ALGORITHMICS - Bachelor of Software Engineering, 2024

LAB GUIDE. SESSION 5

GOALS:

- **Problem solving with dynamic programming**

1. Pattern matching with texts

Given a text and a pattern, we will say that the pattern matches the text if the pattern can represent the entire text using its characters.

Considerations about the **text**:

- The text is any string made up of letters **[a-zA-Z]**.

Considerations about the **pattern**:

- The pattern is any string made up of letters **[a-zA-Z]** and the characters ***** or **?**.
- ***** refers to any sequence of characters (including the empty character).
- **?** refers to any character (including the empty character).

To solve this problem with dynamic programming we can create a table in which the rows are the letters of the text, and the columns are the letters of the pattern to be verified. Table values will be true (T) when the pattern up to a certain column matches the text up to a certain row. It will be false (F) if the pattern does not match the corresponding text.

We could talk about different cases:

1. **Base case**: if the pattern is empty, it will only match the text if the text is also empty (see red letters in the examples).
2. **When both letters match**: when in the row and in the column, we have the same letter then only one thing can happen for the pattern to match the text:
    - If there was a match in the previous position of each string (previous row and previous column), they will now continue to match after adding the new letter (see example 1).
3. **When both letters do not match**: when we have different letters in the row and column then the pattern does not match the text in that specific position (see example 2).
4. **When we find a ?**: when in the column we find a ? two things can happen to make the pattern match the text:
    - If there was a match in the previous position of each string (previous row and previous column), they will now continue to match after adding the letter instead of the ?, since any letter will work (see example 3).
    - If there was a match in the previous position of the pattern (same row and previous column), now they will continue to match if we ignore the ?, since we can act as if it does not exist (see example 3).
5. **When we find an ***: when in the column we find a * three things can happen to make the pattern match the text:

- If there was a match in the previous position of each string (previous row and previous column), they will now continue to match after adding the letter instead of the *, since any letter will work (see example 4).
- If there was a match in the previous position of the pattern (same row and previous column), now they will continue to match if we ignore the *, since we can act as if it does not exist (see example 4).
- If there was a match in the previous position of the text (previous row and same column), now they will continue to match, since an * can represent any number of letters in the text (see example 4).

Example 1:

- Text: **casa**
- Pattern: **casa**
- Matches: **yes**

		c	a	s	a
	T	F	F	F	F
c	F	**T**	F	F	F
a	F	F	**T**	F	F
s	F	F	F	**T**	F
a	F	F	F	F	**T**

Example 2:

- Text: **casa**
- Pattern: **cosa**
- Matches: **no**

		c	o	s	a
	T	F	F	F	F
c	F	**T**	F	F	F
a	F	F	F	F	F
s	F	F	F	F	F
a	F	F	F	F	F

Example 3:

- Text: **casa**
- Pattern: **ca?a**
- Matches: **yes**

		c	a	?	a
	T	F	F	F	F
c	F	**T**	F	F	F
a	F	F	**T**	**T**	F
s	F	F	F	**T**	F
a	F	F	F	F	**T**

Example 4:

- Text: **casa**
- Pattern: ****a**
- Matches: **yes**

		*	*	a
	T	F	F	F
c	F	**T**	**T**	F
a	F	**T**	**T**	**T**
s	F	**T**	**T**	F
a	F	**T**	**T**	**T**

YOU ARE REQUESTED TO:

Design and implement an algorithm using dynamic programming to solve this problem optimally.

a) Implement the algorithm in Java in such a way that it calculates for a specific text string whether the different patterns passed in the text file match the string.
b) What is the time complexity of the algorithm? Do you think the design could be changed to improve complexity?

The entry data will be found on a text file with the specified format:

```
casa

casa true

cosa false

ca?a true

**a true
```

Representing the following:

- First line: **text that we want to check**
- Second and next lines: **information about patterns**
    - First part: the pattern
    - Second part: whether the pattern should coincide with the text or not.

The program must show the results obtained for the given file, including the table showing partial solutions before obtaining the final solution for each test case.

2. Work to be done

- An `algstudent.s5` **package** in your course project. The content of the package should be the Java files used and created during this session.
- A `session5.pdf` **document** using the course template (the document should be included in the same package as the code files). You should create one activity each time you find a "YOU ARE REQUESTED TO" instruction.

Deadline: The deadline is one day before the next lab session of your group.