GTU Department of Computer Engineering
CSE 484 / 654 - Fall 2022
Homework #1 Report

Hasan Mutlu
1801042673

Preparing Input Files

All of the texts in the books are copied to different text files. After that, they are parsed according to the sentences and these sentences are printed line by line into output files with 400 lines at most.

186 files are created at most but only 100 of them are included in the homework file in order to reduce file size.

Algorithm Definitions and Program Flow

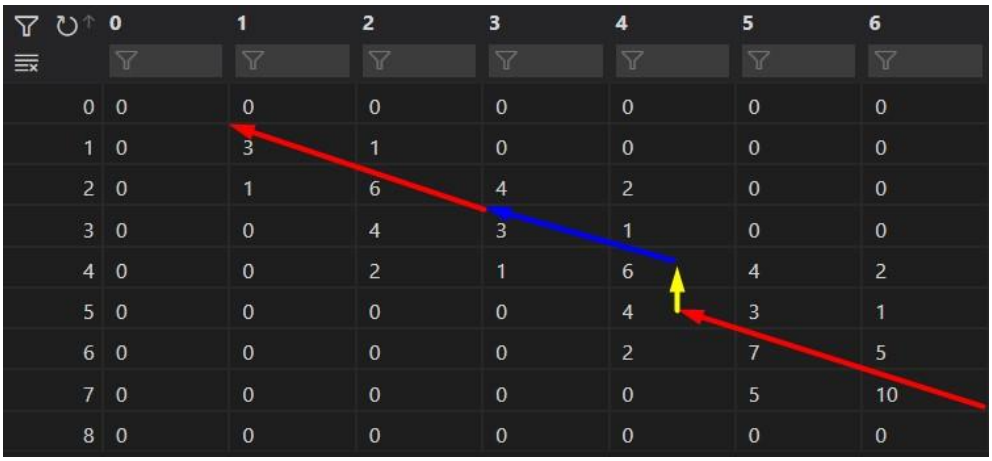
Smith Waterman Algorithm:

```
# Smith-Waterman algorithm part. Matrix is filled.

for i in range (1,len(lines1) + 1):
    for j in range (1,len(lines2) + 1):
        if(lines1[i-1] == lines2[j-1]):
            match = MATCH
        else:
            match = -MATCH
        val = maxOf(matrix[i-1][j] - GAP, matrix[i][j-1] - GAP, matrix[i-1][j-1] + match, 0)
        matrix[i][j] = val
```

After the lines in the input files are read into two arrays, named lines1 and lines2, the matrix is filled using the Smith-Waterman algorithm. MATCH value is 3 and GAP value is 2 by default. I chose these values because examples in the Wikipedia were this way.

After that, the starting index candidates of every path in the matrix are found and saved into arrays. A matrix cell is probably the starting point of a path if its value is higher than the cells surrounding it.



	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	3	1	0	0	0	0
2	0	1	6	4	2	0	0
3	0	0	4	3	1	0	0
4	0	0	2	1	6	4	2
5	0	0	0	0	4	3	1
6	0	0	0	0	2	7	5
7	0	0	0	0	0	5	10
8	0	0	0	0	0	0	0

The reason I used the words “candidate” and “probably” is that not every cell with higher value than the cells surrounding it is the starting point of a path. For example, in the matrix above, the cell at (2,2) indexes have higher value than its surroundings but it is because there is a gap before it. It is not a starting point and in fact is a passing point of a longer path.

In the next part of the program, traceback procedure is applied and indexes like the example above are removed from the candidates list when they are encountered during another path's traceback.

After getting rid of the false candidates, we are left with all of the real starting indexes. Traceback procedure is applied once again starting from these indexes and the output is prepared and printed.

Traceback Procedure:

During the traceback procedure, maxTraceBack(M,a,b,c) function checks the difference between the current cell [i,j] and the three cells coming before it. ([i-1][j] - [i][j-1] - [i-1][j-1]) If their differences are equal to corresponding MATCH or GAP values, their condition is true. The function checks all of these conditions and decides which matrix cell to go to nextly.

