

CSCI 451/551 HW 2

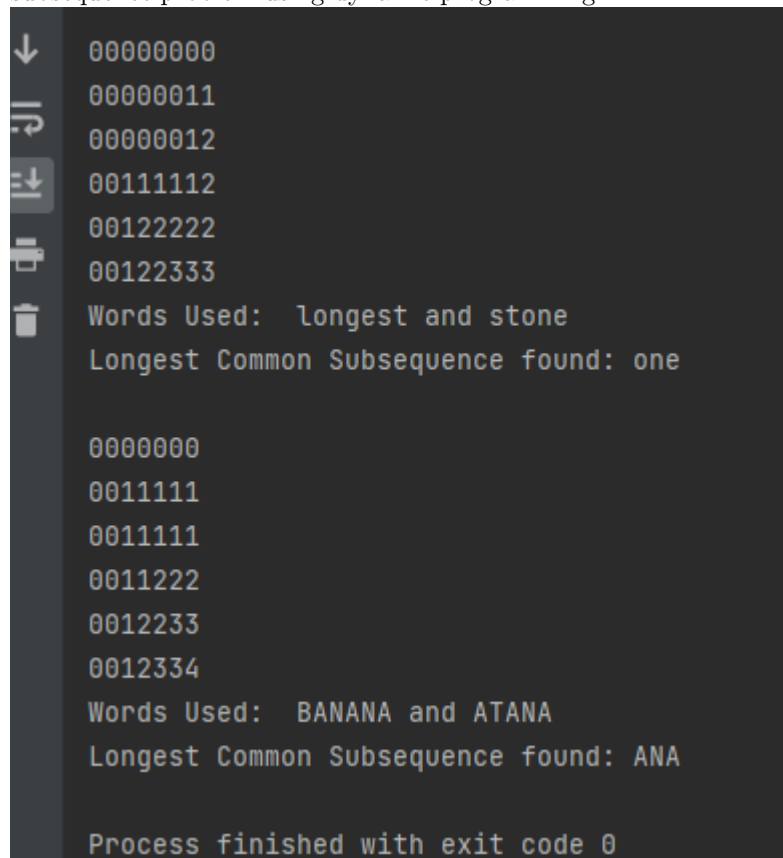
Kelly Joyce, Benjamin Haedt

November 30, 2023

1 Problem 1

Can you propose a dynamic programming solution to solve the longest common subsequence problem?

This shows the output of the program we created to solve the longest common subsequence problem using dynamic programming.

A terminal window with a dark background and light gray text. On the left side, there is a vertical toolbar with icons for back, forward, search, and other terminal functions. The output text is as follows:

```
00000000
00000011
00000012
00111112
00122222
00122333
Words Used:  longest and stone
Longest Common Subsequence found: one

00000000
00111111
00111111
00112222
00122333
00123334
Words Used:  BANANA and ATANA
Longest Common Subsequence found: ANA

Process finished with exit code 0
```

Answer: To create an solution to the longest common sub sequence problem utilizing dynamic programming. Here are the steps I took for solving the algorithm.

1. Fill the first row and first column with 0's, array size is array[x+1][y+1]
2. Attempt to match the characters by row from right to left.
3. If a match is found at array[x][y], then add array[x+1][y+1] to array[x][y] + 1
4. If match isn't found, take the largest value from either array[x+1][y] or array[x][y+1] and put it into the current slot, array[x][y]
5. Once end is reached, start second part of algorithm that reads the array backwards, from the end position last row, last column going left column by column until column 1 is reached, go up a row, and start at the end of the column on the right.
6. If array[x-1][y-1] is greater than array[x][y], take the character at position array[x][y] and add that to our sequence. Else, while array[x-1][y-1] \leq array[x][y], continue iterating through the 2D array.
7. If array[x-1][y-1] equals to 0, return the reverse of the sub-string to get the least common sub sequence.

Here is the solution in Java.

