FACULTY OF COMPUTING & INFORMATION TECHNOLOGY
كلية الحاسبات وتقنية المعلومات

CPCS-223
Theoretical and Empirical analysis study report

Group Members:

- **Mahmued Alardawi - 2135209**
- **Thamer S.Almalki - 2136185**
- **Abdulelah Ali Alturki - 2136110**
- **Abdulkareem Abdullah Al-Ghamdi - 2135037**

- **Submission Date: 1/29/2023**

Table of Contents

Theoretical Analysis of Insertion Sort (in terms of Big-O) 3
Theoretical Analysis of Selection Sort (in terms of Big-O) 4
Theoretical Analysis of Quick Sort (in terms of Big-O) 5
Experiment result (output) 7
N = 10 **7**
N = 100 **8**
N = 1000 **9**
N = 10000 **10**
N = 100000 **11**
Comparison of all three sorting algorithms 12
Conclusion 13

Theoretical Analysis of Insertion Sort (in terms of Big-O)

Pseudocode:

```
INSERTION SORT (A)
1   for j <- 2 to length[A]                                  n
2       do key <- A[j]                                       n-1
3       Insert A[j] into the sorted sequence A[1 . . j - 1].  /
4       i <- j – 1                                           n-1
5       while i > 0 and A[i] > key                           $\sum_{i=2}^{n} i$
6           do A[i + 1] <- A[i]                              $\sum_{i=2}^{n}(i-1)$
7           i <- i – 1                                       $\sum_{i=2}^{n}(i-1)$
8       A[i + 1] <- key                                      n-1
```

$T(n) = n + (n-1) + (n-1) + \sum_{i=2}^{n} i + \sum_{i=2}^{n}(i-1) + \sum_{i=2}^{n}(i-1) + (n-1)$

$T(n) = n + (n-1) + (n-1) + (n(n-1)/2 - 1) + (n(n-1)/2) + (n(n-1)/2) + (n-1)$

$T(n) = n + (n-1) + (n-1) + ((n^2-n)/2 - 1) + ((n^2-n)/2) + ((n^2-n)/2) + (n-1)$

$T(n) = n + 3(n-1) + 2((n^2-n)/2) + ((n^2-n)/2 - 1)$

$T(n) = n + 3n - 3 + 2n^2/2 - 2n/2 + n^2/2 - n/2 - 1$

$T(n) = n^2 + n^2/2 + 3n + n/2 - 4$

$T(n) = an^2 + bn + c$

$T(n) = 3/2\ n^2 + 7/2\ n - 4$

$T(n) = O(n^2)$

Theoretical Analysis of Selection Sort (in terms of Big-O)

Pseudocode:

SELECTION SORT (A)

```
1   for p <- 0 to length[A]                    n+1
2       do index <- p                          n
3       for i <- p + 1 to length[A]            n * n
4           do if A[i] < A[index]              n * n
5               index <- I                     1
6       SWAP A[index] with A[p]                n
```

$T(n) = (n+1) + n + n^2 + n^2 + 1 + n$

$T(n) = 2n^2 + 3n + 2$

$T(n) = O(n^2)$

Theoretical Analysis of Quick Sort (in terms of Big-O)

Pseudocode:

PARTITION (A, low, high)

```
1   l <- low - 1                     1
2   for r <- low to r < high         n+1
3       do if A[r] <= A[high]        n
4           l <- l + 1               n
5       SWAP A[l] with A[r]          n
6   SWAP A[l + 1] with A[high]       1
```

$T(n) = 1 + (n+1) + n + n + n + 1$

$T(n) = 4n + 2$

$T(n) = n$

QUICKSORT (A, low, high)

```
1   if low < high                        1
2       p <- partition(A, low, high)     n
3       Quicksort(A, low, p-1)           T(n) = n/2
4       Quicksort(A, p+1, high)          T(n) = n/2
```

$T(n) = 1 + T(n) + T(n/2) + T(n/2)$

$T(n)\ 2T(n/2) + n + 1$

$T(n) = 2T(n/2) + n$

Using backward substitution:

$T(n) = 2T(n/2) + n$

$T(n/2) = 4T(n/4) + 2n$

$T(n/4) = 8T(n/8) + 3n$

$T(n/8) = 16T(n/16) + 4n$

$T(n) = 2^k T(n/2^k) + kn$

$T(n) = 2^{\log_2 n} * T(1) + n*\log_2 n$

$T(n) = n + n*\log n$

$T(n) = O(n*\log n)$

Experiment result (output)

N = 10

```
Iteration 1 :

Insertion sort: Running time in nanoseconds: 3700

selection sort: Running time in nanoseconds: 5600

Quicksort sort: Running time in nanoseconds: 6000

Iteration 2 :

Insertion sort: Running time in nanoseconds: 2400

selection sort: Running time in nanoseconds: 2100

Quicksort sort: Running time in nanoseconds: 2100

Iteration 3 :

Insertion sort: Running time in nanoseconds: 2000

selection sort: Running time in nanoseconds: 2000

Quicksort sort: Running time in nanoseconds: 2300

Iteration 4 :

Insertion sort: Running time in nanoseconds: 2300

selection sort: Running time in nanoseconds: 2300

Quicksort sort: Running time in nanoseconds: 2600

Iteration 5 :

Insertion sort: Running time in nanoseconds: 2600

selection sort: Running time in nanoseconds: 2400

Quicksort sort: Running time in nanoseconds: 2700

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

Insertion average time: 2600

Selection average time: 2880

Quicksort average time: 3140

---------------------
```

