**(3) Testing and Evaluation**

[**Test results**](#Testing_Evaluation_3)

**Test1:**

Insertion sort performs quite well, Quicksort performs slightly worse even if in theory its performance should be better than Insertion sort given the assumption that there is nothing special with the input array. This can be explained by fairly low number of elements in the array, in which case Insertion sort is superior.

**Test2:**

Insertion sort performs really well, this can be explained by small size of array and the fact that some parts of the array are in ascending order already and as array index increases the value of array element at that index is likely to increase aswell. Quicksort performs worse due to the small size of the array.

**Test3:**

Insertion sort performs worse than Quicksort due to the fact that size of input array is larger compared to previous tests, in addition the array does not appear to have any particular order.

**Test4:**

Insertion sort performs far better than quicksort due to the fact the input array is almost sorted and in that case insertion sort efficiency approaches it’s best case while quicksort approaches the worst case since it need more partitions to sort the array.

**(5) Further testing and evaluation**

[**Test results**](#Testing_Evaluation_5)

**a) In which of your test(s) does the new sorting algorithm perform better than insertion sort? State why you think this happens.**

In test 5. The reason why new sort performs so well is because there is a large number of duplicates in the input array and the reason why Insertion sort does not perform better is because the array does not appear to be in any specific order (is not nearly sorted).

**b) What is the worst case big Oh performance for the new sorting algorithm and when does this occur?**

The worst performance is O(N2), this occurs when there are no duplicates in the array. This way the algorithm has to find minimum value for each position and after that advances the pos index by one everytime.

**c) What is the best case big Oh performance for the new sorting algorithm and when does this occur?**

The best case performance is O(N), this occurs when all of the elements in the array are the same, this way the algorithm needs to go through all the elements to find minimum once, and then go through all of the elements one more time to swap them.

**(3) Testing and Evaluation Results**

---------------------------------Test 1---------------------------------

Insertion sort comparison count: 73

Insertion sort final array:

007 012 018 025 041 051 057 065 066 073 097 101 103 110 118

Quick sort comparison count: 83

Quick sort final array:

007 012 018 025 041 051 057 065 066 073 097 101 103 110 118

---------------------------------Test 2---------------------------------

Insertion sort comparison count: 30

Insertion sort final array:

005 014 018 025 041 043 059 065 066 073 097 103 110 112 118

Quick sort comparison count: 108

Quick sort final array:

005 014 018 025 041 043 059 065 066 073 097 103 110 112 118

---------------------------------Test 3---------------------------------

Insertion sort comparison count: 2840

Insertion sort final array:

001 001 001 001 002 003 005 006 009 011 020 020 020 020 021 023 028 029 030 033

033 043 045 046 046 046 046 046 055 055 055 056 057 061 063 066 069 069 070 071

072 073 076 077 079 080 081 083 085 088 091 092 094 094 095 099 101 101 103 105

106 107 110 113 118 125 127 128 129 136 138 140 143 144 147 148 150 150 153 156

158 169 169 169 170 171 175 178 180 184 184 184 184 189 190 193 198 199 199 199

Quick sort comparison count: 1008

Quick sort final array:

001 001 001 001 002 003 005 006 009 011 020 020 020 020 021 023 028 029 030 033

033 043 045 046 046 046 046 046 055 055 055 056 057 061 063 066 069 069 070 071

072 073 076 077 079 080 081 083 085 088 091 092 094 094 095 099 101 101 103 105

106 107 110 113 118 125 127 128 129 136 138 140 143 144 147 148 150 150 153 156

158 169 169 169 170 171 175 178 180 184 184 184 184 189 190 193 198 199 199 199

---------------------------------Test 4---------------------------------

Insertion sort comparison count: 363

Insertion sort final array:

001 001 001 003 003 003 003 005 005 006 009 009 009 011 021 021 021 028 028 029

030 033 033 033 041 041 041 041 041 043 043 054 055 055 056 056 056 057 060 060

063 066 067 069 069 070 071 073 074 074 079 080 080 080 080 085 091 091 094 094

094 094 094 095 099 101 101 101 101 101 103 105 107 115 115 115 115 118 127 127

136 136 138 147 148 148 148 150 152 152 152 163 169 169 170 170 170 180 190 190

Quick sort comparison count: 1563

Quick sort final array:

001 001 001 003 003 003 003 005 005 006 009 009 009 011 021 021 021 028 028 029

030 033 033 033 041 041 041 041 041 043 043 054 055 055 056 056 056 057 060 060

063 066 067 069 069 070 071 073 074 074 079 080 080 080 080 085 091 091 094 094

094 094 094 095 099 101 101 101 101 101 103 105 107 115 115 115 115 118 127 127

136 136 138 147 148 148 148 150 152 152 152 163 169 169 170 170 170 180 190 190

**(5) Further testing and evaluation results**

---------------------------------Test 3---------------------------------

New sort comparison count: 7388

New sort final array:

001 001 001 001 002 003 005 006 009 011 020 020 020 020 021 023 028 029 030 033

033 043 045 046 046 046 046 046 055 055 055 056 057 061 063 066 069 069 070 071

072 073 076 077 079 080 081 083 085 088 091 092 094 094 095 099 101 101 103 105

106 107 110 113 118 125 127 128 129 136 138 140 143 144 147 148 150 150 153 156

158 169 169 169 170 171 175 178 180 184 184 184 184 189 190 193 198 199 199 199

---------------------------------Test 4---------------------------------

New sort comparison count: 4971

New sort final array:

001 001 001 003 003 003 003 005 005 006 009 009 009 011 021 021 021 028 028 029

030 033 033 033 041 041 041 041 041 043 043 054 055 055 056 056 056 057 060 060

063 066 067 069 069 070 071 073 074 074 079 080 080 080 080 085 091 091 094 094

094 094 094 095 099 101 101 101 101 101 103 105 107 115 115 115 115 118 127 127

136 136 138 147 148 148 148 150 152 152 152 163 169 169 170 170 170 180 190 190

---------------------------------Test 5---------------------------------

New sort comparison count: 1682

New sort final array:

001 001 001 001 001 001 001 001 020 020 020 020 028 028 028 028 028 028 028 046

046 046 046 046 046 046 046 055 055 055 055 055 069 069 069 069 069 069 072 072

072 072 079 079 079 079 079 099 099 099 099 099 099 099 099 099 107 107 107 107

107 107 107 127 127 127 127 127 127 150 150 150 150 150 150 150 150 153 153 153

153 169 169 169 169 169 184 184 184 184 184 184 184 184 199 199 199 199 199 199

Insertion sort comparison count: 2728

Insertion sort final array:

001 001 001 001 001 001 001 001 020 020 020 020 028 028 028 028 028 028 028 046

046 046 046 046 046 046 046 055 055 055 055 055 069 069 069 069 069 069 072 072

072 072 079 079 079 079 079 099 099 099 099 099 099 099 099 099 107 107 107 107

107 107 107 127 127 127 127 127 127 150 150 150 150 150 150 150 150 153 153 153

153 169 169 169 169 169 184 184 184 184 184 184 184 184 199 199 199 199 199 199