SULEYMAN DEMIREL
UNIVERSITY

Project Report

Topic: Parallelization of Smith-Waterman Local Sequence Alignment Algorithm

Submitted by:
190103190 - Bektursyn Azamat
190103157 - Merekeyev Raiymbek

Submitted to :
Azamat Serek

Course Code:
CSS 440
Course Name:
Distributed computing
Practice:
[03-P]

Parallelization of the alignment algorithm Smith-Waterman local sequences

ABSTRACT

In the existing local alignment system for gene sequencing, a sequential process is performed to implement the Smith-Waterman algorithm. This process takes a long time when the data size is huge. With the development of biotechnologies, we must be ready to process more information, as the amount of recorded data is growing day by day. Therefore, we parallelize the for loops that are used to compute the matrix using OpenMP constraints. This will help us reduce the execution time.

At the moment, we have made code in C (serial code) to implement the Smith-Waterman algorithm and made code for OpenMP. We need this in order to compare the results of the implementation for serial and parallel using OpenMP. In the future, we will visually show the results with different data sets. An analysis will be carried out after comparisons are made and the results are presented. Have the goals of our project changed after receiving some preliminary results? The goal of our project remains the same and is that we would like to show that the parallel implementation of local gene sequencing using the Smith-Waterman algorithm is advantageous compared to sequential. With the development of biotechnologies, parallel implementation would really help in processing large datasets used to solve this problem and we are working on it.

System Description

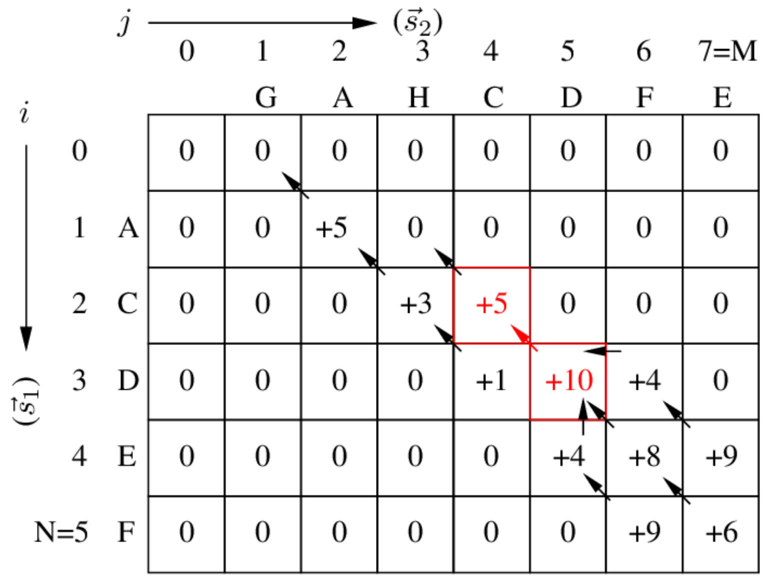
The Smith-Waterman algorithm is designed to obtain local sequence alignment, that is, to identify similar sites of two nucleotide or protein sequences. Instead of looking for entire pattern length given algorithm checks on segments of all possible length.

Smith-Waterman algorithm has several steps as initialization of matrix by filling the array based on matching. Algorithm aligns two sequences by matches and mismatches also insertion and deletion operations are represented by gaps.

1) 1) Initialize the scoring matrix. The dimensions of the scoring matrixare1+length of each sequence respectively. All the elements of the first row and the first column are set to 0. The extra first row and first column make it possible to align one sequence to another at any position, and settingthemto0 makes the terminal gap free from penalty.

2) 2) Scoring. Score each element from left to right, top to bottom in the matrix. The value of each cell is determined from the cell which has the maximum value among the cell just above it, or to its left or diagonal to it. If the cell which has the maximum value is diagonal to the computing cell weaddthevalue of the diagonal cell and the match value or mismatch value in case of match and mismatch respectively. If the cell which has the maximum value is the cell above it or to its left, we add the gap value to the maximum value. If none of the scores is positive, this element gets a 0. Otherwise, the highest score is used and the source of that score is recorded.

3) Traceback. Starting at the element with the highest score, traceback based on the source of each score recursively until 0 is encountered. The segments that have the highest similarity score based on the given scoring system are generated in this process. While performing traceback we write the elements of both gene sequences if the value came from its diagonal element otherwise 17BCE2421 17BCE0803 17BCE213512 | Pagewe insert a gap denoted by a dash. If the value of that cell came from the cell above it we introduce a gap at that position in the sequence which is written horizontally else if the value of that cell came from the cell to its left we introduce a gap at that position in the sequence which is written vertically. We stop when a 0 is encountered.

