**Project Proposal**

**Description:**

Implement the project idea[sorting algorithms benchmark and implementation (2018)](https://wiki.postgresql.org/wiki/GSoC_2018#Sorting_algorithms_benchmark_and_implementation_.282018.29)

**Applicant Information:**

**Name:** Kefan Yang

**Country of Residence:** Canada

**University:** Simon Fraser University

**Year of Study:** Third year

**Major:** Computing Science

**Relevant experience:**

**Mini SQL – Database System Course Project**

A simple SQL database supporting basic SQL syntax like ‘select’, ‘insert’, ’delete’ and ‘update’ Implemented a disk-based b+ tree to index data records

GitHub: [https://github.com/ADS-DBAwesomeGroup/MiniSQL](https://github.com/ADS-DBAwesomeGroup/MiniSQL/)

**The 2017 ACM-ICPC** [**Pacific Northwest Regional Contest**](http://acmicpc-pacnw.org/)

Familiarity with basic sorting algorithms and deep understanding of traditional quick sort

**Deliverables:**

1. A benchmark for sorting algorithms.
2. Benchmark implementation for the sorting algorithms mentioned in research papers.
3. The winner under the given benchmark
4. Industrial implementation for the selected sorting algorithm in **pg\_qsort()**, **pg\_qsort\_args()** and **gen\_qsort\_tuple.pl**

**TimeLine:**

Community Bonding Period

* Get more familiar with codebase. Fully understand the usage of **pg\_qsort()**, **pg\_qsort\_args()** and **gen\_qsort\_tuple.pl**.
* Discuss on how to implement the benchmark – what is the “average” use case?
* Read research paper – what are the candidate algorithms?

Week 1-3

* Implement benchmark

Week 4-6

* Benchmark implementation of candidate algorithms

Week 7-8

* Select the winner and implement its industrial version for **pg\_qsort()** and **pg\_qsort\_args()**

Week 9

* Industrial implementation for **gen\_qsort\_tuple.pl** (may encounter some challenges with Perl script)

Week 10

* Test in database to see if performance is improved for sorting-related operations

Week 11-12

* Final buffer period in case things goes ugly in week 10

**Implementation**

**Benchmark:**

I plan to use integer arrays to test the performance of each sorting algorithm. The two major factors that may influence a performance of a sorting algorithm are

1. size of array
2. order of elements (this may be represented as number of inversions)

Therefore, I plan to generate random arrays with a certain size and inversions to test each algorithm. To generate the test data, I may call a subroutine like:

void generate\_test\_array(int\* array, int array\_size, int inversions)

I find the major challenge is to define the proportion of each test case in regard with how likely they are going to happen in practice. Now I am reading research papers to see if I can handle this issue somehow.

Besides, I still have some concerns like whether the runtime data type will greatly influence the relative performance of different algorithms. If that’s the case, I may need to take the data type into consideration.

**Benchmark implementation of sorting algorithms:**

Implement the pseudo-code provided by the research papers in C.

**Industrial implementation of selected sorting algorithm:**

The industrial version is basically an optimization based on the benchmark implementation. I plan to use optimizations like checking if input array is already sorted or applying insertion sort directly for short arrays to see if the performance can be improved. I am still looking for other common optimization methods.

**Test new qsort implementation in production environment:**

Currently, I am thinking about using SQL ‘sort’ operation to test my modules. However, since disk operation is much expensive than in-memory sorting, I am not sure if we can observe a significant difference in this way. I am still looking for better approaches.