Denni Algorithm An Enhanced Of SMS (Scan, Move and Sort) Algorithm

Denni Aprilsyah Lubis *, Opim Salim Sitompul, Marwan, Tulus, and M. Andri Budiman

*Planning and Development Agency Of Sibolga City Goverment Sibolga, Indonesia

denni.lubis@gmail.com

Abstract. Sorting has been a profound area for the algorithmic researchers, and many resources are invested to suggest a more working sorting algorithm. For this purpose many existing sorting algorithms were observed in terms of the efficiency of the algorithmic complexity. Efficient sorting is important to optimize the use of other algorithms that require sorted lists to work correctly. Sorting has been considered as a fundamental problem in the study of algorithms that due to many reasons namely, the necessary to sort information is inherent in many applications, algorithms often use sorting as a key subroutine, in algorithm design there are many essential techniques represented in the body of sorting algorithms, and many engineering issues come to the fore when implementing sorting algorithms. Many algorithms are very well known for sorting the unordered lists, and one of the well-known algorithms that make the process of sorting to be more economical and efficient is SMS (Scan, Move and Sort) algorithm, an enhancement of Quicksort invented Ramil Mansi in 2010. This paper presents a new sorting algorithm called Denni-algorithm. The Denni algorithm is considered as an enhancement on the SMS algorithm in average, and worst cases. The Denni algorithm is compared with the SMS algorithm and the results were promising.

1. Introduction

In 2010, Ramil Mansi presents the SMS algorithm (Scan, Move and Sort), which is an enhanced of Quicksort algorithm [8]. Because the SMS algorithm is an enhanced of the Quicksort algorithm, the SMS algorithm categorized in Exchange Sort algorithm. In this paper, we presented a new sorting algorithm called Denni-algorithm which is an enhanced of SMS algorithm. The study shows that the Denni algorithm is more efficient in terms of time complexity as compared to the SMS algorithm when dealing with a large size (n) of the input array, for the average and worst case.

2. Denni Algorithm

Denni algorithm consists of three procedures, Scan, Move, and Sort. The first procedure is (Scan), which scans the array and gives the values of the minimum of negative elements (minneg), the maximum of the negative elements (maxneg), the minimum of positive elements (minpos), the maximum of the positive elements (maxpos), the number of positive elements (NOP), and the number of negative elements (NON). Also, this procedure check whether the minimum value of negative equal to the maximum value of the negative and check whether the minimum value of positive equal to the
maximum value of positive, if both conditions are true, then the input array is already sorted, otherwise, calls the procedure (Move).

The second procedure (Move) creates four temporary arrays, FreqPosArray of size (NOP), FreqNegArray of size (NON), PosArray of size (maxpos+1), and NegArray of size (minneg+1), and then initializes the PosArray, the NegArray, and the FreqArray with the value (minneg-1) to denoting the indices that will be skipped in the next phase. Then, this procedure distributes the elements on the four arrays, the positive elements are saved in the PosArray using the element itself as its index, the negative elements are saved in the NegArray using the absolute value of the element itself as its index, the frequent negative elements are saved in the FreqNegArray using a variable (j) as index and the frequent negative elements are saved in the FreqPosArray using a variable (i) as index, the variable I and j started from zero and incremented by one. To initialize and distribute positive elements, the iteration starts from (minpos) step incremented by one up to (maxpos), and iteration to initialize and distributing the negative elements, starts from (maxneg) step incremented by one up to (minneg).

The third procedure (Sort) copies the elements of the NegArray starting from the last index with ignoring the values of (minneg-1). Then it copies the elements of the PosArray starting from the first index with ignoring the values of (minneg-1). Iteration for copying the negative elements of NegArray starting from (minneg) step decremented by one up to (maxneg) and iteration for copying the positive elements of PosArray starting from (minpos) step incremented by one up to (maxpos). The copying is done on the original input array with overwriting the original values with the sorted values. After each copying operation of an element from the NegArray and the PosArray to the original array, the procedure searches the FreqPosArray and the FreqNegArray and then copies all element that are equal to the element that copied in the last copying operation (current element). If the current element is positive, the search will be performed on FreqPosArray and if the current element is negative then the search will be performed on FreqNegArray.

3. The Pseudocode Of Denni Algorithm

The pseudocode of the Scan procedure:

```plaintext
procedure scan(array, size)
    if size > 1 then
        var a, maxpos, minpos, maxneg, minneg, NOP, NON
        maxpos:= 0
        minpos:=100000
        maxneg:= -100000
        minneg:= -1
        NOP:=0
        NON:=0
        for a:= 0 to size-1 do
            if array(a) >= 0 then
                NOP:= NOP+1
                if array(a) > maxpos then
                    maxpos := array(a)
                end if
            else
                NON:= NON+1
                if array(a) < minneg then
                    minneg := array(a)
                end if
            end if
        end for
```

