# Question 3 – Checklist Table

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Tasks | Mark | Checklist  (Yes/no) and person(s) to work on this task |
| 1 | Quick-select algorithm for searching the *k*-th element.  - Correct implementation of searching *k*-th element using quick-select algorithm.  - Output the intermediate results for a random array of 10 elements for tutor to inspect the correctness of the algorithms. | 4 |  |
| 2 | Test the Quick-select algorithm with  - Different array sizes (10,000, 100,000, 1,000,000, etc.) that can show significant results.  - Different pivot (random pivot vs fixed pivot).  - Different cases (e.g. best, average, and worst). | 4 |  |
| 3 | Merge-sort algorithm for searching the *k*-th element.  - Correct implementation of searching *k*-th element using merge-sort algorithm.  - Output the intermediate results for a random array of 10 elements for tutor to inspect the correctness of algorithms. | 4 |  |
| 4 | Test the Merge-sort algorithm with  - Different array sizes (10,000, 100,000, 1,000,000, etc.) that can show significant results.  - Test the Merge-sort algorithm with different cases (e.g. best, average, and worst). | 4 |  |
| 5 | - Include the above experiment results that can be used to perform a comparative analysis (such as drawing the graphs for comparison) between the two algorithms (Quick-select & Merge-sort) in the report.  - Conclude your findings in the report. | 4 |  |

# Quick-select algorithm

## Array size test (k = 80)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Array size, *n* | 10,000 | 100,000 | 1,000,000 | 5,000,000 |
| Attempt 1 (s) | 0.000999 | 0.001983 | 0.033009 | 0.170038 |
| Attempt 2 (s) | 0.000992 | 0.002001 | 0.033166 | 0.179039 |
| Attempt 3 (s) | 0.001001 | 0.002001 | 0.034009 | 0.182043 |
| Attempt 4 (s) | 0.000984 | 0.003 | 0.036008 | 0.183031 |
| Attempt 5 (s) | 0.001001 | 0.002 | 0.035008 | 0.17803 |
| Average Duration (s) | 0.0009954 | 0.002197 | 0.03424 | 0.1784362 |

|  |  |  |  |
| --- | --- | --- | --- |
| Pivot selection | Random | Fixed | kth element |
| Attempt 1 (s) | 0.009415 | 0.010145 | 38 |
| Attempt 2 (s) | 0.009452 | 0 | 1638 |
| Attempt 3 (s) | 0 | 0 | 5690 |
| Attempt 4 (s) | 0 | 0.009825 | 7653 |
| Attempt 5 (s) | 0 | 0.009279 | 8654 |
| Average Duration (s) | 0.0037734 | 0.0058498 |  |

## Pivot test (same set of array size = 10000)

## Test of different cases (same set of array size = 10000)

|  |  |  |  |
| --- | --- | --- | --- |
| Case | Best | Average | Worst |
| Attempt 1 (s) | 0.001 | 0.001 (k=37) | 0.125004 |
| Attempt 2 (s) | 0 | 0.0004 (k=1238) | 0.124515 |
| Attempt 3 (s) | 0.000991 | 0.000661 (k=4632) | 0.124907 |
| Attempt 4 (s) | 0.000992 | 0 (k=9574) | 0.124537 |
| Attempt 5 (s) | 0 | 0.009212 (k=58) | 0.124378 |
| Average duration (s) | 0 | 0.0018424 | 0.1246682 |

Note:

# Merge-sort algorithm

## Array size test

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Array size, *n* | 10,000 | 100,000 | 1,000,000 | 5,000,000 | 10,000,000 |
| Duration (s) |  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Case | Best | Average | Worst |
| Duration (s) | 0.001s |  | 0.00095s |

## Test of different cases (same set of array)

# Comparative analysis

# Conclusion