

Programming Assignment #1

Author: Yu Sheng Lin

Instructor: Wei Chao Chen

March 11, 2016

1 Goals

You have to

1. Learn how to implement some parallel algorithms on GPU.
2. Learn how to use the Thrust library.
3. Be creative!

in this assignment.

2 Requirements

This is a three-part assignment. Part I and II are fixed assignments, meaning you would complete the assignments according to instructions. Part III is an open assignment, where you should try to be as creative as you can to gain additional bonus points.

2.1 Part I: Count the Position in Words (60pts)

In this part you are required to count the position of each character inside the word it belongs to. Listing 1. provides a sample input and output.

```
1 gpu qq  a hello  sonoda (input)
2 123012001012345000123456 (output)
```

Listing 1: Example: Count the Position in Words.

You may easily come up with an $O(n)$ sequential algorithm and an $O(nk)$ parallel algorithm, where n is the length of the input and k is the maximum length of a word. However in this assignment k is purposely large, and we challenge you to come up with an $O(n \ln k)$ algorithm. We provide some hints about the algorithm in a separate PDF, and you can decide whether to read them by yourself.

The input is generated in a pseudo-random fashion, and you can assume that $k = 500$, $n \approx 4 \times 10^7$. Besides, the input only contains characters [a-z] and we use linebreak '\n' as the spaces.

You have to implement a function whose signature is Listing 2.. All pointers are device pointers and `text_size` is the n .

We also provide some hints not relating to the algorithm itself.

```
1 void CountPosition(const char *text, int *pos, int text_size);
```

Listing 2: The function signature of part I.

- A kernel may start before previous kernels have finished. To prevent this, insert `cudaDeviceSynchronize()` between them.
- `gridDim.x` cannot exceed $2^{17} = 130172$ if you don't use `-arch sm_30` compile flag.

2.2 Part II: Find the Heads (40pts)

Using the result of part I, you can find the "heads", i.e., the position of the first characters of each words. You can only use `thrust::*` functions and CUDA API in this part. To be more precise, `__global__` functions are not allowed. We will give you correct answer of part I, so you can do this part even if you cannot finish part I.

Listing 3. is the sample input and output.

```
1 123012001012345000123456 (input)
2 ^ ^ ^ ^ ^
3 0,4,8,10,18 (output)
```

Listing 3: Example: Extract the Heads

Here are some hints:

- You will need the document <https://thrust.github.io/doc/modules.html>
- I have already included some necessary headers.
- I used "_____ " (white text).

The function signature of part II is Listing 4., and `head` is large enough to hold all heads (actually, the same size as `pos`). You should return the number of heads you find, and you should fill ALL HEADS IN-ORDER in `head`. The data after the heads can be left uninitialized/undefined.

```
1 int ExtractHead(const int *pos, int *head, int text_size);
```

Listing 4: The function signature of part II.

2.3 Part III: Be Creative! (20pts bonus)

With the results you got in previous parts, please do something interesting.

If you had no ideas about what to do, then Listing 5. shows some possible outputs. The first one is the "quite difficult" part in assignment #0. I think that it's not that difficult now (hopefully). The second one clamps the words whose lengths are more than 3.

```
1 abcdeffof abc abcd  abab (input)
2 badcfeoff bac badc  baba (possible output 1)
3 abc- abc abc-  aba-  (possible output 2)
```

Listing 5: Sample outputs for part III.

The points you will get is based on your creativity, and you will get 10pts if you implement the examples we provide.

3 Submission

- The submission deadline is 2016/3/28 midnight (Mon.).
- The efficiency of part I will also be considered during grading, and pure CPU implementations would be disqualified.
- The efficiency of part II will NOT be considered during grading, but if you call any `__global__` functions, you will get 0pt.
- For part III, you should submit a report `report.pdf` which describes what your code does and how you do that.
- You can only modify `lab1/counting.cu`, and we will only copy this file from your repo.
- The compile flags are `--std=c++11 -arch sm_30 -O2`.
- Please also refer to assignment #0 for more details.