Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 8, August 2015, pg.188 – 194

RESEARCH ARTICLE

Priority Based Sorting Algorithm for Multiple Attributes

Kalpana Purbiya L[1], Mr. B.L. Pal[2]

[1]Department of Computer Science & Engineering, Mewar University, India
[2]Department of Computer Science & Engineering, Mewar University, India
[1] kalpanapurbia@ymail.com, [2] bpal@mewaruniversity.co.in

Abstract*— *As we know that there are many operations performed in Data Structures. One of the most important operation of Data Structure is sorting that makes us to search, arrange or locate the record easily and efficiently. There are many sorting algorithms having different-different efficiencies based on the number of inputs. Sorting can also be performed on priority basis which displays required record first and later on remaining records. But we don't get any changes in complexity. Sorting can also be done in order to display only required records on the basis of choice of attribute's value as par user's choice. If we give one attribute value, it will take less time. And if we give multiple attributes, it will take much more less time with respect to single attribute because number of records decreases with respect to increasing number of attributes generally. That means the efficiency of searching a record from sorted records increases when we sort on the basis of priorities and attribute's value. If we implement this process, we can increase our main-power performance as well as tool or application because it will take less time.

Keywords*— *Sorting, Priority Queue, Required Data, Complexity, Attribute Value

I. INTRODUCTION

One of the operation in data structure which plays a foremost role in processing of data is "Sorting" which organizes the records in some logical order, i.e. either in ascending order or in descending order [1]. We can't say that one sorting algorithm is better or worse than other sorting algorithm because each sorting algorithm has their own pros and cons. For example, if we want to sort small list of records, then bubble sort is the efficient algorithm whereas if want to sort long list of records, then quick sort is the efficient algorithm.

The main advantage of sorting application is that it requires less time to search for a particular record compared to unsorted list of records [2]. There have been many attempts made to analyse and reduce the complexity of various algorithm. Also improved sorting algorithms have been proposed to get better efficiency in terms of time and memory. The performance of each sorting algorithm is based on the data being sorted and the machine used for sorting [3].

In general, any sorting algorithm performs two operations: first compare the two elements and second swap according to desired order of a user. These two operations proceed over and over until the entire list of record is sorted [4].

The performance of various sorting algorithm can be measured by few dimensions: the execution time, and the space required for the algorithm [5].

II. LITERATURE SURVEY

Since the dawn of computing, the sorting problem has attracted a great deal of research, perhaps due to the complexity of solving it efficiently despite its simple, familiar statement [6].

Tim Bell, "Sorting Algorithms as Special Cases of a Priority Queue Sort", 2011 [7], sorted records by linking most sorting algorithms as special cases of a PQ sort and then incorporating some connections, they were able to show multiple relationships between sorting algorithms, and in principle, each have been derived from another by exploring a generalization or extreme case.

Ahmed M. Aliyu, "A Comparative Analysis of Sorting Algorithms on Integer and Character Arrays", 2013 [8], attempted to compare the performance of two sorting algorithm: Selection Sort and Quick Sort, with the aim of comparing their speed on integer arrays and string arrays. Analysis of these two sorting algorithms proved that integer array have faster CPU time than string arrays although both have the upper bound running time O (n2).

Ghiath Al Aqel, "A New Priority-Sort Based Optimization Algorithm Integrated Process Planning and Scheduling", 2013 [9], stated that sorting algorithm based on priorities can effectively solve IPPS problems.

Jehad Hammad, "A Comparative Study between Various Sorting Algorithms", March 2015 [10], proved that there is no specific algorithm that can solve any problem in absolute.

III. PROBLEM STATEMENT

The sorting of records can be done on the basis of priority which we want to display the record first and then remaining list of records. But the problem arises here is that they are displaying all the records on the basis of priorities, which consumes same amount of time. We are proposing an algorithm which will display required records only as par user's choice on the basis of priorities and attributes. So, that we can save time.

IV. OBJECTIVE

The main objectives of the proposed algorithm are:

- Propose priority based sorting algorithm.
- Apply priority based sorting on multiple attributes of records.
- Save time on the basis of number of operations.

V. PROPOSED ALGORITHM

Step 1: Insert all the records and assign priority to each record.
Step 2: Sort the records on the basis of attribute values.
Step 3: Display sorted records of step 2.
Step 4: Exit.

