ACM40660-Scientific-Programming-Concepts Assignment 2

Ian Towey

04128591

March 19, 2018

Contents

1 Running Tests/Benchmarks

To run the application

```
cd code
make clean && make && clear
./assignment2 $TEST_IDX
```

TEST_IDX is integer in range [16]

- 1) randomarray_tests
- 2) bubblesort_tests
- 3) bubblesort_input_tests
- 4) search_tests
- 5) chopsearch_tests
- 6) benchmark_tests

Or execute shell scripts

- sh randomarray_tests.sh
- sh bubblesort_tests.sh
- sh bubblesort_input_tests.sh
- sh search_tests.sh
- sh chopsearch_tests.sh
- sh benchmark_tests.sh

2 Results

Run benchmark tests

```
cd code
sh benchmark_tests.sh
```

2.1 Search

```
**************************************************************************
* search benchmark test
**************************************************************************

--------------------------------------------------------------------------

    Parameters:
        'n'(length of array)                :   2000
        'max'(max random value)             :   10000
```

```
        's'(value to search array for)          :   10
        'mult'(number of iterations of sort to compute) :   1000000


------------------------------------------------------------------------
total time taken    : 3117319.000 (CLOCKS_PER_SEC)
avg time        : 3.117319000 (CLOCKS_PER_SEC)
------------------------------------------------------------------------
------------------------------------------------------------------------

------------------------------------------------------------------------

    Parameters:
        'n'(length of array)                :   2000
        'max'(max random value)             :   10000
        's'(value to search array for)          :   5000
        'mult'(number of iterations of sort to compute) :   1000000


------------------------------------------------------------------------
total time taken    : 3519216.000 (CLOCKS_PER_SEC)
avg time        : 3.519216000 (CLOCKS_PER_SEC)
------------------------------------------------------------------------
------------------------------------------------------------------------

------------------------------------------------------------------------

    Parameters:
        'n'(length of array)                :   2000
        'max'(max random value)             :   10000
        's'(value to search array for)          :   9000
        'mult'(number of iterations of sort to compute) :   1000000


------------------------------------------------------------------------
total time taken    : 3937326.000 (CLOCKS_PER_SEC)
avg time        : 3.937326000 (CLOCKS_PER_SEC)
------------------------------------------------------------------------
************************************************************************
```

2.2 Chopsearch

```
************************************************************************
* chopsearch benchmark test
************************************************************************

------------------------------------------------------------------------

    Parameters:
        'n'(length of array)                :   2000
```

```
        'max'(max random value)             :   10000
        's'(value to search array for)          :   10
        'mult'(number of iterations of sort to compute) :   1000000


-------------------------------------------------------------------------
total time taken    : 3181894.000 (CLOCKS_PER_SEC)
avg time        : 3.181894000 (CLOCKS_PER_SEC)
-------------------------------------------------------------------------
-------------------------------------------------------------------------

-------------------------------------------------------------------------

    Parameters:
        'n'(length of array)                :   2000
        'max'(max random value)             :   10000
        's'(value to search array for)          :   5000
        'mult'(number of iterations of sort to compute) :   1000000


-------------------------------------------------------------------------
total time taken    : 3169850.000 (CLOCKS_PER_SEC)
avg time        : 3.169850000 (CLOCKS_PER_SEC)
-------------------------------------------------------------------------
-------------------------------------------------------------------------

-------------------------------------------------------------------------

    Parameters:
        'n'(length of array)                :   2000
        'max'(max random value)             :   10000
        's'(value to search array for)          :   9000
        'mult'(number of iterations of sort to compute) :   1000000


-------------------------------------------------------------------------
total time taken    : 3165580.000 (CLOCKS_PER_SEC)
avg time        : 3.165580000 (CLOCKS_PER_SEC)
-------------------------------------------------------------------------
*************************************************************************
```

2.3 Performance

Search seems to have more variation in the benchmark. This is probably due to the position of the search target in the sorted array. The closer the search value is to the max possiblr random value in the array, the more time it takes to find (or not find) the search value.

When the search target is the value 10, this value will probably be around position 2 of the *sorted* array of length 2000, therfore average time taken is 3.117319 clock cycle per seconds.

When the search target is the value 5000, this value will probably be around position 1000 of the *sorted* array of length 2000, therfore average time taken is 3.519216 clock cycle per seconds.

When the search target is the value 9000, this value will probably be around position 1800 of the *sorted* array of length 2000, therfore average time taken is 3.937326 clock cycle per seconds.

This concerning if the array size is extremely large, as the performance of the algorithm will degrade with value for the search target

Chopsearch has very little variation in the benchmark regardless of the position of the search target. This is due to the chopsearch algorithm feature of recursively checking if the search value is in the upper or lower part of the array and at each iteration dicounts half of the array.

When the search target is the value $10, 5000, 9000$, the average time taken is $3.181894, 3.16985, 3.16558$ clock cycle per seconds respectively. Chopsearch exhibits very little variation regardless of search value. This is a good trait for the search algorithm, as we can expect constant performance that can be estimated very accurately regardless of array length and search value.

Based on this analysis *Chopsearch* is a better search algorithm then *Search*.

3 Complexity

Algorithm Complexity Performance			
	Best	Worst	Average
Search	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n/2)$
Chopsearch	$\mathcal{O}(1)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$