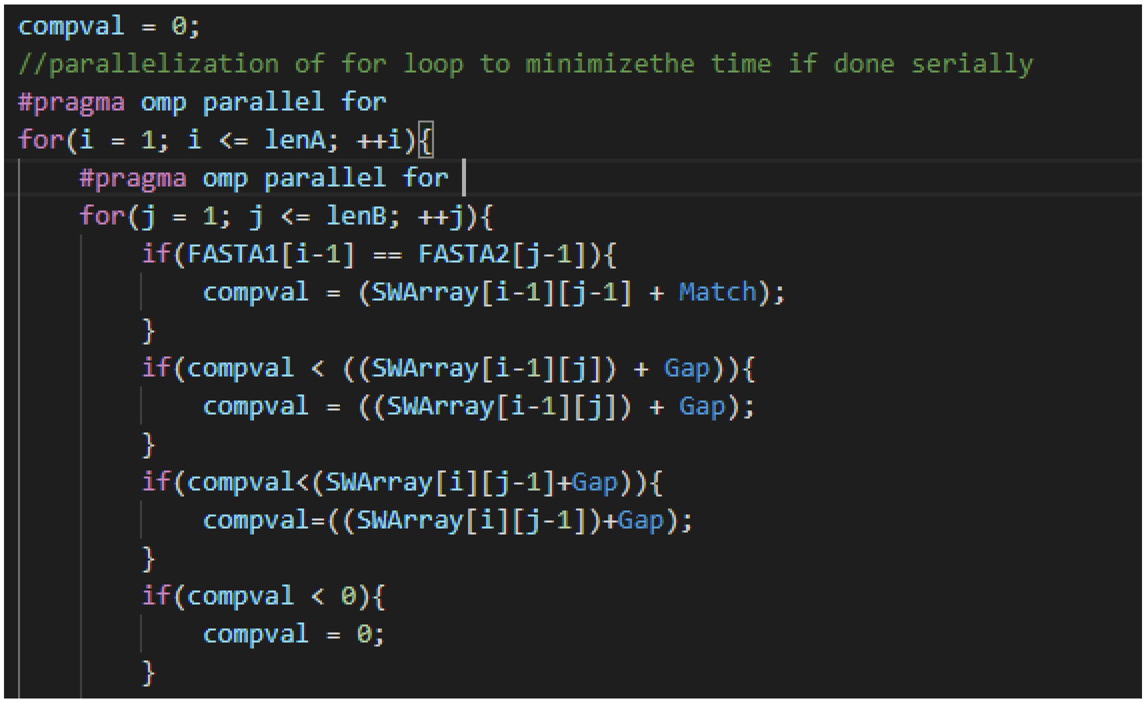


		0	1	2	3	4	5	6	7=M
			G	A	H	C	D	F	E
0		0	0	0	0	0	0	0	0
1	A	0	0	+5	0	0	0	0	0
2	C	0	0	0	+3	+5	0	0	0
3	D	0	0	0	0	+1	+10	+4	0
4	E	0	0	0	0	0	+4	+8	+9
N=5	F	0	0	0	0	0	0	+9	+6

