page, same question

Exercise F (12 points)

You are given a graph $G = (V, E)$, where eacn $x_i \in V$ represents a task, and each edge $(x_i, x_j) \in E$ represents the dependency "x_i must be completed before x_j". Write the pseudocode of an algorithm that finds an execution sequence for the tasks such that all dependencies are satisified. Illustrate how the algorithm works on the graph below. State the complexity of this algorithm in terms of the number of vertices $|V|$ and the number of edges $|E|$ of the input graph.



- Think of the algorithm to "order" vertices in such a graph

- Should G be DAG for such dependencies?

- Does it matter from where to start?

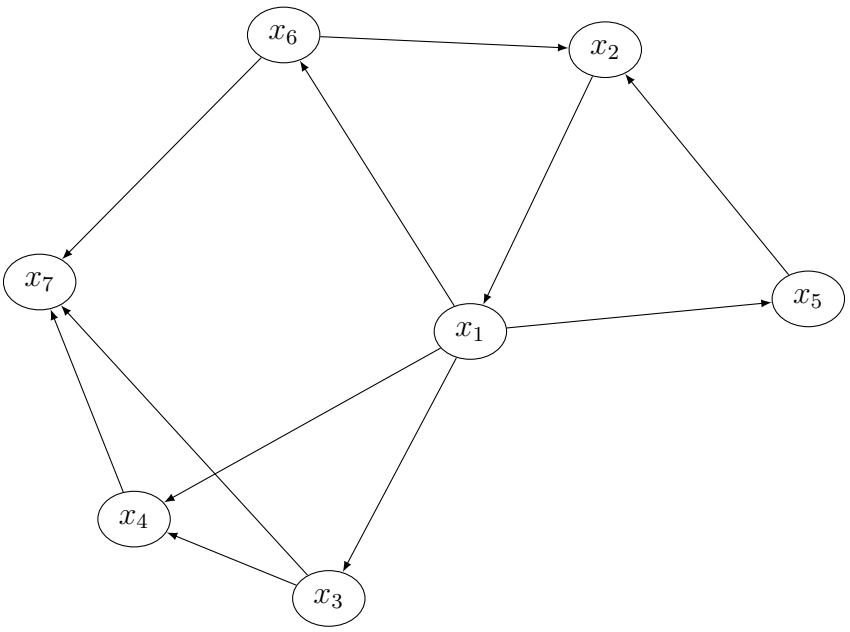
Exercise G (12 points)

Given a directed, unweighted graph $G = (V, E)$, describe an algorithm to solve each of the following problems:

Traversal algos on graph

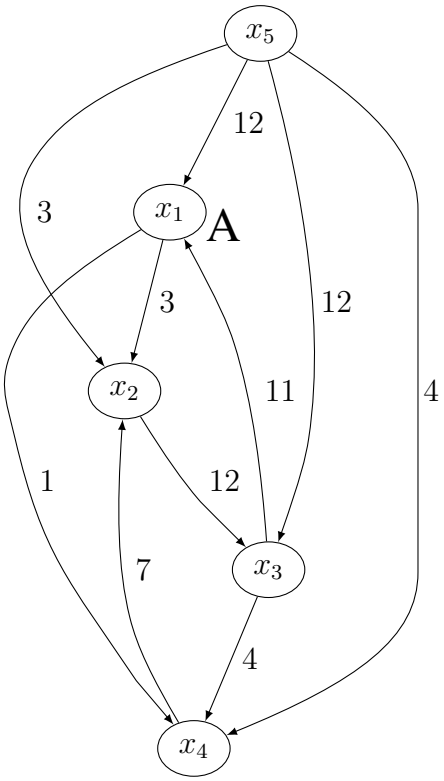
- Return whether the graph is connected.
- Return whether the graph is a tree. no. of edges or check cycle
- Find all the strongly connected components of the graph. check lectures
- Find the shortest path from node x_1 to every other node. check lectures

Show all the steps performed by each algorithm, using as input the graph G given below. Also, state and explain the computational complexity required to solve each problem in terms of the number of nodes $|V|$ and number of edges $|E|$ of the input graph.