VI. THEORETICAL AND EXPERIMENTAL WORK

A. Priority Queue

Priority queues is an abstract data type in data structure. Unlike the simple queue that inserts and removes records in a fixed order (First-In-First-Out), priority queue is assigned a priority for each records represented by an integer value so that the record can be removed from queue on the basis of highest priority represented by the minimal integer value assigned.

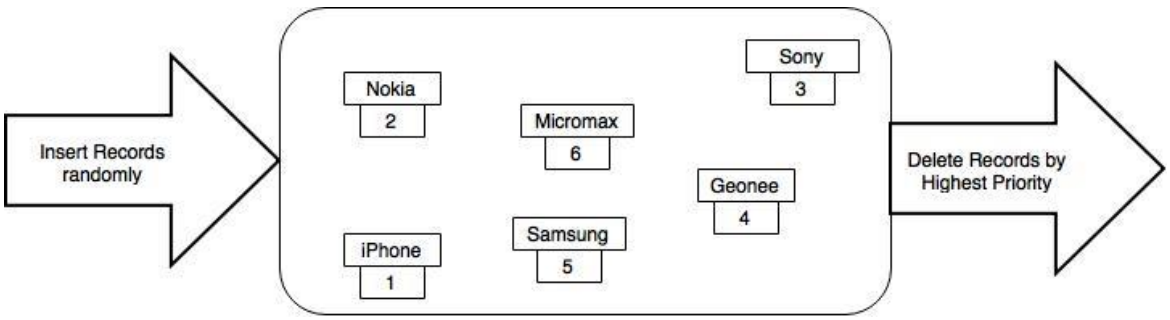

Fig. 1 Concept of Priority Queue

Characteristic Operations:

1) Insert all the items randomly
2) Find and remove the largest (or smallest) item (DeleteMax or DeleteMin).

Algorithm:

Input: a collection S storing n elements


```
Output: the collection S sorted
P = new PQueue()
while !S.isEmpty() do
e = S.removeFirst()
P.insert(e)
while !P.isEmpty()
e = P.removeMin()
S.ADDLAST(E)
```

B. Insertion Sort Algorithm

Insertion sort is one of the simple and efficient comparison methods to sort a list of records. In this algorithm, each iteration removes a record from the input list of records and inserts it into the correct position in the list being sorted. The choice of the record being removed from the input is random and this process is continued until all input records have been gone through. Both average and worst-case time is O(n2) [11].

Algorithm:

```
INSERTION-SORT(A)
for j ←2 length[A]
do key ←A[j]
insert A[j] into the sorted sequence A[1,j-1]
i ←j-1
while i>0 and A[i]>key
do A[i+1] ←A[i]
i ←i-1
A[i+1] ←key
```

C. Test Cases to Sort Record on the Basis of Priorities (No Attribute Value)

Input Values:

Suppose we have a record of 5 employees of different designation and experience who work in a University.

TABLE I
INPUT VALUES

Emp Name	Dept	Desig.	Exp	Priority
Krishna	CSE	Prof.	12	1
Lipika	ECE	Ass.Prof.	10	3
Sharat	EE	Ass.Prof.	12	3
Rupa	CSE	Assoc.Prof	10	2
Shiny	EE	Prof.	10	1

Expected Output for Priority Based Records (No Attribute Value):

Suppose, we want to display required data first (highest priority) and then remaining records (lower priority) of employee.

TABLE II
EXPECTED OUTPUT FOR PRIORITY BASED RECORDS (NO ATTRIBUTE VALUE)

Emp Name	Dept	Desig.	Exp	Priority
Krishna	CSE	Prof.	12	1
Shiny	EE	Prof.	10	1
Rupa	CSE	Assoc.Prof	10	2
Lipika	ECE	Ass.Prof.	10	3
Sharat	EE	Ass.Prof.	12	3

D. Test Cases to Sort Record on the Basis of Required Data (Single Attribute)

Input Values:

Consider input values of *TABLE I.*

Expected Output on the basis of priority (For One Attribute Value-"Exp"):

Suppose, we want to display required data only on the basis of any one attribute value, and suppose we consider "Exp" as one attribute and we want to display records of only those employees whose "Exp" is 12 to be in order on the basis of priority.

