COMP4901D Assignment 1
Equi-Join in CUDA
Due: 5PM on Mar 18 Wednesday

Instructions

- This assignment counts ten points, with each of the two kernel functions counting five points.
- Fill in the two kernel functions in the provided source file *ass1.cu* with the functions implemented as required in the assignment description.
- In your completed "*ass1.cu*" source file, put down your name, ITSC account, and student ID as a comment (surrounded by “/*” and “*/”) on the first line.
- Submit your *ass1.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

A JOIN operator in SQL is used to combine two or more tables in a relational database. Equi-join is a join that uses equality comparisons in the join condition. In this assignment, you are expected to implement a parallel equi-join algorithm in CUDA.

Specifically, given two arrays A and B, whose elements are (key, value) pairs, your algorithm 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. Finally, you are suggested to use the nested-loop join algorithm: For each element in A, loop over B to find all matching elements.

The host code is provided. Your task is to complete the following two kernel functions.

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

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

The two kernel functions have the same parameter signature and output the same result. The only difference is that *equiJoinTiled* is a tiled algorithm with tile width *TILE_WIDTH*.

Parameter	Description
*int *key1*	keys of array A (N1 keys in total)
*float *value1*	values of array A (*N1* values in total)
*int *key2*	keys of array B (*N2* keys in total)
*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[5] = {1,3,5,2,6};
float value1[5] = {0.2,0.5,0.7, 0.9,1.2};
int N1 = 5;
int key2[7] = {3,2,5,9,1,7,8};
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[5] = {4,0,2,1,-1};
```