Exercise H (12 points)

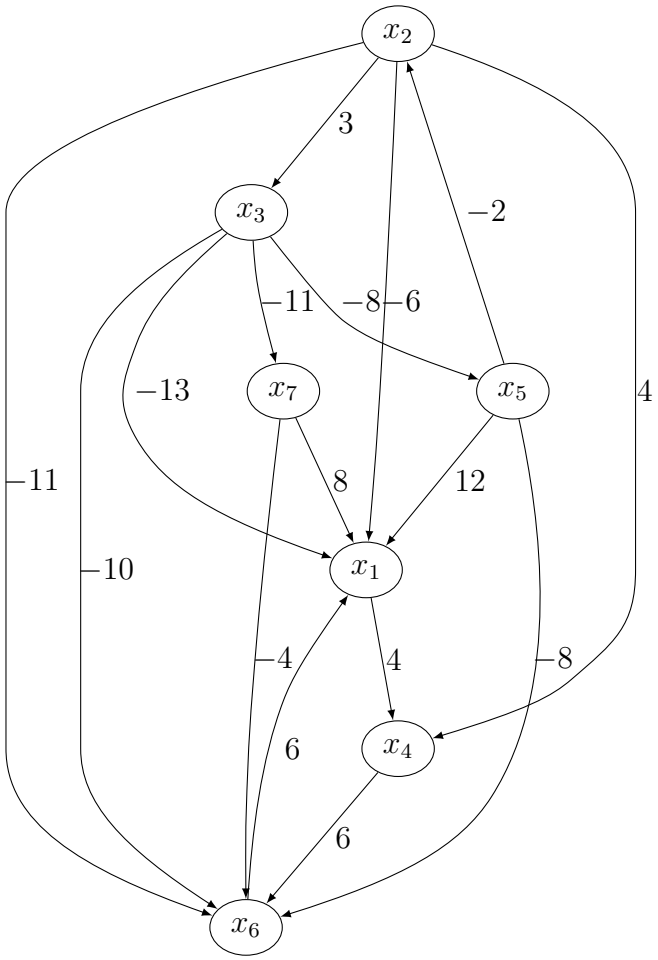
Write the pseudocode of an algorithm for finding the shortest path from a given source vertex to all other vertices in a weighted directed grah $G = (V, E)$. Illustrate how the algorithm works on the graph below for a few steps, showing in particular how the algorithm updates the distance estimate $d[x_i]$ of one of the vertices $x_i \in V$. State the computational complexity of the algorithm in terms of the number of vertices $|V|$ and number of edges $|E|$ of the input graph.



- We have seen same ex. in exam_example,
- discussed in the class

Exercise I (12 points)

Given the following directed graph $G = (V, E)$ with weight funtion $w : E \mapsto \mathbb{Z} \setminus \{0\}$ as specified on the edges of the graph, illustrate an algorithm for finding a shortest path from vertex x_1 to every other vertex in the graph. Show how, at each step, the algorithm updates the distance estimate $d[x_i]$ of every vertex $x_i \in V$. State the computational complexity of the algorithm in terms of the number of nodes $|V|$ and number of edges $|E|$ of the input graph.



- We have seen same ex. in exam_example,
- discussed in the class

Exercise J (12 points)

State whether each of the following 6 statements on the asymptotic behavior of the function f is true or false.

1.	$f(n) = 4n\log(n)$,	$g(n) = 9^n$,	$f(n) \in \Theta(g(n))$
2.	$f(n) = 2n^3 - 3n^2$,	$g(n) = 9^n$,	$f(n) \in \omega(g(n))$
3.	$f(n) = 2\log(n)$,	$g(n) = 8^n$,	$f(n) \in \Theta(g(n))$
4.	$f(n) = 3n$,	$g(n) = 4n^3 - 4n^2 + 2n - 4$,	$f(n) \in \Omega(g(n))$
5.	$f(n) = 8n\log(n)$,	$g(n) = 2n\log(n)$,	$f(n) \in o(g(n))$
6.	$f(n) = 6\log(n)$,	$g(n) = \log(n)$,	$f(n) \in O(g(n))$

- We have seen same ex. in exam_example,
- discussed in the class

Exercise K (12 points)

State whether each of the following 6 statements on the asymptotic behavior of the function f is true or false.