Again, in the example above, at the cell at indexes (5,4), its difference with the cell above is equal to GAP value. Also, its difference with the diagonal cell is equal to GAP value. When more than one conditions are true like this, the function chooses the cell with the biggest value out of them in order to find the longest path. In this case, the cell above is chosen.

Output Format Definition

```
Document 1: booktexts/tarih1Out1.txt
Document 2: booktexts/psikoloji1Out2.txt
----------------------
1. Alignment:
Path length: 7
88 - 16: Ali topu at.
89 - 17: Veli topu tut.
90 - 18: ---------GAP2--------
91 - 19: Ayşe ip atla.
91 - 20: ---------GAP1---------
92 - 21: Çocuklar oyun oynayın.
93 - 22: Kitap okuyun çocuklar.
----------------------
2. Alignment:
Path length: 1
323 - 210: 1.
----------------------
3. Alignment:
Path length: 1
323 - 231: 1.
----------------------
```

In the output, firstly, the input files are displayed. After that, every alignment that are found are printed one by one.

In the alignments, path length is printed. Matching sentences are printed with their line numbers in the text files. The gaps are printed as well. There are two kinds of gaps.

➔ GAP1 is caused when there is an excess line in one of the documents. Example:

abcde – abde => ab-de

➔ GAP2 is caused when there is an unmatching line in both of the documents. Example:

abcde – abzde => ab-de

Small Text Matching Examples:

Input files:

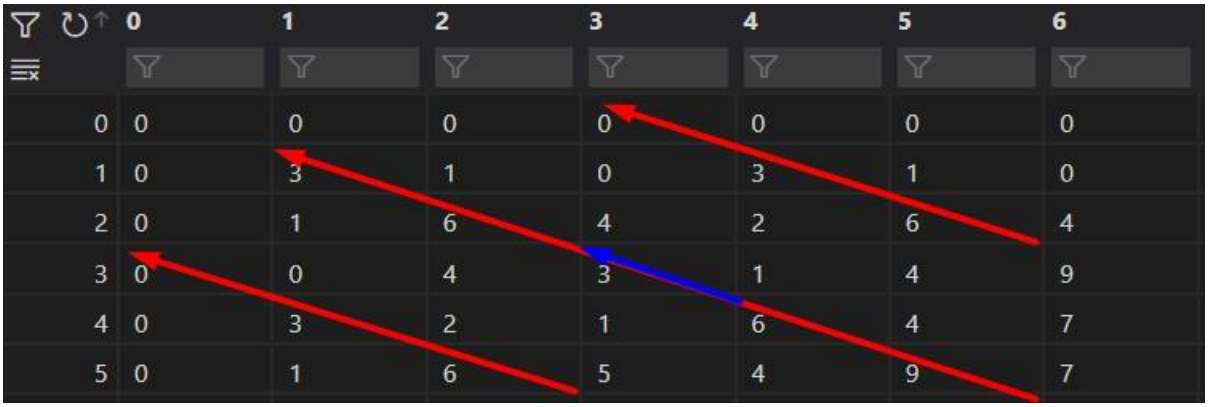
src > deneme2.txt

```
1  Ali topu at.
2  Veli topu tut.
3  Veli topu tut.
4  Ali topu at.
5  Veli topu tut.
6  Selami top oyna.
```

src > deneme1.txt

```
1  Ali topu at.
2  Veli topu tut.
3  Selami top oyna.
4  Ali topu at.
5  Veli topu tut.
```

Matrix result:



	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	3	1	0	3	1	0
2	0	1	6	4	2	6	4
3	0	0	4	3	1	4	9
4	0	3	2	1	6	4	7
5	0	1	6	5	4	9	7

GAP2 is displayed with blue arrow.

