CSE 4065 – Computational Genomics
Programming Assignment #2

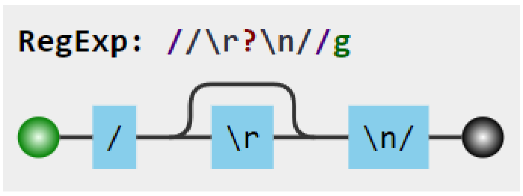
Project Report

Group Members:
Cem Güleç: 150117828
Büşra Gökmen: 150116027
Ömer Faruk Çakı: 150117821

Introduction:
In this assignment it is aimed to implement a dynamic pairwise sequence alignment algorithm. Furthermore, based on this algorithm, metrics such as gap, mismatch penalties, match scores are discussed. Finally, it is tried to find optimal alignment between given input pairs using the optimal scores obtained. (Bonus part, where the penalty needs to be applied is also done!)

Parsing the input file:
Parsed input file is stored inside ***firstLine*** and ***secondLine*** variables. In order to parse it into two lines a regular expression is used.

```
const [firstLine, secondLine] = fs
       .readFileSync(path.join(__dirname, INPUT_FILE))
       .toString()
       .split(/\r?\n/)
       .map(String);
```



<u>Classes</u>:
There are two main classes used which are ***_Node*** and ***Matrix*** classes.

```
class _Node {
  cost: number;
  x: number;
  y: number;
  from: _Node | null;

  constructor(cost = 0, x = 0, y = 0) {
    this.cost = cost;
    this.x = x;
    this.y = y;
    this.from = null;
  }
}
```

Node class is used in order to create a node which will be then used in the grid structure. Each node has several attributes.

***<u>cost</u>*:** Is a number which holds path cost until this node.
<u>x</u>: Is a number which holds x coordinate on the grid.
<u>y</u>: Is a number which holds y coordinate on the grid.
<u>from</u>: Is either a node instance or null value (may be a leaf node) which holds the node on the path which pointed to this node.
***<u>constructor</u>*:** Is the initialization function of a node.

```
class Matrix {
  private table: Array<Array<_Node>> | undefined;
  private first: string;
  private second: string;
  private path: Array<_Node> | undefined;
  private alignedSequences: { first: string; second: string } | undefined;

  constructor(first: string, second: string) {
    this.first = first;
    this.second = second;

    this.createEmptyTable(this.second.length + 1, this.first.length + 1);
    this.calculateScores();
    this.buildPath();
    this.buildSequences();
  }
```

`Matrix class is the main class we used which holds different variables and functions to represent 2D grid structure in our program.`

`table``: Is an arraylist created from _Node instances to represent grid representation.`
`first``: Is the first sequence string.`
`second``: Is the second sequence string.`
`path``: Is an array structure which holds _Node instances to save path from start to sink.`
`alignedSequences``: Is a set of strings which hold final alignment results constructed from the solution path.`
`constructor``: Takes first and second variables as parameters to initialize the grid structure.`

Methodology:

Matrix class is the main class which is used to represent the 2D grid, in each cell there is a _Node instance. Each node is linked with it's parent node, the node which leads to the current node.

In the beginning of the program we create a new Matrix class instance using our two input sequences. After that constructor method calls the initialization methods and we create an empty grid corresponding the **length(sequence1) * length(sequence2).** The starting node is initially assigned, and all other nodes are created one by one and their costs are calculated and assigned. Costs are calculated as follows: first of all we get the all costs of the valid neighbour nodes which are left of the current node, top of the current node and top left of the current node if they exist. Then if it's a match we add **2** to the cost, if it's a mismatch or gap opening we subtract **1** from the costs and lastly if it's a gap extension we subtract **0.5 (bonus part)**.

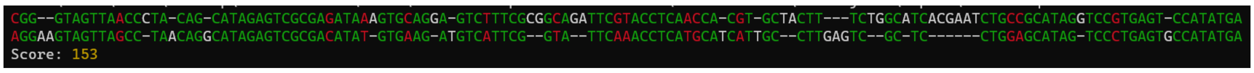
After costs for each node are calculated we calculate the sink node at the latest. After it's completed the cost obtained is our score which we are looking for. In order to obtain the path, we follow all the linked nodes from sink until the starting node and we construct our aligned sequences.

Results:

We have printed colored results. Greens represent matches while reds represent mismatches. First line of the output is the aligned first sequence while the second line is the aligned second sequence.

