COMP4901D Assignment 2
Merge-Join in CUDA
Due: 5PM on Apr 22

Instructions

- This assignment counts 20 points.
- Fill in the kernel function **mergeJoin** in the provided source file *ass2.cu* with the function implemented as required in the assignment description.
- In your completed "*ass2.cu*" source file, put down your name, ITSC account, and student ID as a comment (surrounded by "/*" and "*/") on the first line.
- Submit your *ass2.cu* file through the Course Assignment Submission System (CASS) before the deadline: https://course.cse.ust.hk/cass/submit.html
- Instructions on using CASS are available online: http://cssystem.cse.ust.hk/home.php?docbase=UGuides/cass&req_url=UGuides/cass/student.html
- Your submission will be compiled and tested on a lab machine.
- No late submissions will be accepted.

Assignment Description

Merge-join, also known as sort-merge join, is a join algorithm that first sorts all tables by the join attribute and then performs a merge on these sorted tables. In assignment 2, you are to implement a merge join kernel on two sorted arrays. Specifically, given two arrays A and B of arbitrary sizes, whose elements are (key, value) pairs sorted by keys, your kernel program is to return all pairs of A and B elements that have the same key. For simplicity, the keys are non-negative integers and the values are floats. Moreover, there is no duplicate key in each array. As a result, for each element in array A, there is at most one element in array B that has the same key as the A element.

The merge join kernel on two sorted arrays works as follows: Denote the array of smaller or equal size B and the other A. The number of keys that can fit into the shared memory of a thread block is S. The kernel loads each B chunk of size S (The last chunk may be less than S) into the shared memory, and uses the keys of the first and last elements of each B chunk to identify the start and end positions of its matching chunk in A. Then each thread reads one or more elements from the matching A chunk and performs a search on the B chunk for matching. Either a sequential search or a binary search can be used.

The host code is provided. Your task is to complete the following kernel function. Both array A and array B are **already sorted** by the key attribute. Please keep the host code **as is** to ensure the correct output format.

__global__ void mergeJoin
*(int *key1, float* value1, int *key2, float* value2, int N1, int N2, int *result)*

Parameter	Description
*int *key1*	keys of array A (N1 keys in total, N1 <= 500,000)
*float *value1*	values of array A (*N1* values in total)
*int *key2*	keys of array B (*N2* keys in total, N2 <= 500,000)
*float *value2*	values of array B (*N2* values in total)
int N1	size (in number of elements) of array *A*
int N2	size (in number of elements) of array *B*
*int *result*	output array: *result[i] = j*, if there is an element (*key1[i],value1[i]*) in array *A*, there is an element (*key2[j],value2[j]*) in array *B*, and *key1[i] == key2[j]*; otherwise, *result[i] = -1*.

For example, given the following input

```
int key1[10] = {1,2,3,5,6,9,11,15,19,23};
float value1[10] = {0.2,0.5,0.7, 0.9,1.2, 0.2,0.5,0.7, 0.9,1.2};
int N1 = 5;
int key2[7] = {1,5,7,9,22,50,78};
float value2[7] = {0.5,0.1,0.7,0.2,0.8,1.2,1.6};
int N2 = 7;
```

The result array is

```
int result[10] = {0,-1,-1,1,-1,3,-1,-1,-1,-1};
```