TABLE III
EXPECTED OUTPUT FOR PRIORITY BASED RECORDS (SINGLE ATTRIBUTE VALUE)

Emp Name	Dept	Desig.	Exp	Priority
Krishna	CSE	Prof.	12	1
Sharat	EE	Ass.Prof.	12	3

E. Test Cases to Sort Record on the Basis of Required Data (Multiple Attributes)

Input Values:

Consider input value of TABLE I.

Expected Output on the basis of priority (For Multiple Attributes Value-"Desig" & "Exp"):

Suppose, we want to display required data only on the basis of multiple attribute values, and suppose we consider "Desig" and "Exp" as two attributes and we want to display records of only those employees whose "Desig" is Prof and "Exp" is 10 to be in order on the basis of priority.

TABLE IV
EXPECTED OUTPUT FOR PRIORITY BASED RECORDS (MULTIPLE ATTRIBUTES VALUE)

Emp Name	Dept	Desig.	Exp	Priority
Shiny	EE	Prof.	10	1

VII. RESULTS AND ANALYSIS

A. Results

Given Input:
In the input, we are giving details of 6 different employees who works in University.

TABLE V
INPUT RECORDS

Name	Department	Designation	Experience	Priority
Deeksha	Computer Science	Assistant Professor	10	3
Swikar	Civil	Professor	11	1
Sirin	Chemical	Associate Professor	12	2
Shresht	Computer Science	Professor	12	1
Vidhi	Electrical	Associate Professor	10	2
Krishna	Electrical	Associate Professor	12	2

Output for One Attribute Value on the Basis of Priority:
Now, suppose we want to display only those records whose experience is 12 and sort them on the basis of priority. It will display sorted records priority-wise like Professor first, then Associate Professor and at last Assistant Professor having experience 12 years as shown in Fig. 2, where attribute is Experience only.

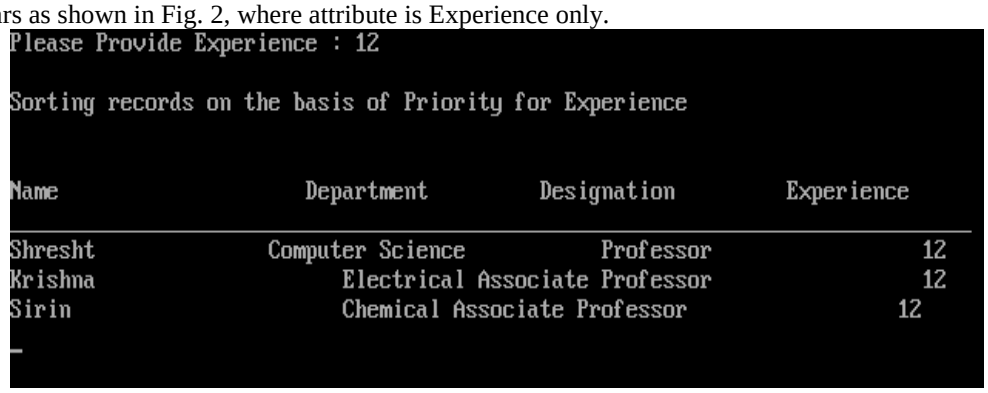

Fig. 1 Output Data for One Attribute Based on Priority

Output for Two Attributes Value on the Basis of Priority:

Suppose, we want to display only Associate Professors who's Experience is 12 and display them in sorted form. It will display sorted records of Associate Professor of all departments having Experience 12 years name-wise as shown below in Fig. 3, where two attributes are Designation and Experience.

