COMP90077 Assignment 2 (20% out of the Subject)

An Experimental Study on Range Trees

Due Date: **11:59PM June 2 (Tuesday), 2020**

Prepared by **Junhao Gan** and **Tony Wirth**

Background. In Assignment 1, the instructors specified the experiments, the measured variables, and the notation, e.g., the percentage of search operations, and the size, n, of the dataset. However, in practice, when designing an algorithm and data structure for some purpose, you would also have to design the experiments by yourselves. Firstly, these experiments should be designed for the purpose of *demonstrating the anticipated performance* of certain algorithms. More importantly, the designed experiments should be reasonable and convincing.

In this assignment, you are asked to *design (part of) and conduct* an experimental study on different implementations of the 2-dimensional range tree, and then write a report (i.e., an experiment section) to: (i) describe and justify your design, and (ii) demonstrate and analyse the experimental results.

Specification. There are **four** main tasks in this assignment:

- (i) implement a data point and query generator and two variances of 2-dimensional range trees;
- (ii) design and conduct an experiment on the efficiency of different construction algorithms;
- (iii) conduct the required experiments on the query efficiency of the range tree variances;
- (iv) write a report to demonstrate and analyse the experimental results.

Task 1: Implementations

About Programming. There is *no* specified programming language in this assignment; you can use whatever programming language you prefer.

Data Point Generator. In the experiment, you will need to implement a data point generator. More specifically, you need to implement the following two functions:

- *generate_a_point*($coord_{min}$, $coord_{max}$): generate and return a point uniformly at random in the 2-dimensional *integer* space $[coord_{min}, coord_{max}]^2$, where $coord_{min}$ and $coord_{max}$ are positive integers.
- *generate_point_set*(n): generate and return a set P of n 2-dimensional points, each of which has a *unique* integer identifier in $[1, n]$ and is generated by *generate_a_point*(1, M), where $M = 10^6$, i.e., a million.

Query Generator.

- *generate_a_query*(s): generate and return a 2-dimensional axis-parallel *square* with a specified *range length* $1 < s < M$ via the following steps:
  - generate a 2-dimensional point q by invoking *generate_a_point*(1, $M - s$);

  - return a query range $Q = [q.x, q.x + s] \times [q.y, q.y + s]$, where $q.x$ and $q.y$ are the x- and y-coordinates of q, respectively.

The Range Tree Variances. You will need to implement the following *two* range tree variances:

- *RangeTree-Org*: The original range tree, where each internal node u in the base tree (on the x-coordinates) is associated with a secondary WB-BST on the y-coordinates of all the points in u's sub-tree.
- *RangeTree-FC*: The range tree incorporating *Fractional Cascading.*

Query algorithm. For each of the two variances, you will need to implement its corresponding *query* algorithm, to answer range reporting queries.

Construction algorithm. For *RangeTree-FC*, you need only implement an $O(n \log n)$-time construction algorithm, where n is the number of points. In contrast, for *RangeTree-Org*, you will need to implement *two* construction algorithms:

- *Contr-Naive*: the naive construction algorithm that takes $O(n \log^2 n)$ time, and
- *Contr-Sorted*: the $O(n \log n)$-time construction algorithm that takes advantage of sorting.

Task 2: Experiments on Construction Efficiency

In this task, you are asked to *design* an experiment to show the performance differences between the two construction algorithms of *RangeTree-Org*, i.e., *Contr-Naive* and *Contr-Sorted.* More specifically, you will need to:

- *describe* clearly your experimental design in detail, including the experiment setting, the datasets, parameters, etc.;
- *justify* your design and discuss what *anticipated results* (i.e., the performance) you want to demonstrate in the experiment;
- *conduct* the experiment you designed;
- *demonstrate* the experimental results in an appropriate way, e.g., using diagrams and/or tables.
- *analyse* the experiment results and discuss whether the results are as expected or not; if they are not as expected, explain the cause.

Task 3: Experiments on Query Efficiency

In this task, you will need to conduct *two* required experiments.

Query Efficiency Experiment 1, with Fixed n and Varying s. In this experiment, you will need to:

- By invoking *generate_point_set*(n), generate a set P of $n = 10^6$ points .
- Construct both *RangeTree-Org* and *RangeTree-FC* on P. In other words, these two range trees are constructed on the *same* set of points, P.
- Generate *five* query workloads, W_i, for $i = 1, \ldots, 5$:

- each W_i contains 100 range-reporting queries generated by invoking *generate_a_query*(s_i), where $s_1 = 1\% \cdot M$, $s_2 = 2\% \cdot M$, $s_3 = 5\% \cdot M$, $s_4 = 10\% \cdot M$ and $s_5 = 20\% \cdot M$; recall that $M = 10^6$, a million.

- For each workload W_i:
  - perform the 100 range reporting queries W_i on *RangeTree-Org* and *RangeTree-FC*, respectively. Observe that the queries performed on both of the two range trees are exactly the same;
  - take*average query time* over all these 100 queries on the two range trees, respectively.

