Shaohua Lu (001514123)

**Program Structures & Algorithms**

**Spring 2021**

**Assignment**

* **Task**

**Merge Sort Improvement(Cutoff/ Insurance/ NoCopy)**

* **Output**

**Time Record for merge sort improvement of Insurance and NoCopy In nano second**

Instrumenting helper for insertion sort with 128 elements

average time random\_ff: 12689

Instrumenting helper for insertion sort with 128 elements

average time random\_ft: 11304

Instrumenting helper for insertion sort with 128 elements

average time random\_tf: 15209

Instrumenting helper for insertion sort with 128 elements

average time random\_tt: 18875

Instrumenting helper for insertion sort with 256 elements

partial sorted average time partialsorted\_ff: 28990

Instrumenting helper for insertion sort with 256 elements

partial sorted average time partialsorted\_ft: 40948

Instrumenting helper for insertion sort with 256 elements

partial sorted average time partialsorted\_tf: 24266

Instrumenting helper for insertion sort with 256 elements

partial sorted average time partialsorted\_tt: 85563

**SortBenchMark Update**

2021-04-02 14:51:59 DEBUG Config - Config.get(huskysort, version) = null

2021-04-02 14:51:59 INFO SortBenchmark - SortBenchmark.main: null with word counts: []

2021-04-02 14:51:59 WARN SortBenchmark - No word counts specified on the command line

2021-04-02 14:51:59 INFO Benchmark\_Timer - Begin run: intArraysorter with 100 runs

2021-04-02 14:52:00 INFO TimeLogger - Raw time per run (mSec): 6.90

2021-04-02 14:52:00 INFO TimeLogger - Normalized time per run (n log n): .76

2021-04-02 14:52:00 INFO Benchmark\_Timer - Begin run: integerArraysorter with 100 runs

2021-04-02 14:52:02 INFO TimeLogger - Raw time per run (mSec): 17.72

2021-04-02 14:52:02 INFO TimeLogger - Normalized time per run (n log n): 1.95

2021-04-02 14:52:02 INFO SortBenchmark - Beginning String sorts

2021-04-02 14:52:02 INFO SortBenchmarkHelper - Testing with words: 2,998 from 3000-common-words.txt

2021-04-02 14:52:02 INFO SortBenchmark - Testing pure sorts with 5,000 runs of sorting 4,000 words

2021-04-02 14:52:02 DEBUG Config - Config.get(benchmarkstringsorters, puresystemsort) = false

2021-04-02 14:52:02 INFO SortBenchmarkHelper - Testing with words: 24,017 from zho-simp-tw\_web\_2014\_10K-sentences.txt

2021-04-02 14:52:02 INFO SortBenchmark - Testing pure sorts with 1,000 runs of sorting 5,000 words

2021-04-02 14:52:02 DEBUG Config - Config.get(helper, instrument) = true

2021-04-02 14:52:02 INFO SortBenchmarkHelper - Testing with words: 24,017 from zho-simp-tw\_web\_2014\_10K-sentences.txt

2021-04-02 14:52:02 INFO SortBenchmark - Testing with 1,000 runs of sorting 5,000 words and instrumented

2021-04-02 14:52:02 DEBUG Config - Config.get(benchmarkstringsorters, mergesort) = true

2021-04-02 14:52:02 DEBUG Config - Config.get(helper, seed) =

2021-04-02 14:52:02 DEBUG Config - Config.get(instrumenting, copies) = true

2021-04-02 14:52:02 DEBUG Config - Config.get(instrumenting, swaps) = true

2021-04-02 14:52:02 DEBUG Config - Config.get(instrumenting, compares) = true

2021-04-02 14:52:02 DEBUG Config - Config.get(instrumenting, inversions) = 0

2021-04-02 14:52:02 DEBUG Config - Config.get(instrumenting, fixes) = true

2021-04-02 14:52:02 DEBUG Config - Config.get(helper, cutoff) =

2021-04-02 14:52:02 INFO SorterBenchmark - run: sort 5,000 elements using SorterBenchmark on class java.lang.String from 24,017 total elements and 1,000 runs using sorter: MergeSort

2021-04-02 14:52:02 INFO Benchmark\_Timer - Begin run: Instrumenting helper for MergeSort with 5,000 elements with 1,000 runs

2021-04-02 14:52:04 INFO TimeLogger - Raw time per run (mSec): 1.14

2021-04-02 14:52:04 INFO TimeLogger - Normalized time per run (n log n): 3.51

2021-04-02 14:52:04 DEBUG InstrumentedHelper - Closing Helper: MergeSort with statPack: StatPack {copies: 100,000; inversions: <unset>; swaps: mean=4,878; stdDev=64; fixes: mean=6,246,153; stdDev=58,698; compares: mean=58,439; stdDev=78}