3. The Pseudocode Of Denni Algorithm

The pseudocode of the Scan procedure:

```plaintext
procedure scan(array, size)
    if size > 1 then
        var a, maxpos, minpos, maxneg, minneg, NOP, NON
        maxpos:= 0
        minpos:=100000
        maxneg:= -100000
        minneg:= -1
        NOP:=0
        NON:=0
        for a:= 0 to size-1 do
            if array(a) >= 0 then
                NOP:= NOP+1
                if array(a) > maxpos then
                    maxpos := array(a)
                end if
            else
                NON:= NON+1
                if array(a) < minneg then
                    minneg := array(a)
                end if
            end if
        end for
```
if array(a) > maxneg then
    maxneg := array(a)
end if

end for
if maxneg < minneg then
    maxneg = minneg
if minpos > maxpos then
    minpos = maxpos
if ((minpos ≠ maxpos) and (minneg ≠ maxneg)) then
    Move(array, size, NOP, NON, maxpos, minpos, maxneg, minneg)
end if
end if
end procedure scan

The pseudocode of the Move procedure:
procedure move(array, size, NOP, NON, maxpos, minpos, maxneg, minneg)
var b, c, d, i, j
i := 0
j := 0
create a new array: FreqPosArray[NOP] and initialize by the value (minneg−1)
create a new array: FreqNegArray[NON] and initialize by the value (minneg−1)
if NOP > 0 then
    create a new array: PosArray[maxpos+1]
    for b := minpos to maxpos do
        PosArray(b) := minneg−1
    end for
end if
if NON > 0 then
    create a new array: NegArray[|minneg|+1]
    for c := |maxneg| to |minneg| do
        NegArray(c) := minneg−1
    end for
end if
for d := 0 to size−1 do
    if array(d) >= 0 then
        if PosArray(array(d)) == minneg−1 then
            PosArray(array(d)) := array(d)
        else
            FreqPosArray(i) := array(d)
i := i + 1
        end if
    else
        if NegArray(|array(d)|) == minneg−1 then
            NegArray(|array(d)|) := array(d)
        else
            FreqNegArray(j) := array(d)
j := j + 1
        end if
    end if
end for
Sort (array, NegArray, PosArray, FreqPosArray, FreqNegArray, NON, NOP, maxpos, minpos, maxneg, minneg, i, j)
end procedure move

The pseudocode of the Sort procedure:

procedure sort (array, NegArray, PosArray, FreqPosArray, FreqNegArray, NON, NOP, maxpos, minpos, maxneg, minneg, i, j)
var index, x, y
index := 0
if NON > 0 then
for x := |minneg| downto |maxneg| do
if NegArray(x) ≠ minneg - 1 then
array(index) := NegArray(x)
index := index + 1
for y := 0 to j do
if FreqNegArray(y) == array(index - 1) then
array(index) := FreqNegArray(y)
index := index + 1
end if
end for
end if
end for
end if
if NOP > 0 then
for x := minpos to maxpos do
if PosArray(x) ≠ minneg - 1 then
array(index) := PosArray(x)
index := index + 1
for y := 0 to i do
if FreqPosArray(y) == array(index - 1) then
array(index) := FreqPosArray(y)
index := index + 1
end if
end for
end if
end for
end if
end procedure sort

4. Comparison Denni Algorithm With SMS Algorithm

The main concept of the SMS algorithm and the Denni Algorithm exactly the same, which distributes the elements of the input array on additional temporary array. SMS Algorithm and Denni algorithm consists of three same procedures, Scan, Move and Sort. Differences of SMS algorithms and Denni Algorithm are as follows:

- The Scan procedure of the SMS algorithm useful only to get the values of the minimum (min), the maximum (max), the number of positive elements (NOP), and the number of negative elements of input array (NON) [8], however the Scan procedure of Denni algorithm useful to get minneg, maxneg, minpos, maxpos, NOP, and NON from the input array. The minneg, maxneg, minpos and maxpos useful to minimize the number of iteration on the Move procedure and Sort procedure in the SMS algorithm.
• The Move procedure of the SMS algorithm creates three temporary arrays, FreqArray of size (n) that contains all of frequent elements, PosArray of size (max+1) that contains positive elements, and NegArray of size (min+1) that contains positive elements [8], however, The Move procedure of the Denni algorithm creates four temporary arrays, FreqPosArray size of (NOP) that contains frequent positive elements, FreqNegArray of size (NON) that contains frequent negative elements, PosArray of size (maxpos+1) that contains positive elements, and NegArray of size (minneg+1) that contains positive elements.

• On the procedure Move of the SMS algorithm, iteration to initialize PosArray starts from 0 (zero) step incremented by one up to (max), and iteration to initialize NegArray starts from 0 (zero) step incremented by one up to (min), but on the procedure Move of the Denni algorithm, the number of iteration is minimized, where to initialize PosArray, iteration starts from (minpos) step incremented by one up to (maxpos), and iteration to initialize NegArray starting from (maxneg) step incremented by one up to (minneg).

• On the Sort procedure of the SMS algorithm, iteration for copying the elements of NegArray starting from (min) step decremented by one up to (maxneg) and iteration for copying the elements of PosArray start from (minpos) step increment by one up to (maxpos).