Optimum alignment score: +10

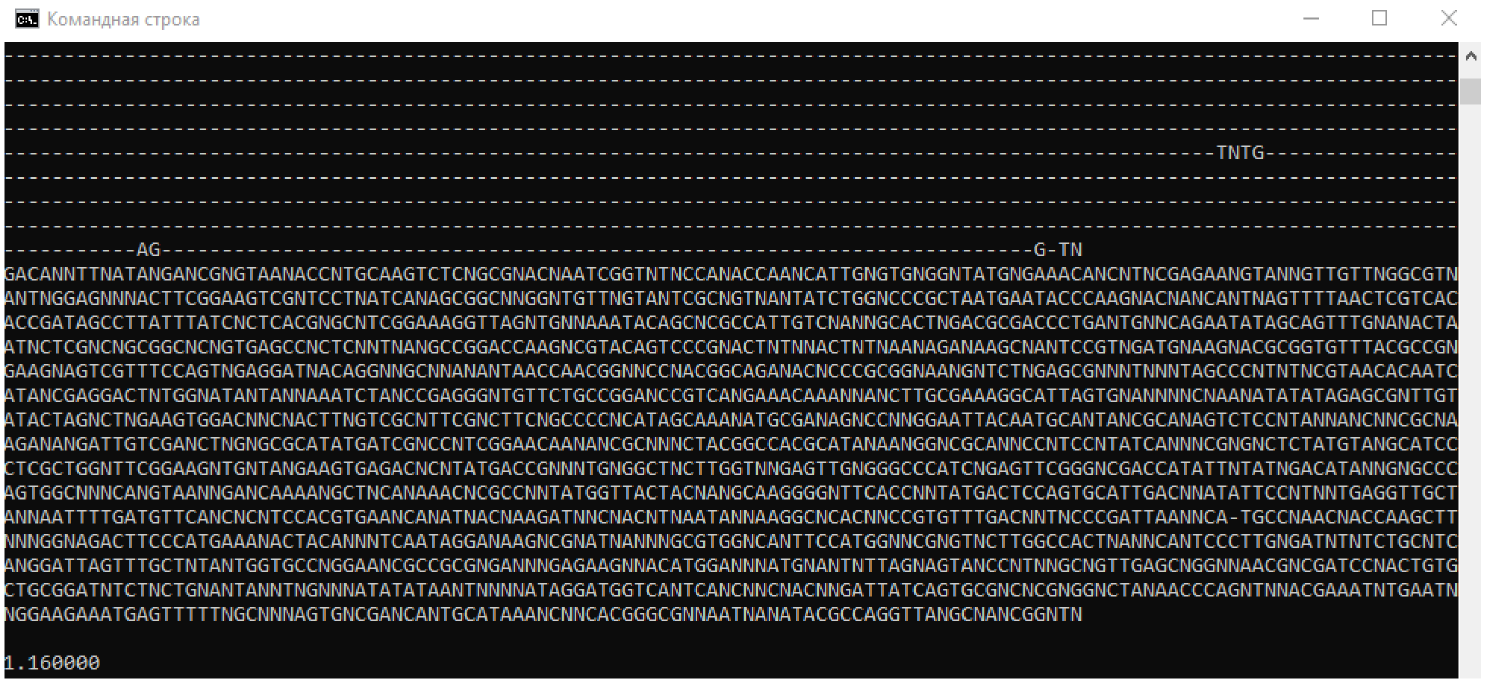
C	D
C	D
+5	+5

So, in this algorithm, we can parallelize the part where we are computing the scoring matrix, as we require a nested for loop for computing the scoring matrix.



