MSBD5009 Parallel Programming
Assignment #3: CUDA Programming

Due on 12 May 2020 at 11:59pm

Instructions

- This assignment counts for 20 points.
- This is an individual assignment. You can discuss with others and search online resources, but your submission should be your own code.
- Fill your name, student ID and email in the first line of comments.
- Submit your assignment through Canvas before the deadline.
- Your submission will be compiled and tested on Azure NC6_Promo machines
- **No late submissions will be accepted!**

Assignment Description

The Smith-Waterman algorithm identifies the similar regions in two genome sequences. In this assignment, you will implement a **CUDA version** of the Smith-Waterman algorithm to measure the similarity between two strings.

The pseudocode of the Smith-Waterman algorithm is shown in Algorithm 1. Given string $A = a_1, a_2, \ldots, a_n$ and string $B = b_1, b_2, \ldots, b_m$, we first construct a scoring matrix H of size $(n+1) * (m+1)$ and set the first row and column to zero. Then, we iteratively compute the scoring matrix using the following dynamic programming formula:

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s\left(a_i, b_j\right) \\ H_{i-1,j} - w \\ H_{i,j-1} - w \\ 0 \end{cases} \qquad (1 \le i \le n, 1 \le j \le m)$$

where s is the substitution matrix $s\left(a_i, b_j\right) = \begin{cases} u, & a_i = b_j \\ v, & a_i \neq b_j \end{cases}$, u is the match score, v is the mismatch score, and w is the gap penalty. The value of u, v and w are constant and are given as part of the input to the algorithm. Finally, we return the highest score in the scoring matrix as the similarity score between A and B.

Input and Output

The input file will be in the following format:

1. The first line contains two integers a_len and b_len ($1 \le a\_len \le 20000, 1 \le b\_len \le 20000$), separated by a space. They represent the length of string a and b, respectively.
2. The second line is a string a of length a_len, consisting of four letters A, T, G, C.
3. The third line is a string b of length b_len, consisting of four letters A, T, G, C.

Algorithm 1: The Smith-Waterman Algorithm

Input : string $A = a_1, a_2, \ldots, a_n$ and $B = b_1, b_2, \ldots, b_m$, match score u, mismatch score v, gap penalty w

Output: the similarity score between A and B

```
1  int score[|A| + 1][|B| + 1];
2  for i ← 0 to |A| do
3  |  score[i][0] ← 0;
4  end
5  for i ← 0 to |B| do
6  |  score[0][i] ← 0;
7  end
8  int max_score ← 0;
9  for i ← 1 to |A| do
10 |  for j ← 1 to |B| do
11 |  |  score[i][j] ← max(0,
12 |  |                    score[i − 1][j] − w,
13 |  |                    score[i][j − 1] − w,
14 |  |                    score[i − 1][j − 1] + sub_mat(a_i, b_j));
15 |  |  max_score ← max(max_score, score[i][j]);
16 |  end
17 end
18 return max_score;
19 Procedure sub_mat(char x, char y)
20 |  if x == y then
21 |  |  return u;
22 |  else
23 |  |  return v;
24 |  end
```

The output of the program consists of three lines:

1. The similarity score between a and b, i.e., the highest score in the scoring matrix.
2. The elapsed time in seconds of the execution.
3. The driver time in seconds of the execution.

We assume that the match score is 3, the mismatch score is -3, and the gap penalty is 2.

Here is an example input/output for your reference:

Input:

```
9 8
GGTTGACTA
TGTTACGG
```

Output:

```
Max socre: 13
Elapsed time: 0.00001 s
Driver time: 0.0000095 s
```

Submission and Grading

The code skeleton `cuda_smith_waterman_skeleton.cu` is provided. Your task is to complete the following function in the code:

```
int smith_waterman(int blocks_per_grid, int threads_per_block, char *a, char *b, int a_len, int b_len);
```

The description of the parameters is as follows:

Parameter	Description
`int blocks_per_grid`	Number of blocks per grid
`int threads_per_block`	Number of threads per block
`char *a`	The string a
`char *b`	The string b
`int a_len`	The length of string a
`int b_len`	The length of string b

Notes:

1. You will be given three files `cuda_smith_waterman_skeleton.cu`, `main.cu` and `cuda_smith_waterman.h`. You only need to complete and submit the `cuda_smith_waterman_skeleton.cu` to the Canvas.
2. The sequential Smith-Waterman algorithm code is also provided for your reference. You can use it to learn the logic flow and verify the correctness of your CUDA version. You can also compare the sequential running time with your parallel program performance.
3. You can add helper functions and variables as you wish in the `cuda_smith_waterman_skeleton.cu` file, but keep the other two files: `main.cu` and `cuda_smith_waterman.h`, unchanged.
4. We will use different input files and specify different CUDA kernel launch parameters ($13 \leq$ `block_per_grid` ≤ 39 and $32 \leq$ `threads_per_block` $\leq$ `1024` in `./cuda_smith_waterman <input file> <block_per_grid> <threads_per_block>`) to test your program.
5. The correctness, elapsed time and speedup of your program will be considered in grading.
6. We will perform code similarity checks. In case a submission has code similarity issues, we may request clarification and deduct partial marks or full marks on a case-by-case basis.