2021-04-02 14:52:04 DEBUG Config - Config.get(benchmarkstringsorters, insurance) = false

2021-04-02 14:52:04 DEBUG Config - Config.get(benchmarkstringsorters, noCopy) = false

2021-04-02 14:52:04 DEBUG Config - Config.get(benchmarkstringsorters, quicksort3way) = false

2021-04-02 14:52:04 DEBUG Config - Config.get(benchmarkstringsorters, quicksort) = true

2021-04-02 14:52:04 INFO SorterBenchmark - run: sort 5,000 elements using SorterBenchmark on class java.lang.String from 24,017 total elements and 1,000 runs using sorter: QuickSort dual pivot

2021-04-02 14:52:04 INFO Benchmark\_Timer - Begin run: Instrumenting helper for QuickSort dual pivot with 5,000 elements with 1,000 runs

2021-04-02 14:53:44 INFO TimeLogger - Raw time per run (mSec): 99.22

2021-04-02 14:53:44 INFO TimeLogger - Normalized time per run (n log n): 304.93

2021-04-02 14:53:44 DEBUG InstrumentedHelper - Closing Helper: QuickSort dual pivot with statPack: StatPack {copies: 0; inversions: <unset>; swaps: mean=29,096; stdDev=1,942; fixes: mean=7,042,596; stdDev=785,880; compares: mean=72,834; stdDev=3,093}

2021-04-02 14:53:44 DEBUG Config - Config.get(benchmarkstringsorters, introsort) = true

2021-04-02 14:53:44 INFO SorterBenchmark - run: sort 5,000 elements using SorterBenchmark on class java.lang.String from 24,017 total elements and 1,000 runs using sorter: Intro sort

2021-04-02 14:53:44 INFO Benchmark\_Timer - Begin run: Instrumenting helper for Intro sort with 5,000 elements with 1,000 runs

2021-04-02 14:55:24 INFO TimeLogger - Raw time per run (mSec): 98.33

2021-04-02 14:55:24 INFO TimeLogger - Normalized time per run (n log n): 302.17

2021-04-02 14:55:24 DEBUG InstrumentedHelper - Closing Helper: Intro sort with statPack: StatPack {copies: 0; inversions: <unset>; swaps: mean=33,444; stdDev=2,007; fixes: mean=7,063,533; stdDev=804,655; compares: mean=76,072; stdDev=3,124}

2021-04-02 14:55:24 DEBUG Config - Config.get(benchmarkstringsorters, insertionsort) = false

2021-04-02 14:55:24 INFO SortBenchmark - Beginning LocalDateTime sorts

2021-04-02 14:55:24 DEBUG Config - Config.get(benchmarkdatesorters, timsort) = true

2021-04-02 14:55:24 INFO Benchmark\_Timer - Begin run: Sort LocalDateTimes using Arrays::sort (TimSort) with 100 runs

2021-04-02 14:55:28 INFO SortBenchmark - 28.16ms

2021-04-02 14:55:28 INFO Benchmark\_Timer - Begin run: Repeat Sort LocalDateTimes using timSort::mutatingSort with 100 runs

2021-04-02 14:55:32 INFO SortBenchmark - 28.14ms

2021-04-02 14:55:32 INFO SorterBenchmark - run: sort 100,000 elements using SorterBenchmark on class java.time.LocalDateTime from 100,000 total elements and 100 runs using sorter: Timsort

2021-04-02 14:55:32 INFO Benchmark\_Timer - Begin run: Helper for Timsort with 0 elements with 100 runs

2021-04-02 14:55:36 INFO TimeLogger - Raw time per run (mSec): 32.87

2021-04-02 14:55:36 INFO TimeLogger - Normalized time per run (n log n): 3.61

* **Relationship Conclusion:**

With random array, when only cutoff was implemented, the time was 8029 in nano second, with only insurance, the time was 7507, 7593 with only no copy implemented. I was hoping to see that when insurance and no Copy was both true, the time would be the shortest but turns out it takes more time than run them individually.

With partially sorted array, the first 128 number were in order and the next 128 are randomly picked. With only cutoff, the time performed well, 15899 in nano second. No copy performed badly and almost double the time than cutoff. Insurance performed well with only 500 nanosecond difference than cutoff. Insurance and No Copy performed bad when they are both running at the same time. 39511 is the largest number we get.

* **Evidence to support the conclusion:**

|  |  |
| --- | --- |
| Random Array | Time in Nano sec |
| random\_ff | 8029 |
| random\_ft | 7593 |
| random\_tf | 7507 |
| random\_tt | 7795 |
| partial order Random Array | Time in Nano sec |
| partial order random\_ff | 15899 |
| partial order random\_ft | 30010 |
| partial order random\_tf | 16336 |
| partial order random\_tt | 39511 |

* **Graphical representation:**
* **Unit tests result:**

Graphical user interface, text, application, email

Description automatically generated