CSE331 : data structure and algorithms

Project Description

Guidelines

- Please form teams of 6 to 10 members.
- You need to fill this form before Saturday the 22th of November (https://forms.gle/NKeFALGCVf3cXEHK6) - only one submission per team
  You will lose marks in case of failure to fill in the form on time!
- For the project, you can use implementations of the following data structures: arrays, dynamic arrays (vectors), linked lists, stacks, queues, priority queues (heap).
- You **MUST** implement any other data structure **from scratch**.
- You should use GitHub/Gitlab to manage your projects (Your project should be private so if you can't create private repos on Github then you should use Gitlab).
  It's really important to learn how to use Git and Github now.
  Those courses are nice ones
  (Git: https://www.udacity.com/course/version-control-with-git--ud123)
  (READMEs and markdown: https://www.udacity.com/course/writing-readmes--ud777)

Submission

- You will need to submit **a short video** to demonstrate the different features of the program.
- You will need to write a **report to describe the project**. Imagine that you are describing the project/ data structures used to someone who has a coding background but who doesn't know much about the project.
    - The document should contain a number of words in the range of 1000 to 2000 words and should contain diagrams/ screenshots that demonstrate the project's background and implementation details.
    - The report should be divided into sections such as background, implementation details and complexity of operations.
    - The report should contain a references section where the resources that you have used during the project are listed.

Idea

XML (Extensible Markup Language) is one of the most famous formats for storing and sharing information among different devices. Some text editors such as Sublime Text are able to parse such files and do some basic operations. In this project, you will work on developing a GUI (Graphical User Interface) based program to parse and visualize an XML file.

In our system the XML file will represent users in a social network
Each user has id (unique), name, list of posts, list of followers.
Each post has text and list of topics.

- Sample of input file: https://drive.google.com/file/d/1_GeHIjYQZEAZNu6ZxdXEotPnHhiKnaWn/view?usp=sharing

Requirements - Level 1

- Build a **desktop application** that can process XML files.
- The application should work on two modes:
    1. **Command line mode**

        User can run the application through the command line using the command xml_editor followed by the action, input and output files locations (if needed)

        xml_editor verify -i input_file.xml

        xml_editor compress -i input_file.xml -o output_file.xml
    2. **GUI mode**

        The GUI should allow the user to provide the input file either through a browse button or a text file in the application

        The GUI should allow the user to see the operations outputs in a read-only text field and also support saving the output file in a location determined by the user.

        The GUI should contain buttons for the user to do the required operations on the xml

Requirements - Level 1

- Building a **GUI** in which the user can specify the location of an input XML file.
- Checking the XML consistency: The input XML may have inconsistencies like missing any of the closing and opening tags or not matching tags.
- The program should be able to detect and visually show any errors in consistency.
- The program will also be able to solve the errors.

Requirements - Level 1

```
<users>
    <user>
        <id>1
        <name>user1</name>
        <posts>
            <post>
                Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
                tempor incididunt ut labore et dolore magna aliqua.
            </post>
            <post>
                Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi
                ut aliquip ex ea commodo consequat.
            </post>
        </posts>
        <followers>
            <follower>
                <name>2</id>
            </follower>
            <follower>
                <id>4</id>
            </follower>
    </user>
</users>
```

Requirements - Level 1

- Formatting (Prettifying) the XML: the XML file should be well formatted by keeping the indentation for each level.

```
<users>
<user>
<id>1</id>
<name>user1</name>
<posts>
<post>
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
    eiusmod tempor incididunt ut labore et dolore magna aliqua.
</post>
<post>
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
    nisi ut aliquip ex ea commodo consequat.
</post>
</posts>
<followers>
<follower>
<id>2</id>
</follower>
<follower>
<id>4</id>
</follower>
</followers>
</user>
</users>
```

```
<users>
    <user>
        <id>1</id>
        <name>user1</name>
        <posts>
            <post>
            Lorem ipsum dolor sit amet, consectetur adipiscing elit,
                sed do eiusmod tempor incididunt ut labore et dolore
                magna aliqua.
            </post>
            <post>
            Ut enim ad minim veniam, quis nostrud exercitation ullamco
                laboris nisi ut aliquip ex ea commodo consequat.
            </post>
        </posts>
        <followers>
            <follower>
                <id>2</id>
            </follower>
            <follower>
                <id>4</id>
            </follower>
        </followers>
    </user>
</users>
```