2 LCSS algorithm.java

```
public class Main {
    public static void main(String[] args) {
        String one = "longest";
        String two = "stone";
        lcs(one, two);
        one = "BANANA";
        two = "ATANA";
        lcs(one, two);
    }
    public static void lcs(String stringOne, String stringTwo) {
        if (stringOne.length() < stringTwo.length()) {
            String temp = stringOne;
            stringOne = stringTwo;
            stringTwo = temp;
        }
        int lengthS1 = stringOne.length();
        int lengthS2 = stringTwo.length();
        int[][] array;
        if (lengthS1 >= lengthS2) {
            ;
        } else {
```

```

        System.out.println("ERROR");
    }
    array = new int[lengthS2 + 1][lengthS1 + 1];
    for (int i = 1; i < lengthS2 + 1; i++) {
        for (int j = 1; j < lengthS1 + 1; j++) {
            if (stringOne.charAt(j - 1) == stringTwo.charAt(i - 1)) {
                array[i][j] = array[i - 1][j - 1] + 1;
            } else if (array[i - 1][j] > array[i][j - 1]) {
                array[i][j] = array[i - 1][j];
            } else {
                array[i][j] = array[i][j - 1];
            }
        }
    }
    String match = "";
    int lengthOfSubString = array[lengthS2][lengthS1];
    FOUND:
    for (int i = lengthS2; i > 1; i--) {
        for (int j = lengthS1; j > 1; j--) {
            if (array[i][j - 1] == array[i - 1][j] && array[i - 1][j]
                == array[i - 1][j - 1] && array[i - 1][j - 1] !=
                array[i][j]) {
                match += stringOne.charAt(j - 1);
                if (array[i - 1][j - 1] == 0) {
                    break FOUND;
                }
                break;
            }
        }
    }
    System.out.println("");
    for (int x = 0; x < lengthS2 + 1; x++) {
        for (int y = 0; y < lengthS1 + 1; y++) {
            System.out.print(array[x][y]);
        }
        System.out.println("");
    }
    String LCSFound = "";
    for (int i = 0; i < match.length(); i++) {
        LCSFound = match.charAt(i) + LCSFound;
    }
    System.out.println("Words Used: " + stringOne + " and " +
        stringTwo);
    System.out.println("Longest Common Subsequence found: " +
        LCSFound);
}
}

```

3 Problem 2

Given two sequences S and T (not necessarily the same length), let G, L, and H be the scores of an optimal global alignment, an optimal local alignment, and an optimal global alignment without counting the beginning space of S and end space of T, respectively.

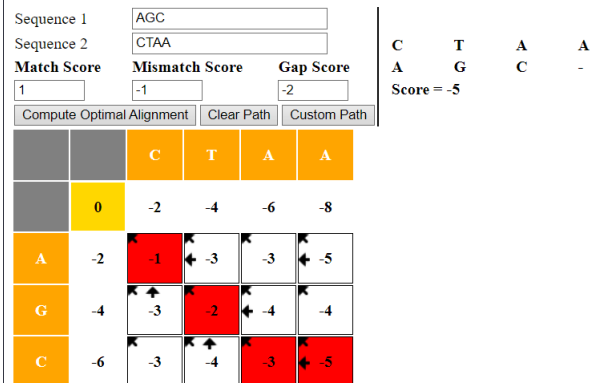
1. Give an example of S and T so that all three scores G, L, and H are different.

Answer: I used $S = AGC$, and $T = CTAA$. Running each through the local alignment, global alignment, and global alignment without counting the beginning space of S and end space of T. We were able to come up with different scores for each. With the optimal global alignment, we had a score of -5. With the optimal local alignment algorithm, we had a score of 1. Finally, with the optimal global alignment, not counting the beginning space of S and end space of T, we had a score of -4.

Global Alignment Algorithm Results:

Interactive demo for Needleman-Wunsch algorithm

The motivation behind this demo is that I had some difficulty understanding the algorithm, so matrix is constructed and how the algorithm works. Also I wanted to allow some freedom for t [github](#) and is released under GNU/GPL3. If you wish to contact me send me an email to mosta



Local Alignment Algorithm Results:

Input:

Sequence *a*:

Sequence *b*:

Scoring in *s*: Match Mismatch Gap

Hint:
For similarity maximization,
match scores should be positive and all other scores lower.

Recursion:

$$S_{i,j} = \max \begin{cases} S_{i-1,j-1} + s(a_i, b_j) \\ S_{i-1,j} + s(a_i, -) \\ S_{i,j-1} + s(-, b_j) \\ 0 \end{cases} = \max \begin{cases} S_{i-1,j-1} + 1 & a_i = b_j \\ S_{i-1,j-1} + -1 & a_i \neq b_j \\ S_{i-1,j} + -2 & b_j = - \\ S_{i,j-1} + -2 & a_i = - \\ 0 \end{cases}$$

Output:

<i>S</i>	C ₁	T ₂	A ₃	A ₄
	0	0	0	0
A ₁	0	0	0	1
G ₂	0	0	0	0
C ₃	0	1	0	0

Score: 1

Results
You can select a result to get the related traceback.