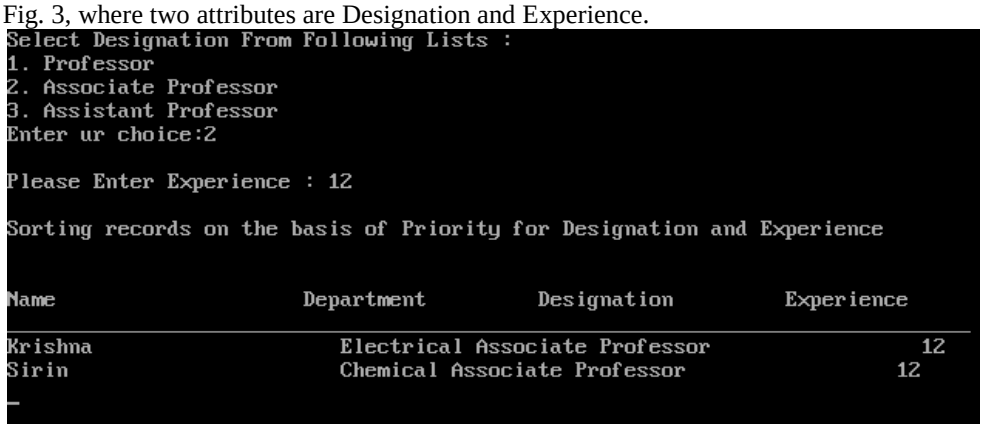

Fig. 3 Output Data for Multiple Attributes Based on Priority

B. Analysis

We have analyzed 3 cases as follows:

- Case 1: General Sorting for all data v/s Priority-based Sorting for all data
- Case 2: General Sorting for all data v/s Priority-based Sorting for one given attribute value
- Case 3: General Sorting for all data v/s Priority-based Sorting for multiple given attribute values

Case 1: General Sorting for All Data v/s Priority-Based Sorting for All Data

In this case, we are analysing the time complexity and space complexity of general sorting and priority-based sorting for all records. In general sorting, we are comparing each element and arranging them in the desired order. So the time complexity becomes $O(n^2)$. In priority-based sorting, we are comparing each element but we are arranging them according to priority. But since we are comparing each records, time complexity remains same, i.e. $O(n^2)$.

TABLE VI
COMPARISON BETWEEN GENERAL SORTING AND PRIORITY-BASED SORTING FOR ALL DATA

<table>
<tr><th colspan="2">General Sorting for All Data</th><th colspan="2">Priority-based sorting for All Data</th></tr>
<tr><th>Pseudo code</th><th>Time</th><th>Pseudo code</th><th>Time</th></tr>
<tr><td>Insert ‘n’ record to an array
Sort ‘n’ records of an array
For i=0 to n
For j=i+1 to n
If A[i] >A[j]
Exchange</td><td>n

n
n

1</td><td>Insert ‘n’ records to an array and assign priority to each record
Sort ‘n’ records on the basis of priority
For i=0 to n
For j=i+1 to n
If A[i].Pr>A[j].Pr
Exchange</td><td>n

n
n

1</td></tr>
<tr><td>Expected Output:
<table><tr><th>Name</th><th>Dept</th><th>Desig.</th><th>Exp</th></tr><tr><td>Krishna</td><td>CSE</td><td>Prof.</td><td>12</td></tr><tr><td>Lipika</td><td>ECE</td><td>Ass.Prof.</td><td>10</td></tr><tr><td>Rupa</td><td>CSE</td><td>Assoc.Prof</td><td>10</td></tr><tr><td>Sharat</td><td>EE</td><td>Ass.Prof.</td><td>12</td></tr><tr><td>Shiny</td><td>EE</td><td>Prof.</td><td>10</td></tr></table></td><td></td><td>Expected Output:
<table><tr><th>Name</th><th>Dept</th><th>Desig.</th><th>Exp</th><th>Pr.</th></tr><tr><td>Krishna</td><td>CSE</td><td>Prof.</td><td>12</td><td>1</td></tr><tr><td>Shiny</td><td>EE</td><td>Prof.</td><td>10</td><td>1</td></tr><tr><td>Rupa</td><td>CSE</td><td>Asso.Prof</td><td>10</td><td>2</td></tr><tr><td>Lipika</td><td>ECE</td><td>Ass.Prof.</td><td>10</td><td>3</td></tr><tr><td>Sharat</td><td>EE</td><td>Ass.Prof.</td><td>12</td><td>3</td></tr></table></td><td></td></tr>
<tr><td>Time Complexity, $T(n)1 \approx O(n^2)$</td><td></td><td>Time Complexity, $T(n)1 \approx O(n^2)$</td><td></td></tr>
<tr><td>Space Complexity, S(n)=1, since it takes 1 temporary variable to exchange the records. So S(n)=O(1).</td><td></td><td>Space Complexity, S(n)=1, since it takes 1 temporary variable to exchange the records. So S(n)=O(1).</td><td></td></tr>
</table>