Requirements - Level 1

- Converting XML to JSON: JSON (Javascript Object Notation) is another format that is used to represent data. It's helpful to convert the XML into JSON, especially when using javascript as there's tons of libraries and tools that use json notation.

```
{
    "users": [
        "user": {
            "id": "1",
            "name": "user1",
            "posts": [
                    "Lorem ipsum dolor sit amet, consectetur
                        adipiscing elit, sed do eiusmod tempor
                        incididunt ut labore et dolore magna aliqua.",
                    "Ut enim ad minim veniam, quis nostrud
                        exercitation ullamco laboris nisi ut aliquip
                        ex ea commodo consequat."
                ],
            "followers": [
                    {
                        "id": "2"
                    },
                    {
                        "id": "4"
                    }
                ]
            }
        ]
}
```

Requirements - Level 1

- Minifying the XML file: Since spaces and newlines (**\n**) are actually characters that can increase the size of an XML document. This feature should aim at decreasing the size of an XML file (compressing it) by deleting the white spaces and indentations.
- Compressing the data in the XML/JSON file: You should come-up with a way to reduce the size of the file using a data compression technique. You can invent your own ad-hoc method for such compression. On the other hand, you can check how JSON works and try to distill ideas from it. Finally, you can use a data compression technique such as byte pair encoding (https://en.wikipedia.org/wiki/Byte_pair_encoding).
  The smaller the output file is, the more efficient your algorithm is.
- Decompressing the compressed data to return to its original format.

Requirements - Level 1

Operation	Command	Expected Output
Checking the XML consistency	xml_editor verify -i input_file.xml -f -o output_file.xml	Output valid in the console if the xml is consistent and invalid otherwise In case of invalid xml file: write how many errors and in which lines If the user used the -f option and -o output_file.xml The application should fix the xml file errors if any and output the correct version in output_file.xml

The options in orange color are optional

Requirements - Level 1

Operation	Command	Expected Output
Formatting (Prettifying) the XML	xml_editor format -i input_file.xml -o output_file.xml	Format the input_file.xml and save the formatted version in output_file.xml
Converting XML to JSON	xml_editor json -i input_file.xml -o output_file.json	Convert the input_file.xml to json and save it in output_file.json
Minifying the XML file	xml_editor mini -i input_file.xml -o output_file.xml	Minifies the input_file.xml and save it in output_file.xml

Requirements - Level 1

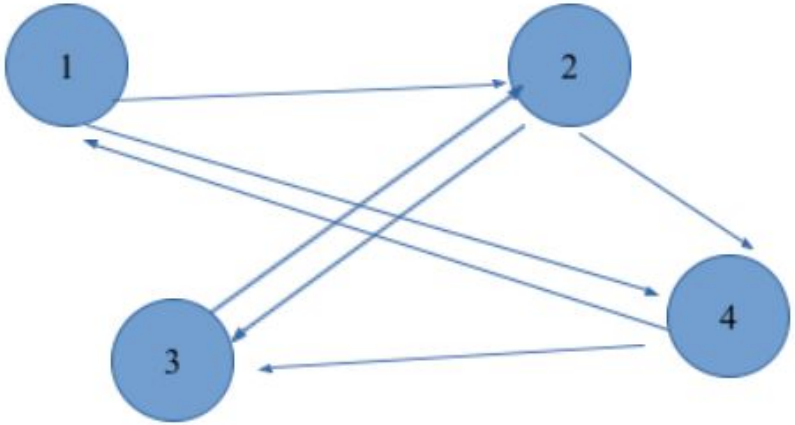
Operation	Command	Expected Output
Compressing XML file	xml_editor compress -i input_file.xml -o output_file.comp	Compress the input_file.xml and save the compressed version in output_file.comp
Decompressing XML file	xml_editor decompress -i input_file.comp -o output_file.xml	Decompress the input_file.comp and save the decompressed version in output_file.xml