Global Alignment Algorithm excluding the beginning space of S and end space of T Results:

Interactive demo for Needleman-Wunsch algorithm

The motivation behind this demo is that I had some difficulty understanding the algorithm matrix is constructed and how the algorithm works. Also I wanted to allow some freedom and is released under GNU/GPL3. If you wish to contact me send me an email to github

Sequence 1:

Sequence 2:

Match Score:
Mismatch Score:
Gap Score:

		C	T	A
	0	-2	-4	-6
G	-2	-1	-3	-5
C	-4	-1	-2	-4

	C	T	A
C			
T			
A			

Score = -4

Web pages used to generate solutions for problem 2, question 1.:

<http://rna.informatik.uni-freiburg.de/Teaching/index.jsp?toolName=Smith-Waterman>

<http://experiments.mostafa.io/public/needleman-wunsch/>

2. Prove or disprove the statement $L \geq H \geq G$.

Since L has no gaps and runs the full length of $m * n$ in the array, we get the highest score possible to find our sequence. Since L is not affected by gaps and mismatches, the highest score is its only score that Local Alignment Algorithm can find. Looking at H , the highest value of H would have to have a score which isn't affected by gaps and mismatches, this value would be equal to L , any mismatches or gaps would decrease its score. Also, the greater the length of the target substring, the more possible need for revisions which would lower the score for both iterations of the Global Alignment Algorithm, or, the less of a string you align, the less need for revisions, which results in a higher score on the GAA. This keeps within our assumption that $L \geq H$.

Next, we look at H compared to G . Since both H and G are Global Alignment Algorithms, we look at the conditions that make H different. H , when ran, excludes the beginning space of S and end space of T . Since we take out two possible points where H can have a mismatch or gap, it will have a score higher than if we ran it without the exclusions. Therefore, H can be equal to or larger than G . Which fits our assumption that $H \geq G$.

In conclusion, the statement $L \geq H \geq G$ is true for all strings compared in all three algorithms.

4 Problem 3

Answer screenshots:

```
What is the filename of the fasta file (please include .fa)  
sequences.fa
```

```
Sequence 1: ACTGGGAAA
```

```
Sequence 2: CTGGAACA
```

```
Delta match value:
```

```
1
```

```
Delta mismatch value:
```

```
0
```

```
Delta Insertion/Deletion value:
```

```
0
```

```
Best Outcome(s) (Score: 7)
```

```
ACTGGGAA_A
```

```
_CT_GGAACA
```

```
ACTGGGAA_A
```

```
_CTG_GAACA
```

```
ACTGGGAA_A
```

```
_CTGG_AACA
```

```
What is the filename of the fasta file (please include .fa)
```

```
sequences.fa
```

```
Sequence 1: ACTGGGAAA
```

```
Sequence 2: CTGGAACA
```

```
Delta match value:
```

```
0
```

```
Delta mismatch value:
```

```
-1
```

```
Delta Insertion/Deletion value:
```

```
-2
```

```
Best Outcome(s) (Score: -4)
```

```
ACTGGGAAA
```

```
_CTGGAACA
```

What is the filename of the fasta file (please include .fa)

test.fa

Sequence 1: CAAT

Sequence 2: CAT

Delta match value:

1

Delta mismatch value:

0

Delta Insertion/Deletion value:

0

Best Outcome(s) (Score: 3)

CAAT

C_AT

CAAT

CA_T

What is the filename of the fasta file (please include .fa)

test2.fa

Sequence 1: AGATCCH

Sequence 2: AGTCHA

Delta match value:

1

Delta mismatch value:

0

Delta Insertion/Deletion value:

0

Best Outcome(s) (Score: 5)

AGATCCH_

AG_T_CHA

AGATCCH_

AG_TC_HA

5 Main.java

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.Scanner;

public class Main {
    public static String seq1;
    public static String seq2;
    public static Point[][] matrix;
    public static int dmatch;
    public static int dmismatch;
    public static int dinser;

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        System.out.println("What is the filename of the fasta file (please
            include .fa)");
        Scanner sc=new Scanner(System.in);
        String filename=sc.nextLine();

        try {
            BufferedReader br = new BufferedReader(new
                FileReader(filename));

            br.readLine();
            seq1=br.readLine();
            br.readLine();
            seq2=br.readLine();