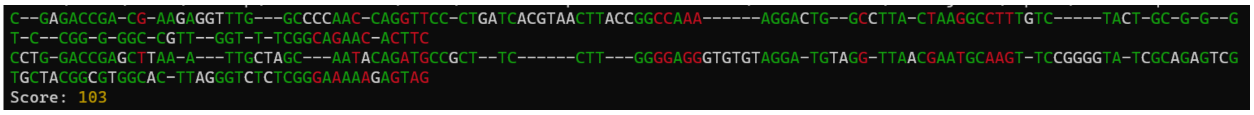
Scores for the tests and their outputs are below:

Test 1: 153



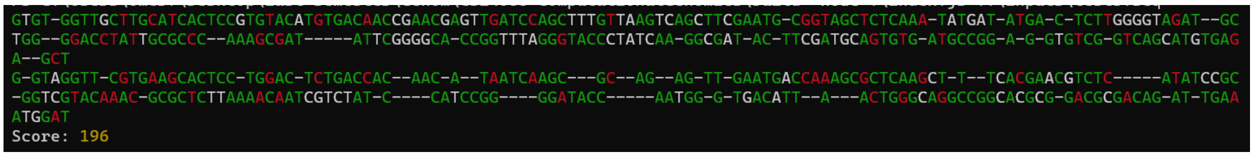
```
CGG--GTAGTTAACCCTA-CAG-CATAGAGTCGCGAGATAAAGTGCAGGA-GTCTTTCGCGGCAGATTCGTACCTCAACCA-CGT-GCTACTT---TCTGGCATCACGAATCTGCCGCATAGGTCCGTGAGT-CCATATGA
AGGAAGTAGTTAGCC-TAACAGGCATAGAGTCGCGACATAT-GTGAAG-ATGTCATTCG--GTA--TTCAAACCTCATGCATCATTGC--CTTGAGTC--GC-TC------CTGGAGCATAG-TCCCTGAGTGCCATATGA
Score: 153
```

Test 2: 103 *(output lines wrapped)*



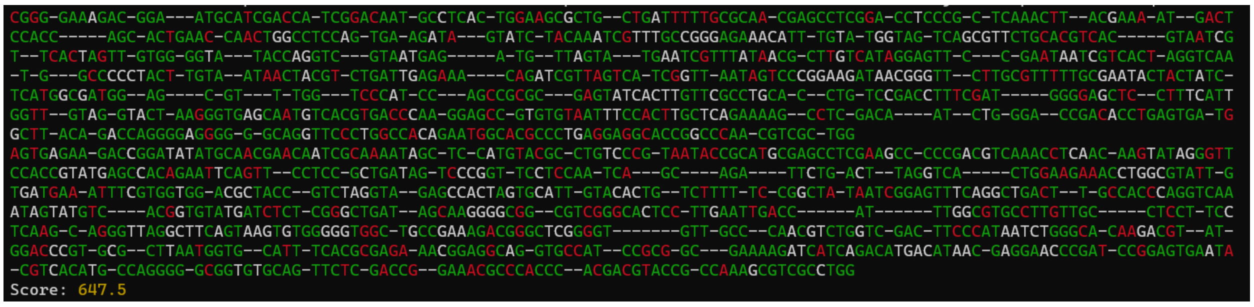
```
C--GAGACCGA-CG-AAGAGGTTTG---GCCCCAAC-CAGGTTCC-CTGATCACGTAACTTACCGGCCAAA------AGGACTG--GCCTTA-CTAAGGCCTTTGTC-----TACT-GC-G-G--G
T-C--CGG-G-GGC-CGTT--GGT-T-TCGGCAGAAC-ACTTC
CCTG-GACCGAGCTTAA-A---TTGCTAGC---AATACAGATGCCGCT--TC------CTT---GGGGAGGGTGTGTAGGA-TGTAGG-TTAACGAATGCAAGT-TCCGGGGTA-TCGCAGAGTCG
TGCTACGGCGTGGCAC-TTAGGGTCTCTCGGGAAAAAGAGTAG
Score: 103
```

Test 3: 196 *(output lines wrapped)*



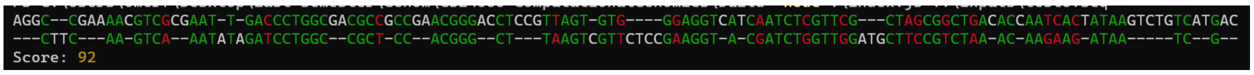
```
GTGT-GGTTGCTTGCATCACTCCGTGTACATGTGACAACCGAACGAGTTGATCCAGCTTTGTTAAGTCAGCTTCGAATG-CGGTAGCTCTCAAA-TATGAT-ATGA-C-TCTTGGGGTAGAT--GC
TGG--GGACCTATTGCGCCC--AAAGCGAT-----ATTCGGGGCA-CCGGTTTAGGGTACCCTATCAA-GGCGAT-AC-TTCGATGCAGTGTG-ATGCCGG-A-G-GTGTCG-GTCAGCATGTGAG
A--GCT
G-GTAGGTT-CGTGAAGCACTCC-TGGAC-TCTGACCAC--AAC-A--TAATCAAGC---GC--AG--AG-TT-GAATGACCAAAGCGCTCAAGCT-T--TCACGAACGTCTC-----ATATCCGC
-GGTCGTACAAAC-GCGCTCTTAAAACAATCGTCTAT-C----CATCCGG----GGATACC-----AATGG-G-TGACATT--A---ACTGGGCAGGCCGGCACGCG-GACGCGACAG-AT-TGAA
ATGGAT
Score: 196
```