Requirements - Level 2

- representing the users data using the graph data structure: the XML file will represent the users data in a social network (their posts, followers, ...etc).
  The user data is his id (unique across the network), name, list of his posts and followers.
  So you should represent the relation between the followers using the graph data structure as it will be very helpful for the network analysis.
  If the input file was like this (the dots mean that there are additional tags inside the user tag) :
  Then you should build a graph relation between the user that looks like the graph beneath.

Operation	Command	Expected Output
Representing the xml in a graph	xml_editor draw -i input_file.xml -o output_file.jpg	Draws the input data as a graph in output_file.jpg

Requirements - Level 2




```
<users>
    <user>
        <id>1</id>
        ...
        <followers>
            <follower>
                <id>2</id>
            </follower>
            <follower>
                <id>4</id>
            </follower>
        </followers>
    </user>
    <user>
        <id>2</id>
        ...
        <followers>
            <follower>
                <id>3</id>
            </follower>
            <follower>
                <id>4</id>
            </follower>
        </followers>
    </user>
    <user>
        <id>3</id>
        ...
        <followers>
            <follower>
                <id>2</id>
            </follower>
        </followers>
    </user>
    <user>
        <id>4</id>
        ...
        <followers>
            <follower>
                <id>1</id>
            </follower>
            <follower>
                <id>3</id>
            </follower>
        </followers>
    </user>
</users>
```

Requirements - Level 2

- Network analysis: by representing the network using the graph data structure, we can extract some important data:
  - who is the most influencer user (has the most followers)
  - who is the most active user (connected to lots of users)
  - the mutual followers between n users
  - for each user, suggest a list of users to follow (the followers of his followers)
- Post search: given a specific word or topic, get the posts where this word or topic was mentioned.

Operation	Command	Expected Output
Network Analysis	xml_editor most_active -i input_file.xml	Writes the most active user name and id
	xml_editor most_influencer -i input_file.xml	Writes the most influencer user name and id
	xml_editor mutual -i input_file.xml -ids 1,2,3	Writes a list of the mutual users between users with id 1, id 2 and id 3

Requirements - Level 2

- Network analysis: by representing the network using the graph data structure, we can extract some important data:
  - who is the most influencer user (has the most followers)
  - who is the most active user (connected to lots of users)
  - the mutual followers between n users
  - for each user, suggest a list of users to follow (the followers of his followers)
- Post search: given a specific word or topic, get the posts where this word or topic was mentioned.

Operation	Command	Expected Output
Network Analysis	xml_editor suggest -i input_file.xml -id 1	Writes a list of suggested users for user with id 1
Post search	xml_editor search -w word -i input_file.xml	Writes a list of posts where the given word was mentioned
	xml_editor search -t topic -i input_file.xml	Writes a list of posts where the given topic was mentioned

Requirements - Level 2

- Graph visualization: you're free to use any tool or 3rd party library to help you visualize the graph of the network showing how the user is connected to each other.

Submission Requirement

- The program should be working properly with no crashes
- The program should have a simple GUI window containing a set of buttons that are used to choose the operation that needs to be done on the XML file.
- The GUI should allow the user to choose the XML file that will be used or enter the xml manually in a text area.
- The GUI should allow viewing the input XML and the output of each operation as much as possible (If you face problems with large files then you can view only the first lines of the file).
- The GUI should allow the user to save the results of the operations to a new file.
- You have to fully deliver Level 1 requirements to get Level 2 requirements degrees
- Deadline : 30 December 2025

Why this project

In this project, you will learn how to understand and parse XML and json files. Additionally, you will work on designing a GUI (Graphical User Interface) to visualize XML and json files. You should also work on designing the program such that it makes use of optimal data structures for the implemented features.