            System.out.println("Sequence 1: "+seq1);
            System.out.println("Sequence 2: "+seq2);
            br.close();
        } catch (Exception e) {
            System.out.println("Could not find file or had error in reading
                file. Please run again.");
            System.out.println("Error: "+e);
            sc.close();
            return;
        }

        System.out.println("Delta match value: ");
        dmatch=sc.nextInt();
        System.out.println("Delta mismatch value: ");
        dmismatch=sc.nextInt();
        System.out.println("Delta Insertion/Deletion value: ");
        dinser=sc.nextInt();
    }
}
```

```

        sc.close();

        matrix=new Point[seq1.length()+1][seq2.length()+1];
        buildMatrix(seq1.length(),seq2.length());

        System.out.println("Best Outcome(s) (Score:
            "+matrix[seq1.length()][seq2.length()].getNum()+"");
        printAlignments("", "", seq1.length(), seq2.length());
    }

    public static int buildMatrix(int i,int j) {
        int match=Integer.MIN_VALUE;
        int deletion=Integer.MIN_VALUE;
        int insertion=Integer.MIN_VALUE;

        if(i==0 && j==0) { //for [0][0]
            if(matrix[i][j]==null) {
                matrix[i][j]=new Point(0);
            }
            return 0;
        }else {
            // was multiple base cases, but errors occurred with sum of
            delta values being negative and Integer.MIN_VALUE
            if(i-1>=0 && j-1>=0) {
                Point check1=matrix[i-1][j-1];
                if(check1==null) {
                    match=buildMatrix(i-1,j-1)+testMatch(i-1,j-1);
                } else {
                    match=check1.getNum()+testMatch(i-1,j-1);
                }
            }

            if(i-1>=0 && j>=0) {
                Point check2=matrix[i-1][j];
                if(check2==null) {
                    deletion=buildMatrix(i-1,j)+dinsert;
                } else {
                    deletion=check2.getNum()+dinsert;
                }
            }

            if(i>=0 && j-1>=0) {
                Point check3=matrix[i][j-1];
                if(check3==null) {
                    insertion=buildMatrix(i,j-1)+dinsert;
                } else {
                    insertion=check3.getNum()+dinsert;
                }
            }
        }
    }

```

```

    }

    boolean[] maxArray=new boolean[3]; // [match/mismatch,
        deletion, insertion]
    int point=0;

    if(match>=deletion && match>=insertion) { //builds point arrows
        using booleans
        maxArray[0]=true;
        point=match;
    }
    if(deletion>=match && deletion>=insertion) {
        maxArray[1]=true;
        point=deletion;
    }
    if(insertion>=match && insertion>=deletion) {
        maxArray[2]=true;
        point=insertion;
    }
    matrix[i][j]=new Point(point,maxArray);

    return point;
}
}

private static int testMatch(int s, int t) {
    // TODO Auto-generated method stub
    if(seq1.charAt(s)==seq2.charAt(t)) {
        return dmatch;
    }else {
        return dmismatch;
    }
}

private static void printAlignments(String s1,String s2,int i,int j) {
    // TODO Auto-generated method stub

    if(i==0 && j==0) { //base case
        System.out.println(s1);
        System.out.println(s2);
        System.out.println();
    }else { //works backwards from bottom right in matrix, checks each
        path so prints all paths
        boolean[] array=matrix[i][j].getArray();
        if(array[0]) {
            char letter1=seq1.charAt(i-1);
            char letter2=seq2.charAt(j-1);
            String new_s1=letter1+s1;

```

```

        String new_s2=letter2+s2;
        printAlignments(new_s1,new_s2,i-1,j-1);
    }
    if(array[1]) {
        char letter=seq1.charAt(i-1);
        String new_s1=letter+s1;
        String new_s2="_"+s2;
        printAlignments(new_s1,new_s2,i-1,j);
    }
    if(array[2]) {
        char letter=seq2.charAt(j-1);
        String new_s1="_"+s1;
        String new_s2=letter+s2;
        printAlignments(new_s1,new_s2,i,j-1);
    }
}
}
}

```

6 Point.java

```
public class Point {
    private int num;
    private boolean[] array; // [match/mismatch, deletion, insertion]

    public Point(int n) {
        // TODO Auto-generated constructor stub
        this.num=n;
        this.array=new boolean[3];
    }
    public Point(int n,boolean[] a) {
        this.num=n;
        this.array=a;
    }

    public int getNum() {
        return this.num;
    }
    public boolean[] getArray() {
        return this.array;
    }
}
```