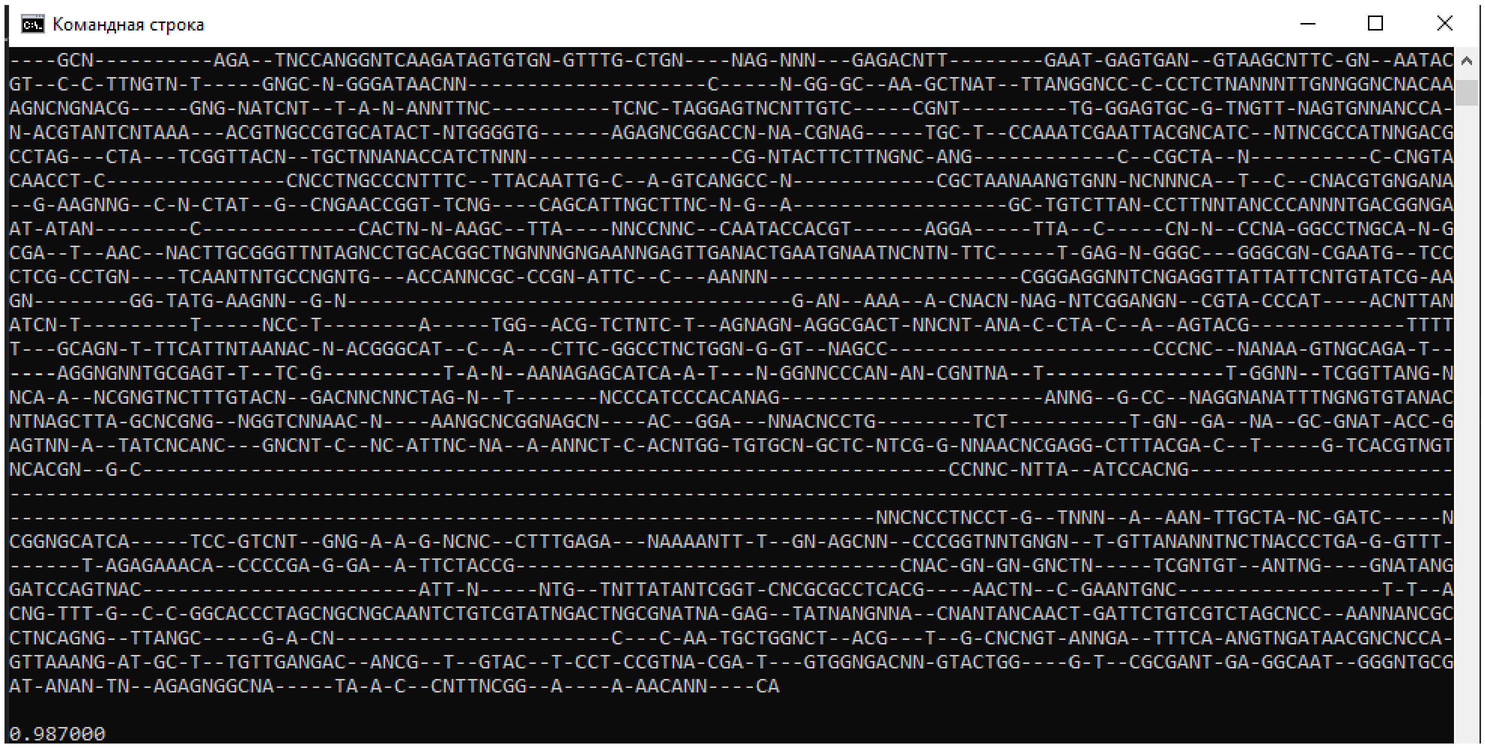
```c
compval = 0;
//parallelization of for loop to minimizethe time if done serially
#pragma omp parallel for
for(i = 1; i <= lenA; ++i){
    #pragma omp parallel for
    for(j = 1; j <= lenB; ++j){
        if(FASTA1[i-1] == FASTA2[j-1]){
            compval = (SWArray[i-1][j-1] + Match);
        }
        if(compval < ((SWArray[i-1][j]) + Gap)){
            compval = ((SWArray[i-1][j]) + Gap);
        }
        if(compval<(SWArray[i][j-1]+Gap)){
            compval=((SWArray[i][j-1])+Gap);
        }
        if(compval < 0){
            compval = 0;
        }
```