Query Efficiency Experiment 2, with Fixed s and Varying n. In this experiment, you will need to:

- By invoking *generate_a_query*(s) with $s = 5\% \cdot M$, generate a query workload, W, of 100 range-reporting queries.
- Generate *ten* sets of points, P_i, for $i = 1, 2, \ldots, 10$, by invoking *generate_point_set*(n_i), where $n_i = 2^i \cdot 10^3$.
- For point set P_i:
  - construct both *RangeTree-Org* and *RangeTree-FC* on (the same) P_i;
  - perform the 100 range reporting queries on W with *RangeTree-Org* and *RangeTree-FC*, respectively;
  - take *average query time* over all these 100 queries on the two range trees, respectively.

Task 4: Report Writing

You will need to write and submit a report on the experimental results. More specifically, the report should contain the following components:

- **[1 mark out of 20]** *Experiment Environment.*
  Explain clearly the necessary information for others to *reproduce* your experiment, such as: the CPU frequency, memory, operating system, programming language, compiler (if applicable), etc.

- **[1 mark out of 20]** *Data Point and Query Generation.*
  Describe how you obtain or generate the data (i.e., data points and queries). Although in the above specification, certain details of their generation are already given, you will need to describe them in *your own words.*

- **[8 marks out of 20]** *Experiments on Construction Efficiency.*
  - **[1 marks]** Description of your experimental design.
  - **[2 marks]** Justification of the design: why did you choose to design the experiment in the way you did?
  - **[2 marks]** Discussion on the anticipated experimental results, with reference to the theoretical properties.
  - **[1 mark]** Experimental results demonstration: you will need show the results in a way that addresses the previous items.

  - **[2 marks]** Analysis on the results: compare with your anticipation and explain if the results do not match the anticipation.

- **[8 marks out of 20]** *Required Experiments on Query Efficiency.*
  Each of the two experiments is worth **4 marks**. In the report, you will need to:

  - **[1 mark]** discuss what experimental results you anticipate, according to the theoretical properties;
  - **[1 mark]** demonstrate the experiment results in a way that addresses the above item;
  - **[2 marks]** analyse the results, compare with your anticipation and explain if the results do not match the anticipation.

- **[2 marks out of 20]** *Conclusion.*
  Summarize and conclude your findings in these experiments. For example, in which situations would you choose a particular algorithm and data structure.

Marking Scheme. The marks for each component in the report are as shown. The grading will be based on the *clarity* and *rigorousness* of your description or analysis for each part. Others should be able reproduce the whole experimental study with your report and obtain similar results.

Submissions. You should lodge your submission for Assignment 2 via the LMS (i.e., Canvas). *You must identify yourself in* ***each*** *of your source files and the report.* Poor-quality scans of solutions written or printed on paper will *not* be accepted. There are scanning apps for smartphones, etc. Solutions generated directly on a computer are of course acceptable. Submit *two* files:

- A *report.pdf* file comprising your report for the experimental study.
- A *code.zip* file containing all your sources files of the implementations for the experiments.

Do not include the testing files, as these might be large. **REPEAT: DO NOT INCLUDE TESTING FILES!** It is very important, so that you can justify ownership of your work, that you detail your contributions in comments in your code, and in your report.

Administrative Issues.

When is late? What do I do if I am late? The due date and time are printed on the front of this document. The lateness policy is on the handout provided at the first lecture. As a reminder, the late penalty for non-exam assessment is *two* marks per day (or part thereof) overdue. Requests for extensions or adjustment must follow the University policy (the Melbourne School of Engineering "owns" this subject), including the requirement for appropriate evidence.

Late submissions should also be lodged via the LMS, but, as a courtesy, please also email both the two lecturers (Junhao Gan and Tony Wirth) when you submit late. If you make both on-time and late submissions, please consult the subject coordinators as soon as possible to determine which submission will be assessed.

Individual work. You are reminded that your submission for this Assignment is to be your own individual work. Students are expected to be familiar with and to observe the University's Academic Integrity policy `http://academicintegrity.unimelb.edu.au/`. For the purpose of ensuring academic integrity, every submission attempt by a student may be inspected, regardless of the number of attempts made.

Students who allow other students access to their work run the risk of also being penalized, even if they themselves are sole authors of the submission in question. **By submitting your work electronically, you are declaring that this is your own work.** Automated similarity checking software may be used to compare submissions.

You may re-use code provided by the teaching staff, and you may refer to resources on the Web or in published or similar sources. Indeed, you are encouraged to read beyond the standard teaching materials. However, *all* such sources *must* be cited fully and, apart from code provided by the teaching staff, you must *not* copy code.

Finally. *Despite all these stern words, we are here to help!* There is information about getting help in this subject on the LMS pages. Frequently asked questions about the Assignment will be answered in the LMS discussion group.