Output:

```
Document 1: deneme1.txt
Document 2: deneme2.txt
-----------------------
1. Alignment:
1 - 4: Ali topu at.
2 - 5: Veli topu tut.
3 - 6: Selami top oyna.
-----------------------
2. Alignment:
4 - 1: Ali topu at.
5 - 2: Veli topu tut.
-----------------------
3. Alignment:
1 - 1: Ali topu at.
2 - 2: Veli topu tut.
3 - 3: ---------GAP2--------
4 - 4: Ali topu at.
5 - 5: Veli topu tut.
```

Input files:

src > deneme1.txt

```
1  Ali topu at.
2  Veli topu tut.
3  Selami top oyna.
4  Ayşe ip atla.
5  Ela un ele.
6  Çocuklar oyun oynayın.
7  Kitap okuyun çocuklar.
```

src > deneme2.txt

```
1  Ali topu at.
2  Veli topu tut.
3  Süt için çocuklar.
4  Ayşe ip atla.
5  Çocuklar oyun oynayın.
6  Kitap okuyun çocuklar.
```

Matrix result:

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	3	1	0	0	0	0
2	0	1	6	4	2	0	0
3	0	0	4	3	1	0	0
4	0	0	2	1	6	4	2
5	0	0	0	0	4	3	1
6	0	0	0	0	2	7	5
7	0	0	0	0	0	5	10
8	0	0	0	0	0	0	0

GAP2 is displayed with blue arrow and GAP1 is displayed with yellow arrow.

Output:

```
Document 1: deneme1.txt
Document 2: deneme2.txt
----------------------
1. Alignment:
1 - 1: Ali topu at.
2 - 2: Veli topu tut.
3 - 3: ---------GAP2--------
4 - 4: Ayşe ip atla.
5 - 4: ---------GAP1---------
6 - 5: Çocuklar oyun oynayın.
7 - 6: Kitap okuyun çocuklar.
```

Insert Same Line of Text Example:

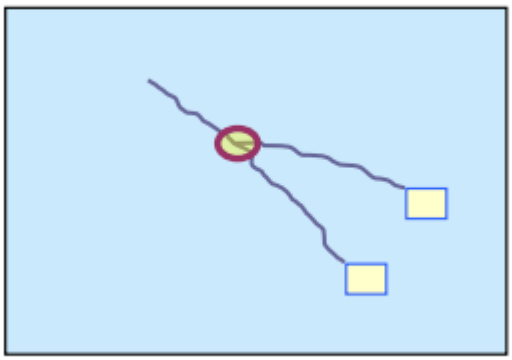
Example 1:

```
Document 1: booktexts/tarih1Out1.txt
Document 2: booktexts/psikoloji1Out2.txt
-----------------------
1. Alignment:
Path length: 7
88 - 16: Ali topu at.
89 - 17: Veli topu tut.
90 - 18: ---------GAP2--------
91 - 19: Ayşe ip atla.
91 - 20: ---------GAP1---------
92 - 21: Çocuklar oyun oynayın.
93 - 22: Kitap okuyun çocuklar.
-----------------------
2. Alignment:
Path length: 1
323 - 210: 1.
-----------------------
3. Alignment:
Path length: 1
323 - 231: 1.
-----------------------
4. Alignment:
Path length: 1
325 - 15: 2.
-----------------------
5. Alignment:
Path length: 1
327 - 244: 3.
```

The lines in the second example above are inserted into the tarih1Out1.txt and psikoloji1Out2.txt files. These matching lines are found on the output. It also found some other matching lines, which are sentences only made of a number, in these two documents.

Example 2:

The first document of every book have İstiklal Marşı and Gençliğe Hitabe in them. So I used one as an example. edebiyat1Out1.txt and edebiyat2Out1.txt files have even longer matching sequence of lines. Since its output is too long, It is attached as an external file named edebiyat1_1-edebiyat2_1.



In the output of edebiyat1_1-edebiyat2_1.txt, a condition like above is happened as well. There were two distinct paths that would eventually come together at a point. Both of these paths are included in the output.

Example 3:

Same input file is used in the program. felsefe1Out1.txt is compared with itself. As expected, all lines are matched. Its output is attached externally as felsefe1_1-felsefe1_1.txt