<u>Data-set:</u> 100000



```
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------TNTG----------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-----------AG-------------------------------------------------------------------------G-TN
GACANNTTNATANGANCGNGTAANACCNTGCAAGTCTCNGCGNACNAATCGGTNTNCCANACCAANCATTGNGTGNGGNTATGNGAAACANCNTNCGAGAANGTANNGTTGTTNGGCGTN
ANTNGGAGNNNACTTCGGAAGTCGNTCCTNATCANAGCGGCNNGGNTGTTNGTANTCGCNGTNANTATCTGGNCCCGCTAATGAATACCCAAGNACNANCANTNAGTTTTAACTCGTCAC
ACCGATAGCCTTATTTATCNCTCACGNGCNTCGGAAAGGTTAGNTGNNAAATACAGCNCGCCATTGTCNANNGCACTNGACGCGACCCTGANTGNNCAGAATATAGCAGTTTGNANACTA
ATNCTCGNCNGCGGCNCNGTGAGCCNCTCNNTNANGCCGGACCAAGNCGTACAGTCCCGNACTNTNNACTNTNAANAGANAAGCNANTCCGTNGATGNAAGNACGCGGTGTTTACGCCGN
GAAGNAGTCGTTTCCAGTNGAGGATNACAGGNNGCNNANANTAACCAACGGNNCCNACGGCAGANACNCCCGCGGNAANGNTCTNGAGCGNNNTNNNTAGCCCNTNTNCGTAACACAATC
ATANCGAGGACTNTGGNATANTANNAAATCTANCCGAGGGNTGTTCTGCCGGANCCGTCANGAAACAAANNANCTTGCGAAAGGCATTAGTGNANNNNCNAANATATATAGAGCGNTTGT
ATACTAGNCTNGAAGTGGACNNCNACTTNGTCGCNTTCGNCTTCNGCCCCNCATAGCAAANATGCGANAGNCCNNGGAATTACAATGCANTANCGCANAGTCTCCNTANNANCNNCGCNA
AGANANGATTGTCGANCTNGNGCGCATATGATCGNCCNTCGGAACAANANCGCNNNCTACGGCCACGCATANAANGGNCGCANNCCNTCCNTATCANNNCGNGNCTCTATGTANGCATCC
CTCGCTGGNTTCGGAAGNTGNTANGAAGTGAGACNCNTATGACCGNNNTGNGGCTNCTTGGTNNGAGTTGNGGGCCCATCNGAGTTCGGGNCGACCATATTNTATNGACATANNGNGCCC
AGTGGCNNNCANGTAANNGANCAAAANGCTNCANAAACNCGCCNNTATGGTTACTACNANGCAAGGGGNTTCACCNNTATGACTCCAGTGCATTGACNNATATTCCNTNNTGAGGTTGCT
ANNAATTTTGATGTTCANCNCNTCCACGTGAANCANATNACNAAGATNNCNACNTNAATANNAAGGCNCACNNCCGTGTTTGACNNTNCCCGATTAANNCA-TGCCNAACNACCAAGCTT
NNNGGNAGACTTCCCATGAAANACTACANNNTCAATAGGANAAGNCGNATNANNNGCGTGGNCANTTCCATGGNNCGNGTNCTTGGCCACTNANNCANTCCCTTGNGATNTNTCTGCNTC
ANGGATTAGTTTGCTNTANTGGTGCCNGGAANCGCCGCGNGANNNGAGAAGNNACATGGANNNATGNANTNTTAGNAGTANCCNTNNGCNGTTGAGCNGGNNAACGNCGATCCNACTGTG
CTGCGGATNTCTNCTGNANTANNTNGNNNATATATAANTNNNNATAGGATGGTCANTCANCNNCNACNNGATTATCAGTGCGNCNCGNGGNCTANAACCCAGNTNNACGAAATNTGAATN
NGGAAGAAATGAGTTTTTNGCNNNAGTGNCGANCANTGCATAAANCNNCACGGGCGNNAATNANATACGCCAGGTTANGCNANCGGNTN

1.160000
```



```
----GCN----------AGA--TNCCANGGNTCAAGATAGTGTGN-GTTTG-CTGN----NAG-NNN---GAGACNTT--------GAAT-GAGTGAN--GTAAGCNTTC-GN--AATAC
GT--C-C-TTNGTN-T-----GNGC-N-GGGATAACNN--------------------C-----N-GG-GC--AA-GCTNAT--TTANGGNCC-C-CCTCTNANNNTTGNNGGNCNACAA
AGNCNGNACG-----GNG-NATCNT--T-A-N-ANNTTNC----------TCNC-TAGGAGTNCNTTGTC-----CGNT---------TG-GGAGTGC-G-TNGTT-NAGTGNNANCCA-
N-ACGTANTCNTAAA---ACGTNGCCGTGCATACT-NTGGGGTG------AGAGNCGGACCN-NA-CGNAG-----TGC-T--CCAAATCGAATTACGNCATC--NTNCGCCATNNGACG
CCTAG---CTA---TCGGTTACN--TGCTNNANACCATCTNNN----------------CG-NTACTTCTTNGNC-ANG------------C--CGCTA--N---------C-CNGTA
CAACCT-C---------------CNCCTNGCCCNTTTC--TTACAATTG-C--A-GTCANGCC-N------------CGCTAANAANGTGNN-NCNNNCA--T--C--CNACGTGNGANA
--G-AAGNNG--C-N-CTAT--G--CNGAACCGGT-TCNG----CAGCATTNGCTTNC-N-G--A-----------------GC-TGTCTTAN-CCTTNNTANCCCANNNTGACGGNGA
AT-ATAN-------C------------CACTN-N-AAGC--TTA----NNCCNNC--CAATACCACGT------AGGA-----TTA--C-----CN-N--CCNA-GGCCTNGCA-N-G
CGA--T--AAC--NACTTGCGGGTTNTAGNCCTGCACGGCTNGNNNGNGAANNGAGTTGANACTGAATGNAATNCNTN-TTC-----T-GAG-N-GGGC---GGGCGN-CGAATG--TCC
CTCG-CCTGN----TCAANTNTGCCNGNTG---ACCANNCGC-CCGN-ATTC--C---AANNN---------------------CGGGAGGNNTCNGAGGTTATTATTCNTGTATCG-AA
GN--------GG-TATG-AAGNN--G-N-------------------------------------G-AN--AAA--A-CNACN-NAG-NTCGGANGN--CGTA-CCCAT----ACNTTAN
ATCN-T---------T-----NCC-T--------A-----TGG--ACG-TCTNTC-T--AGNAGN-AGGCGACT-NNCNT-ANA-C-CTA-C--A--AGTACG------------TTTT
T---GCAGN-T-TTCATTNTAANAC-N-ACGGGCAT--C--A---CTTC-GGCCTNCTGGN-G-GT--NAGCC----------------------CCCNC--NANAA-GTNGCAGA-T--
----AGGNGNNTGCGAGT-T--TC-G----------T-A-N--AANAGAGCATCA-A-T---N-GGNNCCCAN-AN-CGNTNA--T--------------T-GGNN--TCGGTTANG-N
NCA-A--NCGNGTNCTTTGTACN--GACNNCNNCTAG-N--T-------NCCCATCCCACANAG----------------------ANNG--G-CC--NAGGNANATTTNGNGTGTANAC
NTNAGCTTA-GCNCGNG--NGGTCNNAAC-N----AANGCNCGGNAGCN----AC--GGA---NNACNCCTG--------TCT----------T-GN--GA--NA--GC-GNAT-ACC-G
AGTNN-A--TATCNCANC---GNCNT-C--NC-ATTNC-NA--A-ANNCT-C-ACNTGG-TGTGCN-GCTC-NTCG-G-NNAACNCGAGG-CTTTACGA-C--T-----G-TCACGTNGT
NCACGN--G-C-------------------------------------------------------------------CCNNC-NTTA--ATCCACNG----------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------NNCNCCTNCCT-G--TNNN--A--AAN-TTGCTA-NC-GATC-----N
CGGNGCATCA-----TCC-GTCNT--GNG-A-A-G-NCNC--CTTTGAGA---NAAAANTT-T--GN-AGCNN--CCCGGTNNTGNGN--T-GTTANANNTNCTNACCCTGA-G-GTTT-
------T-AGAGAAACA--CCCCGA-G-GA--A-TTCTACCG--------------------------------CNAC-GN-GN-GNCTN-----TCGNTGT--ANTNG----GNATANG
GATCCAGTNAC-----------------------ATT-N-----NTG--TNTTATANTCGGT-CNCGCGCCTCACG----AACTN--C-GAANTGNC-----------------T-T--A
CNG-TTT-G--C-C-GGCACCCTAGCNGCNGCAANTCTGTCGTATNGACTNGCGNATNA-GAG--TATNANGNNA--CNANTANCAACT-GATTCTGTCGTCTAGCNCC--AANNANCGC
CTNCAGNG--TTANGC-----G-A-CN-----------------------C---C-AA-TGCTGGNCT--ACG---T--G-CNCNGT-ANNGA--TTTCA-ANGTNGATAACGNCNCCA-
GTTAAANG-AT-GC-T--TGTTGANGAC--ANCG--T--GTAC--T-CCT-CCGTNA-CGA-T---GTGGNGACNN-GTACTGG----G-T--CGCGANT-GA-GGCAAT--GGGNTGCG
AT-ANAN-TN--AGAGNGGCNA-----TA-A-C--CNTTNCGG--A----A-AACANN----CA

0.987000
```

