COMP4901D Assignment 3
Hash-Join in CUDA
Due: 5PM on May 15

Instructions

- This assignment counts 20 points.
- Fill in the function **hashJoin** in the provided source file *ass3.cu* with the function implemented as specified in the assignment description.
- In your completed "*ass3.cu*" source file, put down your name, ITSC account, and student ID as a comment (surrounded by "/*" and "*/") on the first line.
- Submit your *ass3.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

Radix hash join is a join algorithm based on the split primitive. Denote the two input arrays A and B, where A's size is greater than or equal to that of B. The radix hash join on the two arrays works in two phases:

Phase 1) Partitioning. We split A and B into the same number of partitions using a given number of radix bits so that on average each B partition fits into the shared memory. The join on A and B is decomposed into multiple small joins on an A partition and its corresponding B partition.

Phase 2) Matching. Each block of threads load a B partition as the inner partition into the shared memory and use the corresponding A partition as the outer partition. For each key in the outer partition, we perform a sequential search or a binary search on the inner partition for matching. If the binary search is used, the inner partition should be sorted in the shared memory.

In assignment 3, you are to implement the radix hash join function on two unordered arrays. Specifically, given two arrays of arbitrary sizes, whose elements are (key, value) pairs in arbitrary order, your program is to return all pairs of elements that have the same key. For simplicity, the keys are non-negative integers in range [0, 2^19) and the values are floats. Moreover, there is no duplicate key in each array. As a result, for each element in one array, there is at most one element in the other array that has the same key.

The functions main, hashJoin, and a number of helper functions are provided.

Your task is to complete the two functions **split** and **join**.

Please keep the provided functions **as they are** to ensure the correct output format.

void hashJoin
*(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[7] = {3,2,5,9,1,7,8};
float value1[7] = {0.5,0.1,0.7,0.2,0.8,1.2,1.6};
int N1 = 7;
int key2[5] = {1,3,5,2,6};
float value2[5] = {0.2,0.5,0.7, 0.9,1.2};
int N2 = 5;
```

The result array is

```
int result[7] = {1,3,2,-1,0,-1,-1};
```