1.	$f(n) = 2$,	$g(n) = 4$,	$f(n) \in \omega(g(n))$
2.	$f(n) = 2n^3 + n^2$,	$g(n) = 3n^4 - 4n$,	$f(n) \in O(g(n))$
3.	$f(n) = 5^n$,	$g(n) = 7^n$,	$f(n) \in O(g(n))$
4.	$f(n) = 2^n$,	$g(n) = 4n^3 - 3n^2 + 4n$,	$f(n) \in \Theta(g(n))$
5.	$f(n) = n^2 + 4n$,	$g(n) = 2\log(n)$,	$f(n) \in o(g(n))$
6.	$f(n) = 2n^3 - n$,	$g(n) = 2^n$,	$f(n) \in \omega(g(n))$

- We have seen same ex. in exam_example,
- discussed in the class

You are likely to see similar exercise to this one in your upcoming exam. Such exercises are an effective way to evaluate your understanding of the topic without requiring extensive pseudocode or written responses

Exercise L (20 points)

Answer the following statements with either True or False. Correct answers give 1 point. If the statement ends with the sentence "(explain your answer)", you can get an additional point by providing a correct and more detailed answer to the corresponding question.

S1 The worst-case computational complexity of Breadth-First Search (BFS) is asymptotically higher than the worst-case computational complexity of Depth-First Search (DFS) (explain your answer).

False:

Why?

S2 Given an undirected graph $G = (V, E)$ and a source node $s \in V$, Breadth-First Search BFS(G, s) computes the shortest path $\delta(s, v)$ between s and all nodes $v \in V$ (explain your answer).

True:

Why?

S3 Given a directed graph $G = (V, E)$ with weight function $w : E \to \mathbb{R}$, the algorithm BELLMAN-FORD(G, w, s) returns FALSE if and only if there exists a cycle p such that $\sum_{(i,j)\in p} w_{ij} < 0$.

True.

S4 Given a directed acyclic graph $G = (V, E)$ with weight function $w : E \to \mathbb{R}$ and a topological sort of the nodes in V, computing the shortest path between a source node s and all other nodes requires at least $|V|^2$ calls to RELAX(u, v, w) (explain your answer).

False:

Why?

S5 Dijkstra's algorithm computes the minimum distances between a given source and all nodes in a graph (explain your answer).

True:

True, if...?

S6 The Floyd-Warshall algorithm computes the minimum distances between all pairs of nodes in a weighted graph.

True.

S7 Given a dynamic set with n elements, the minimum depth of a Binary Search Tree representation of the set is $n \log n$ (explain your answer).

False:

S8 All known algorithms for sorting a dynamic set have exponential worst-case complexity (explain your answer).

False: give an example of a contradiction

S9 If $f(n) = n^3/2 - 2n$ and $g(n) = 3n$, then $f(n) = O(g(n))$ (explain your answer).

False: because there is no c, such 0<= ?(n) <= c.?(n)

S10 A stack is a dynamic set which follows the First-In-First-Out (FIFO) policy.

T or F?

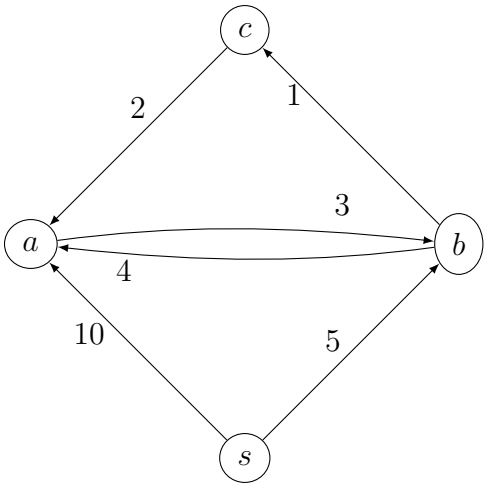
S11 The worst-case running time of the Insersion sort algorithm for sorting dynamic sets is the same as the worst-case running time of the Quick-sort algorithm (explain your answer).

