THE WORST CASE COMPLEXITY OF 
MC DIARMID AND REED’S 
VARIANT OF BOTTOM-UP-HEAP SORT IS LESS THAN 
\( n \log n + 1.1n \)

Ingo Wegener *
FB Informatik, LS II, Universität Dortmund 
Postfach 500 500, 4600 Dortmund 50, Fed. Rep. of Germany 

Abstract

BOTTOM-UP-HEAP SORT is a variant of HEAP SORT which beats on average even the clever variants of QUICK SORT, if \( n \) is not very small. Up to now, the worst case complexity of BOTTOM-UP-HEAP SORT can be estimated only by \( 1.5n \log n \). McDiarmid and Reed (1989) have presented a variant of BOTTOM-UP-HEAP SORT which needs extra storage for \( n \) bits. The worst case number of comparisons of this (almost internal) sorting algorithm is estimated by \( n \log n + 1.1n \). It is discussed how many comparisons can be saved on average.

1. INTRODUCTION

Sorting is one of the most fundamental problems in computer science. In this paper only general and sequential sorting algorithms are studied.

All results should be compared with the simple lower bound

\[ \log(n!) = n \log n - n \log e + \Theta(\log n) \approx n \log n - 1.442695n \]

for the worst and average case number of comparisons of general sorting algorithms. With respect to this lower bound sorting by merging and sorting by insertion and binary search are quite efficient. But MERGE SORT works efficiently only on an array of length \( 2n \) and INSERTION SORT uses \( \Theta(n^2) \) operations for the data transport.

HEAP SORT (Floyd (1964), Williams (1964)) needs \( 2n \log n \) comparisons. Because of the factor 2 HEAP SORT is in almost all cases less efficient than QUICK SORT. All versions of QUICK SORT are inefficient in the worst case but efficient in the average case. Let \( H(n) = 1 + \frac{1}{2} + \ldots + \frac{1}{n} \) be the \( n \)-th harmonic number, \( Q(n) \) the average number of comparisons of QUICK SORT and \( CQ(n) \) the average number of comparisons of the best-of-three variant of QUICK SORT called CLEVER QUICK SORT. Then (see e.g. Wegener (1990))

\[ Q(n) = 2(n + 1)H(n) - 4n \]

Supported in part by DFG grants We 1066-2/1 and Me 872-1/3
\[ CQ(n) = \frac{12}{7} (n+1)H(n-1) - \frac{477}{147} n + \frac{223}{147} + \frac{252}{147n} \]

\[ \approx 1.386294n \log n - 2.845569n \]

and for \( n \geq 6 \)

\[ \approx 1.188252n \log n - 2.255385n. \]

Because of these results HEAP SORT has been considered for a long time only for theoretical reasons. Carlsson (1987a) presented a new variant of HEAP SORT whose average and worst case complexity is \( n \log n + \Theta(n \log \log n) \). This algorithm does not beat CLEVER QUICK SORT on average for \( n \leq 10^{16} \). Another variant of HEAP SORT is called BOTTOM-UP-HEAP SORT (Carlsson (1987 b), Wegener (1990)). Its average case complexity cannot be computed because the heaps constructed in the selection phase are not random. Using realistic models one can argue that the average case number of comparisons equals \( n \log n + f(n)n \) where \( f(n) \in [0.355, 0.39] \) for \( n \geq 3000 \) depends on \( n \) (see Wegener (1990)). BOTTOM-UP-HEAP SORT is a general and internal sorting algorithm which beats on average QUICK SORT for \( n \geq 400 \) and CLEVER QUICK SORT for \( n \geq 16000 \). The worst case complexity can be bounded by \( 1.5n \log n - 0.4n \) (Wegener (1990)). In this situation a variant of BOTTOM-UP-HEAP SORT (McDiarmid and Reed (1989)) which we call MDR-HEAP SORT is interesting. MDR-HEAP SORT is not an internal sorting algorithm in the strong sense, since extra storage for \( n \) bits is necessary.

In Section 2 we present MDR-HEAP SORT and discuss some details of its implementation. McDiarmid and Reed (1989) have analysed the average case complexity of the heap creation phase of their algorithm and have left the average and worst case analysis of the whole algorithm as an open problem.

In Section 3 we prove that the worst case number of comparisons of MDR-HEAP SORT is remarkably small, it can be bounded by \( n \log n + 1.1n \).

No rigid analysis of the average case complexity of any HEAP SORT variant is known, since the heaps constructed during the selection phase are not random. In Section 4 we use realistic models to estimate the difference between the average and worst case complexity of MDR-HEAP SORT. We finish the paper with a comparison of MDR-HEAP SORT and QUICK SORT.

2. MDR-HEAP SORT

We assume that the reader is familiar with HEAP SORT. The same basic idea is used here.