• After each copying operation of an element from the NegArray and the PosArray to the original array, in the procedure sort of the SMS algorithm searches the FreqArray and copies all element that are equal to the element that copied in the last copying operation (current element). This resulted will be done iteration (n) times each found elements that not equal to (min-1). However, on the Sort procedure of the Denni algorithm number of the iteration is minimized, if the current element is positive, the search will be performed on FreqPosArray and if the current element is negative then the search will be performed on FreqNegArray, the number of elements in FreqPosArray (NOP) and FreqNegArray (NON) smaller than the number of elements on FreqArray (n), because FreqPosArray and FreqNegArray is simplification of FreqArray, where FreqPosArray contains the positive elements of FreqArray and FreqNegArray contains the negative elements of FreqArray (NOP+NON=n).

• The Denni algorithm is best used to sort an array of distinct elements. In this case, the value of (fneg and fpos) will be equals to 1, and the algorithm takes O(n+(maxpos-minpos)+(minneg-|maxneg|)) time.

• Denni algorithm is faster than SMS algorithm when dealing with a large size (n) of the input array, and when the values (maxpos/max) and (minneg/|min|) are much less than the value (n). In the best case, the SMS algorithm and the Denni algorithm equally takes O(n) time, however, in the average and worst case the Denni Algorithm takes O(n+(fpos*(maxpos−minpos))/(|maxneg−|maxneg|)))) time, while the SMS algorithm takes O(n+f*(nilai maksimum−nilai minimum)) time [8].

• In the other hand, SMS algorithm and Denni algorithm equally needs O(n+max+|min|+2) additional space.

5. Case Study

To prove that the proposed algorithm is faster than SMS algorithm when dealing with a large size of the input array, In this case study, the Denni and the SMS algorithms have been applied to sort a list of (100.000) elements for (10) sets of data and each set of data tested (10) times. Denni algorithm and SMS algorithm is implemented using Software (Bloodshed Dev-C + + version 4.9.9.2 GNU General Public License) and built-in functions (clock ()) is used to measure the elapsed time of both algorithms on the same computer using the same data set. The test results are as follows.
Table 1. Execution time for SMS and Denni algorithms for 100,000 elements of Input Array.

| Set Of Data | Execution Time SMS Algorithm (ms) | Standart Deviation SMS Algorithm (ms) | Execution Time Denni Algorithm (ms) | Standart Deviation Denni Algorithm (ms) |
|-------------|-----------------------------------|--------------------------------------|-------------------------------------|----------------------------------------|
| 1           | 8427                              | 51                                   | 4303                                | 62                                     |
| 2           | 7892                              | 41                                   | 3995                                | 52                                     |
| 3           | 7847                              | 50                                   | 4006                                | 49                                     |
| 4           | 7862                              | 52                                   | 4033                                | 67                                     |
| 5           | 7856                              | 47                                   | 4016                                | 46                                     |
| 6           | 7812                              | 54                                   | 4010                                | 66                                     |
| 7           | 7836                              | 56                                   | 3986                                | 81                                     |
| 8           | 7863                              | 71                                   | 4001                                | 54                                     |
| 9           | 7889                              | 65                                   | 4049                                | 71                                     |
| 10          | 7819                              | 73                                   | 3994                                | 71                                     |

6. Conclusions

In this paper presented a new sorting algorithm an enhancement of SMS algorithm is called Denni algorithm. Denni algorithm improve the efficiency of time complexity of SMS algorithm by minimizing iteration that occurs on the Move and Sort procedure. SMS algorithm and Denni algorithm equally takes O(n) time complexity in the best case. But in the average and worst case Denni algorithm is faster than SMS algorithm. In the average and worst case SMS algorithm takes O(n+f*(max+|min|)) time complexity, while Denni algorithm takes O(n+(fpos*(maxpos-minpos))+(fneg*(|minneg|- |maxneg|))) time complexity, where fpos is the number of frequent positive elements and fneg is the number of frequent negative elements.

In terms of space complexity, SMS algorithm and Denni Algorithm equally requires O(n+max+|min|+2) additional space to sort n elements.

References

[1] Alnihoud, J. & Mansi, R. Januari 2010. An Enhancement of Major Sorting Algorithms. The International Arab Journal of Information Technology. vol. 7. No. 1.
[2] Cormen, T., Leiserson, C., Rivest R. & Stein, C. 2001. Introduction to Algorithms. McGraw Hill.
[3] Dean, C. 2006. A Simple Expected Running Time Analysis for Randomized Divide and Conquer Algorithms. Computer Journal of Discrete Applied Mathematics. vol. 154. no. 1. pp. 1-5.
[4] Deependra, Kr. Dwivedi. 2011. Comparison Analysis of Best Sorting Algorithms. VSRD-JCSIT. vol. 1 (4). Pp. 261-267.
[5] Hoare, R. 1962. Quicksort. The Computer Journal. pp. 10-15.
[6] Kruse, R. & Ryba, A. 1999. Data Structures and Program Design in C++, Prentice Hall.
[7] Knuth, E. 1998. The Art of Computer Programming Sorting and Searching. 2nd edition. Addison-Wesley.
[8] Rami Mansi. April 2010. Enhanced Quicksort Algorithm. The International Arab Journal of Information Technology. vol. 7 no. 2.