Result:

Data-set	100_000
serial	1.16000
OpenMP	0.987000

CONCLUSION:

In our project, we have implemented the Smith-Waterman Algorithm Serial and parallel manner (openMP) for local alignment of gene sequences. We have compared the results of the implementation of sequential, parallel using OpenMP. Analysis has been carried out after comparisons and results have been presented. At last, we would like to conclude that parallel implementation of local gene sequencing using the Smith-Waterman algorithm is advantageous when compared to sequential one. With the development in biotechnology parallel implementation would actually help in processing out large data sets used for this problem.

We have taken datasets from this site :

https://www.ncbi.nlm.nih.gov/nuccore

The role of each team member:

Merekeyev Raiymbek (190103157) - Implementation of traceback part of the algorithm, parallelization in OpenMP

Bektusyn Azamat (190103190) - Implementation of Initialization part of the algorithm, parallelization in OpenMP

Link for project proposal

https://docs.google.com/presentation/d/1Nu3gPCljd8bKvl7vf_Gzbtim88ERWdFyTw2ccleKxG0/edit?usp=sharing

Link for Smith-Waterman algorithm explanation

https://docs.google.com/presentation/d/1-CGTlzV8zuuKCwrmMqHcAZAkilivtZHUY38i8-BYQwg/edit?usp=sharing