N = 100

```
Iteration 1 :

Insertion sort: Running time in nanoseconds: 254900

selection sort: Running time in nanoseconds: 105200

Quicksort sort: Running time in nanoseconds: 26200

Iteration 2 :

Insertion sort: Running time in nanoseconds: 123900

selection sort: Running time in nanoseconds: 254800

Quicksort sort: Running time in nanoseconds: 112800

Iteration 3 :

Insertion sort: Running time in nanoseconds: 122100

selection sort: Running time in nanoseconds: 97100

Quicksort sort: Running time in nanoseconds: 173600

Iteration 4 :

Insertion sort: Running time in nanoseconds: 127000

selection sort: Running time in nanoseconds: 98100

Quicksort sort: Running time in nanoseconds: 185000

Iteration 5 :

Insertion sort: Running time in nanoseconds: 128600

selection sort: Running time in nanoseconds: 97500

Quicksort sort: Running time in nanoseconds: 5300

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

Insertion average time: 151300

Selection average time: 130540

Quicksort average time: 100580

---------------------
```

N = 1000

```
Iteration 1 :

Insertion sort: Running time in nanoseconds: 2274400

selection sort: Running time in nanoseconds: 2150800

Quicksort sort: Running time in nanoseconds: 70700

Iteration 2 :

Insertion sort: Running time in nanoseconds: 748000

selection sort: Running time in nanoseconds: 783400

Quicksort sort: Running time in nanoseconds: 93500

Iteration 3 :

Insertion sort: Running time in nanoseconds: 1747900

selection sort: Running time in nanoseconds: 2449900

Quicksort sort: Running time in nanoseconds: 67900

Iteration 4 :

Insertion sort: Running time in nanoseconds: 582200

selection sort: Running time in nanoseconds: 1357500

Quicksort sort: Running time in nanoseconds: 71800

Iteration 5 :

Insertion sort: Running time in nanoseconds: 634800

selection sort: Running time in nanoseconds: 788500

Quicksort sort: Running time in nanoseconds: 1167400

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

Insertion average time: 1197460

Selection average time: 1506020

Quicksort average time: 294260

---------------------
```

N = 10000

```
Iteration 1 :

Insertion sort: Running time in nanoseconds: 146488700

selection sort: Running time in nanoseconds: 89818300

Quicksort sort: Running time in nanoseconds: 1564200

Iteration 2 :

Insertion sort: Running time in nanoseconds: 88681900

selection sort: Running time in nanoseconds: 110279800

Quicksort sort: Running time in nanoseconds: 41985700

Iteration 3 :

Insertion sort: Running time in nanoseconds: 282635400

selection sort: Running time in nanoseconds: 61418000

Quicksort sort: Running time in nanoseconds: 617300

Iteration 4 :

Insertion sort: Running time in nanoseconds: 61004700

selection sort: Running time in nanoseconds: 89871900

Quicksort sort: Running time in nanoseconds: 1284000

Iteration 5 :

Insertion sort: Running time in nanoseconds: 81379800

selection sort: Running time in nanoseconds: 60713900

Quicksort sort: Running time in nanoseconds: 761700

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

Insertion average time: 132038100

Selection average time: 82420380

Quicksort average time: 9242580

---------------------
```

N = 100000

```
Iteration 1 :

Insertion sort: Running time in nanoseconds: 5668426800

selection sort: Running time in nanoseconds: 4281077600

Quicksort sort: Running time in nanoseconds: 6907800

Iteration 2 :

Insertion sort: Running time in nanoseconds: 5644101300

selection sort: Running time in nanoseconds: 4292260200

Quicksort sort: Running time in nanoseconds: 6949100

Iteration 3 :

Insertion sort: Running time in nanoseconds: 5684530700

selection sort: Running time in nanoseconds: 4274883500

Quicksort sort: Running time in nanoseconds: 7019300

Iteration 4 :

Insertion sort: Running time in nanoseconds: 5616283000

selection sort: Running time in nanoseconds: 2001562500

Quicksort sort: Running time in nanoseconds: 7166400

Iteration 5 :

Insertion sort: Running time in nanoseconds: 5678997900

selection sort: Running time in nanoseconds: 2010097600

Quicksort sort: Running time in nanoseconds: 7008200

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

Insertion average time: 5658467940

Selection average time: 3371976280

Quicksort average time: 7010160

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

Process finished with exit code 0
```

Comparison of all three sorting algorithms

Value of N	Run	Insertion Sort (Running time in Nanoseconds)	Selection Sort (Running time in Nanoseconds)	Quick Sort (Running time in Nanoseconds)
N=10	1	3700	5600	6000
	2	2400	2100	2100
	3	2000	2000	2300
	4	2300	2300	2600
	5	2600	2400	2700
Average Time		2600	2880	3140
N=100	1	254900	105200	26200
	2	123900	254800	112800
	3	122100	97100	173600
	4	127000	98100	185000
	5	128600	97500	5300
Average Time		151300	130540	100580
N=1000	1	2274400	2150800	70700
	2	748000	783400	93500
	3	1747900	2449900	67900
	4	582200	1357500	71800
	5	634800	788500	1167400
Average Time		1197460	1506020	294260
N=10000	1	146488700	89818300	1564200
	2	88681900	110279800	41985700
	3	282635400	61418000	617300
	4	61004700	89871900	1284000
	5	81379800	60713900	761700
Average Time		132038100	82420380	9242580
N=100000	1	5668426800	4281077600	6907800
	2	5644101300	4292260200	6949100
	3	5684530700	4274883500	7019300
	4	5616283000	2001562500	7166400
	5	5678997900	2010097600	7008200
Average Time		5658467940	3371976280	7010160

Conclusion

Inserion sort = O(n^2)

Selcetion sort = O(n^2)

Quicksort = O(n*log n)

This means Quick sort has the best time complexity.