Algorithm Check if *text* matches *pattern* with support of wildcard characters $*$ and ?

Ensure: Zero Based Indexing for the Matrix, One Based Index for the Strings

1: **function** $Wildcard\_Pattern\_Matching(text, pattern)$
▷ $match[i][j]$ is $True$ if the first i characters of *text* matches the first j characters of *pattern*

2: $match[i][j] \leftarrow false \quad \forall i \quad \forall j$ ▷ Initialise the DP Matrix

3: $match[0][0] \leftarrow true$ ▷ Both strings are empty

4: **for** $i = 1$ **to** $text.length$ **do** ▷ The *pattern* is empty
5: $match[i][0] \leftarrow false$

6: $j \leftarrow 1$
7: **while** $j \leq pattern.length$ **and** $pattern[j]$ **is** $*$ **do** ▷ Text is empty
8: $match[0][j] \leftarrow true$ ▷ It can only match with $***\ldots$
9: $j \leftarrow j + 1$

10: **for** $i = 1$ **to** $text.length$ **do**
11: **for** $j = 1$ **to** $pattern.length$ **do**
12: $last\_t \leftarrow text[i]$
13: $last\_p \leftarrow pattern[j]$
14: **if** $last\_p$ **is** ? **then** ▷ Wildcard
15: $match[i][j] \leftarrow match[i-1][j-1]$
16: **else if** $last\_p$ **is** $*$ **then** ▷ Wildcard
17: $match[i][j] \leftarrow match[i][j-1]$ **or** $match[i-1][j]$
18: **else if** $last\_p == last\_t$ **then** ▷ Not a wildcard
19: $match[i][j] \leftarrow match[i-1][j-1]$
20:
21: **return** $match[text.len][pattern.len]$

Miscellaneous

State Transition

Suppose that the *text* has length i and *pattern* has length j

1. If the last character of the *pattern* is ?, it means, that we necessarily have to match it with the last character of *text*. Hence, now we need to find whether the first $i-1$ characters of *text* matches with the first $j-1$ characters of *pattern*. Hence, we recur for $match[i-1][j-1]$

2. If the last character of the *pattern* is $*$, then we have 2 choices, either match the wildcard to the last character of the text or skip it. If we decide to skip it, it means that the wildcard is matched with the empty sequence and hence we need to check if we can match the entire *text* of length i to the *pattern* of length $j-1$. On the other hand, if we decide to match it with the last character, we can still use the wildcard to match other characters. Hence, we need to check if we can match the first $i-1$ characters of *text* to the first j characters of *pattern*. We take the best of the two outcomes. Hence, we recur for $match[i][j-1]$ **or** $match[i-1][j]$

3. If the last character of *pattern* is not a wildcard, we have to ensure that it matches with the last character of *text*. If it does, we can recur for $match[i-1][j-1]$, else we set it to $false$.

Base Case

1. If both the *pattern* and *text* has length 0, then they match. Hence, $match[0][0] = true$

2. If the *pattern* has length 0, it cannot be matched to any *text* of positive length. Hence, $match[i][0] = false$ $\forall i > 0$

3. If the *text* has length 0, it can only be matched with a *pattern* of the type $***\ldots**$. Hence, we iterate the *text* and keep setting $match[0][j] = true$ until we hit a character which is not $*$