<table>
<tr><td colspan="2">TOPIC 1</td></tr>
<tr><td>Title:</td><td>Performance comparison of the DSW algorithm working over a reference-based binary tree and an array-based binary tree</td></tr>
<tr><td>Tasks:</td><td>

Implement the DSW algorithm to work over a reference-based binary tree and an array-based binary tree

Including the capability to visualize at all steps
Provide a link to the implementation’s public Github repository
Provide instructions to run the code in the README.md file

If the code is Python, use virtual environments
If in another language, use docker

Provide the datasets in the repository

Run experiments to show the difference in time and memory complexity

Scenarios include search, insert, and delete
Experiments must be run over incremental dataset sizes
Integer types are enough

Show performance graphs and discuss the experiments’ results

Note if the results are expected
If the results diverge from expectations, explain the reason

</td></tr>
<tr><td>Seminar structure:</td><td>The seminar must have at least 3000 words or between 6 and 8 pages including figures. The seminar text should be single-spaced.
The seminar must at least have the following sections:

Introduction
Algorithm presentation
Experiment setup
Experiment results
Discussion and conclusion

</td></tr>
<tr><td>Note:</td><td>Use benchmarking tools to gather your results, such as Python profilers https://docs.python.org/3/library/profile.html, gperf, or valgrind.

Be sure to use a sequential memory data structure for arrays. Python lists are not adequate for this, but you can use arrays https://docs.python.org/3/library/array.html or NumPy arrays.

For visualization we suggest using GraphViz.

Experiments MUST be reproducible.</td></tr>
</table>

<table>
<tr><td colspan="2">TOPIC 2</td></tr>
<tr><td>Title:</td><td>Performance comparison of AVL tree implementations over a reference-based binary tree and an array-based binary tree</td></tr>
<tr><td>Tasks:</td><td>

Implement the AVL tree to work over a reference-based binary tree and an array-based binary tree

Including the capability to visualize at all steps
Provide a link to the implementation’s public Github repository
Provide instructions to run the code in the README.md file

If the code is Python, use virtual environments
If in another language, use docker

Provide the datasets in the repository

Run experiments to show the difference in time and memory complexity

Scenarios include search, insert, and delete
Experiments must be run over incremental dataset sizes
Integer types are enough

Show performance graphs and discuss the experiments’ results

Note if the results are expected
If the results diverge from expectations, explain the reason

</td></tr>
<tr><td>Seminar structure:</td><td>The seminar must have at least 3000 words or between 6 and 8 pages including figures. The seminar text should be single-spaced.
The seminar must at least have the following sections:

Introduction
Algorithm presentation
Experiment setup
Experiment results
Discussion and conclusion

</td></tr>
<tr><td>Note:</td><td>Use benchmarking tools to gather your results, such as Python profilers https://docs.python.org/3/library/profile.html, gperf, or valgrind.

Be sure to use a sequential memory data structure for arrays. Python lists are not adequate for this, but you can use arrays https://docs.python.org/3/library/array.html or NumPy arrays.

For visualization we suggest using GraphViz.

Experiments MUST be reproducible.</td></tr>
</table>

TOPIC 3	
Title:	**Performance comparison of an AVL tree and B-tree**

Tasks:	• Implement an AVL tree and B-tree with varying m-values
	  ○ Including the capability to visualize at all steps
	  ○ Provide a link to the implementation's public Github repository
	  ○ Provide instructions to run the code in the README.md file
	  ▪ If the code is Python, use virtual environments
	  ▪ If in another language, use docker
	  ○ Provide the datasets in the repository
	• Run experiments to show the difference in time and memory complexity
	  ○ Scenarios include search, insert, and delete
	  ○ Experiments must be run over incremental dataset sizes
	  ○ Integer types are enough
	• Show performance graphs and discuss the experiments' results
	  ○ Note if the results are expected
	  ○ If the results diverge from expectations, explain the reason

Seminar structure:	The seminar must have at least 3000 words or between 6 and 8 pages including figures. The seminar text should be single-spaced. The seminar must at least have the following sections: 1. Introduction 2. Algorithm presentation 3. Experiment setup 4. Experiment results 5. Discussion and conclusion
Note:	Use benchmarking tools to gather your results, such as Python profilers https://docs.python.org/3/library/profile.html, gperf, or valgrind. Be sure to use a sequential memory data structure for arrays. Python lists are not adequate for this, but you can use arrays https://docs.python.org/3/library/array.html or NumPy arrays. For visualization we suggest using GraphViz. Experiments **MUST** be reproducible.

TOPIC 4	
Title:	**Performance comparison of a (Prefix) Trie and Patricia trie**

Tasks:	• Implement a Prefix trie and a Patricia trie over a tree
	  ○ Including the capability to visualize at all steps
	  ○ Provide a link to the implementation's public Github repository
	  ○ Provide instructions to run the code in the README.md file
	  ▪ If the code is Python, use virtual environments
	  ▪ If in another language use docker
	  ○ Provide the datasets in the repository
	• Run experiments to show the difference in time and memory complexity
	  ○ Scenarios include search, range-search. insert, and delete
	  ○ Experiments must be run over incremental dataset sizes
	• Show performance graphs and discuss the experiments' results
	  ○ Note if the results are expected
	  ○ If the results diverge from expectations, explain the reason

Seminar structure:	The seminar must have at least 3000 words or between 6 and 8 pages including figures. The seminar text should be single-spaced. The seminar must at least have the following sections: 1. Introduction 2. Algorithm presentation 3. Experiment setup 4. Experiment results 5. Discussion and conclusion
Note:	Use benchmarking tools to gather your results, such as Python profilers https://docs.python.org/3/library/profile.html, gperf, or valgrind. Be sure to use a sequential memory data structure for arrays. Python lists are not adequate for this, but you can use arrays https://docs.python.org/3/library/array.html or NumPy arrays. For visualization we suggest using GraphViz. Experiments **MUST** be reproducible.

TOPIC 5	
Title:	**Performance comparison of a Suffix array and a Prefix trie if the strings are inverted before insertion**

Tasks:	• Implement a Suffix array and a Prefix trie where strings are first inverted before insertion
	  ○ Including the capability to visualize at all steps
	  ○ Provide a link to the implementation's public Github repository
	  ○ Provide instructions to run the code in the README.md file
	  ▪ If the code is Python, use virtual environments
	  ▪ If in another language, use docker
	  ○ Provide the datasets in the repository
	• Run experiments to show the difference in time and memory complexity
	  ○ Scenarios include search, range-search. insert, and delete
	  ○ Experiments must be run over incremental dataset sizes
	• Show performance graphs and discuss the experiments' results
	  ○ Note if the results are expected
	  ○ If the results diverge from expectations, explain the reason

Seminar structure:	The seminar must have at least 3000 words or between 6 and 8 pages including figures. The seminar text should be single-spaced. The seminar must at least have the following sections: 1. Introduction 2. Algorithm presentation 3. Experiment setup 4. Experiment results 5. Discussion and conclusion
Note:	Use benchmarking tools to gather your results, such as Python profilers https://docs.python.org/3/library/profile.html, gperf, or valgrind. Be sure to use a sequential memory data structure for arrays. Python lists are not adequate for this, but you can use arrays https://docs.python.org/3/library/array.html or NumPy arrays. For visualization we suggest using GraphViz. Experiments **MUST** be reproducible.