Test 4: 647.5 *(output lines wrapped)*



```
CGGG-GAAAGAC-GGA---ATGCATCGACCA-TCGGACAAT-GCCTCAC-TGGAAGCGCTG--CTGATTTTTGCGCAA-CGAGCCTCGGA-CCTCCCG-C-TCAAACTT--ACGAAA-AT--GACT
CCACC-----AGC-ACTGAAC-CAACTGGCCTCCAG-TGA-AGATA---GTATC-TACAAATCGTTTGCCGGGAGAAACATT-TGTA-TGGTAG-TCAGCGTTCTGCACGTCAC-----GTAATCG
T--TCACTAGTT-GTGG-GGTA---TACCAGGTC---GTAATGAG-----A-TG--TTAGTA---TGAATCGTTTATAACG-CTTGTCATAGGAGTT-C---C-GAATAATCGTCACT-AGGTCAA
-T-G---GCCCCCTACT-TGTA--ATAACTACGT-CTGATTGAGAAA----CAGATCGTTAGTCA-TCGGTT-AATAGTCCCGGAAGATAACGGGTT--CTTGCGTTTTTGCGAATACTACTATC-
TCATGGCGATGG--AG----C-GT---T-TGG---TCCCAT-CC---AGCCGCGC---GAGTATCACTTGTTCGCCTGCA-C--CTG-TCCGACCTTTCGAT-----GGGGAGCTC--CTTTCATT
GGTT--GTAG-GTACT-AAGGGTGAGCAATGTCACGTGACCCAA-GGAGCC-GTGTGTAATTTCCACTTGCTCAGAAAAG--CCTC-GACA----AT--CTG-GGA--CCGACACCTGAGTGA-TG
GCTT-ACA-GACCAGGGGAGGGG-G-GCAGGTTCCCTGGCCACAGAATGGCACGCCCTGAGGAGGCACCGGCCCAA-CGTCGC-TGG
AGTGAGAA-GACCGGATATATGCAACGAACAATCGCAAAATAGC-TC-CATGTACGC-CTGTCCCG-TAATACCGCATGCGAGCCTCGAAGCC-CCCGACGTCAAACCTCAAC-AAGTATAGGGTT
CCACCGTATGAGCCACAGAATTCAGTT--CCTCC-GCTGATAG-TCCCGGT-TCCTCCAA-TCA---GC----AGA----TTCTG-ACT--TAGGTCA-----CTGGAAGAAACCTGGCGTATT-G
TGATGAA-ATTTCGTGGTGG-ACGCTACC--GTCTAGGTA--GAGCCACTAGTGCATT-GTACACTG--TCTTTT-TC-CGGCTA-TAATCGGAGTTTCAGGCTGACT--T-GCCACCCAGGTCAA
ATAGTATGTC----ACGGTGTATGATCTCT-CGGGCTGAT--AGCAAGGGGCGG--CGTCGGGCACTCC-TTGAATTGACC------AT------TTGGCGTGCCTTGTTGC-----CTCCT-TCC
TCAAG-C-AGGGTTAGGCTTCAGTAAGTGTGGGGGTGGC-TGCCGAAAGACGGGCTCGGGGT-------GTT-GCC--CAACGTCTGGTC-GAC-TTCCCATAATCTGGGCA-CAAGACGT--AT-
GGACCGT-GCG--CTTAATGGTG--CATT-TCACGCGAGA-AACGGAGGCAG-GTGCCAT--CCGCG-GC---GAAAAGATCATCAGACATGACATAAC-GAGGAACCCGAT-CCGGAGTGAATA
-CGTCACATG-CCAGGGG-GCGGTGTGCAG-TTCTC-GACCG--GAAACGCCCACCC--ACGACGTACCG-CCAAAGCGTCGCCTGG
Score: 647.5
```

Test 5: 92



```
AGGC--CGAAAACGTCGCGAAT-T-GACCCTGGCGACGCCGCCGAACGGGACCTCCGTTAGT-GTG----GGAGGTCATCAATCTCGTTCG---CTAGCGGCTGACACCAATCACTATAAGTCTGTCATGAC
---CTTC---AA-GTCA--AATATAGATCCTGGC--CGCT-CC--ACGGG--CT---TAAGTCGTTCTCCGAAGGT-A-CGATCTGGTTGGATGCTTCCGTCTAA-AC-AAGAAG-ATAA-----TC--G--
Score: 92
```