Uppsala University

Algorithms and Data Sructures I

Assignment - 2

Group - 4

Amer Slaiman
Dalal Abdel Khafez
Elaf Salam
Rutha Tesfazghi Gilazghi

Sunday 3rd October, 2021

UPPSALA
UNIVERSITET

Contents

1 Introduction

In this report a comparison between the three sorting algorithms *Insertion Sort, Quick Sort* and *Heap Sort* will be presented. The implementation of the sorting algorithms will be attached in separate files.

2 Comparison

Running Time (seconds)			
Algorithm	Case 1	Case 2	Case 3
Insertion Sort	−0.000313997268677	−0.000503063201904	−0.000814914703369
Quick Sort	−0.000271081924438	−0.000243902206421	−0.00080394744873
Heap Sort	−0.000216960906982	−0.000236988067627	−0.000581979751587

Table 1: Running time for the implemented sorting algorithms for three randomly generated arrays.

From Table 1 we can conclude that Heap Sort has the most effective running time of the sorting algorithms in each of the cases. With Quick Sort ranked second to Heap Sort and Insertion Sort last with the least effective running time.

2.1 Insertion Sort

When dealing with small inputs, Insertion Sort is the fastest. However, it has quadratic time complexity for most of large inputs

Insertion Sort time complexity is $O(n^2)$

2.2 Quick Sort

Among the three presented algorithms Quick Sort is the "fastest" sorting technique for most inputs, because it has the best performance in the average case for most inputs. However, the algorithm becomes less efficient for larger inputs. The worst-case complexity of Quick Sort is quadratic which is worse than the worst-case complexity of Heap Sort and similar to Insertion Sort.

Quick Sort time complexity is $O(n^2)$

2.3 Heap Sort

It is proved that the time complexity of Heap Sort is $O(n \log n)$, and works in place with constant time ($O(n)$) extra memory. However, Heap Sort almost always runs in $\theta(n \log n)$ as it should traverse the input array even if the input is already sorted.

Heap Sort time complexity is $O(n \log n)$

2.4 Conclusion

Insertion sort can be the best sorting algorithm to use when the number of elements is small or when a big array is almost sorted, and just few elements are misplaced. It is the best because its best-case running time is the most effective among the other sorting algorithms. Quick Sort is generally assumed the "fastest" sorting algorithm because it has the best performance in the average case for most inputs. That means that if we have an average big list that averagely unsorted then Quick Sort is the best, or if we do not know any information about the data then Quick Sort can be a good guess. Heap Sort is considered to be in average slower than Quick Sort but Heap Sort worst-case running time is always $O(n \log n)$ and because of that Heap Sort is the best when it comes to the most complicated big arrays.