True: Why?

Exercise M(10 points)

Describe Dijkstra's algorithm, answering in particular the following questions: what problem does the algorithm solve? Under which conditions is the algorithm correct (which assumptions does it make on the input graph)? What is the worst-case computational complexity of the algorithm?

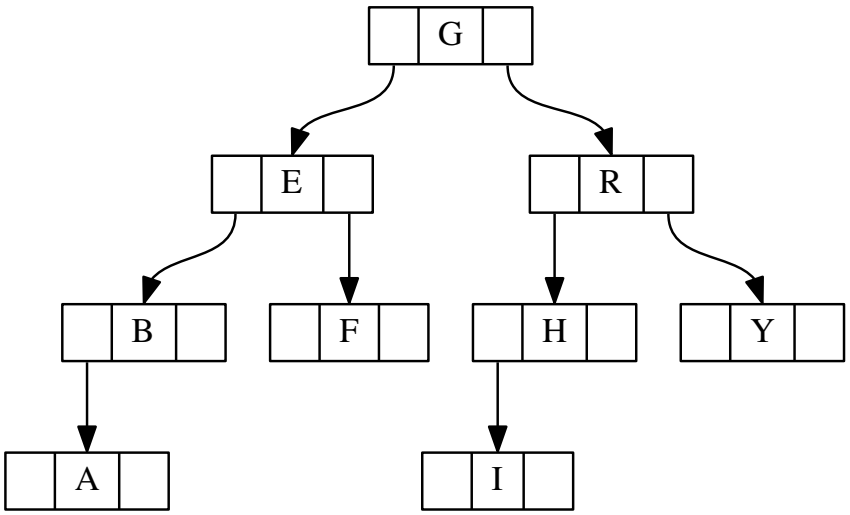
Illustrate how the algorithm works with the following directed weighted graph, where s is the source node.



Exercise N (10 points)

Illustrate the Binary Search Tree (BT) data structure. Describe the algorithms for inserting a new element into a BT, finding the maximum element in a BT, finding the successor (or predecessor) of an element in a BT. What is the computational cost of these operations? Why is it important that BTs are balanced?

Illustrate how the BT below is modified after performing the following two operations in order: insert a new element with key "D"; remove the element with key "H".



- We have seen a kind of similar ex. in exam_example,
- discussed in the class

A relatively challenging question!

You may see such an ex. in the exam as a bonus!

Exercise O (10 points)

You are given graph $G = (V, E)$ with $|V| = n$ vertices and weight function $w : E \to \mathbb{R}$. You are also given a $n \times n$ matrix

$$W^{(0)} = \left(w_{ij}^{(0)}\right),$$

where $w_{ij}^{(0)}$ is the value in the i-th row and j-th column. The matrix represents the weights of the edges in G, that is,

$$w_{ij}^{(0)} = \begin{cases} w(i,j), & \text{if } (i,j) \in E, \\ \infty, & \text{otherwise.} \end{cases}$$

The pseudo-code below is a dynamic programming algorithm which takes as input the matrix $W^{(0)}$ (you have studied this algorithm in the course).

```
MYSTERY-ALGORITHM(W^(0))
1  n = W.rows
2  for k = 1 to n
3        let W^(k) = (w_ij^(k)) be a new n × n matrix
4        for i = 1 to n
5              for j = 1 to n
6                    w_ij^(k) = min(w_ij^(k-1), w_ik^(k-1) + w_kj^(k-1))
7  return W^(n)
```

What does the algorithm do? How does the algorithm modify the input matrix? What is the computational complexity of the algorithm?

You will see the algorithm related to this ex. in the last lecture,
on the topic of APSP.
By checking the slide you will understand what algorithm I mean.