Case 2: General Sorting for All Data v/s Priority-Based Sorting for One Given Attribute Value

In this case, we are analysing the time complexity and space complexity of general sorting and priority-based sorting for one given attribute value. In general sorting, we are comparing each element and arranging them in the desired order. So the time complexity becomes $O(n^2)$. In priority-based sorting for one attribute value, we are comparing only those records whose attribute value has been given and we are arranging them according to priority. Hence, we are reducing the time of comparing the number of recordss. The complexity becomes $O(n'^2)$.

TABLE VII
COMPARISON BETWEEN GENERAL SORTING FOR ALL DATA AND PRIORITY-BASED SORTING FOR SINGLE GIVEN ATTRIBUTE VALUE

<table>
<tr><th colspan="2">General Sorting for All Data</th><th colspan="2">Priority-Based Sorting for One Given Attribute Value</th></tr>
<tr><th>Pseudo code</th><th>Time</th><th>Pseudo code</th><th>Time</th></tr>
<tr><td>Insert ‘n’ record to an array
Sort ‘n’ records of an array
For i=0 to n
For j=i+1 to n
If A[i] >A[j]
Exchange</td><td>n

n
n

1</td><td>Insert ‘n’ records to an array and assign priority to each record
Give any one attribute value whose data you want to display
For i=0 to n
If attrib_val = A[i].attrib_val
Insert the record into “Temporary” array, say T[i] and increment ‘counter’ to 1
Sort “Temporary” array on the basis of priority
For i=0 to count
For j=i+1 to count
If T[i].Pr>T[j].Pr
Exchange</td><td>n

1

n

1

n'
n'

1</td></tr>
<tr><td>Expected Output:
Consider previous case.</td><td></td><td colspan="2">Expected Output:
<table><tr><th>Name</th><th>Dept</th><th>Desig.</th><th>Exp</th><th>Pr.</th></tr><tr><td>Krishna</td><td>CSE</td><td>Prof.</td><td>12</td><td>1</td></tr><tr><td>Sharat</td><td>EE</td><td>Ass.Prof.</td><td>12</td><td>3</td></tr></table></td></tr>
<tr><td>Time Complexity, $T(n) \approx O(n^2)$</td><td></td><td colspan="2">Time Complexity, $T(n) \approx O(n'^2)$, where n'<n. hence time complexity reduced.</td></tr>
<tr><td>Space Complexity, S(n)=1, since it takes 1 temporary variable to exchange the records. So S(n)=O(1).</td><td></td><td colspan="2">Space Complexity, S(n)=n'+1, where n' indicates size of temporary array and 1 indicates temporary variable to exchange the records. So S(n)=O(n'+1).</td></tr>
<tr><td colspan="4">Note: The time complexity for remaining records becomes $O(n^2-n'^2)$ and space complexity for remaining record becomes $O(n^2-n'^2)$.</td></tr>
</table>

Case 3: General Sorting for All Data v/s Priority-Based Sorting for Multiple Given Attribute Values

In this case, we are analysing the time complexity and space complexity of general sorting and priority-based sorting for one given attribute value. In general sorting, we are comparing each element and arranging them in the desired order. So the time complexity becomes $O(n^2)$. In priority-based sorting for multiple attribute value, we are comparing only those records whose attribute value has been given and we are arranging them according to priority. Hence, we are reducing the time of comparing the number of records. The complexity becomes $O(n''^2)$.

TABLE VIII
COMPARISON BETWEEN GENERAL SORTING FOR ALL DATA AND PRIORITY-BASED SORTING FOR MULTIPLE GIVEN ATTRIBUTES VALUE

General Sorting for All Data		Priority-Based Sorting for Multiple Given Attribute Value	
Pseudo code	**Time**	**Pseudo code**	**Time**
Insert ‘n’ record to an array Sort ‘n’ records of an array For i=0 to n For j=i+1 to n If A[i] >A[j] Exchange	n n n 1	Insert ‘n’ records to an array and assign priority to each record Give any two attribute values whose data you want to display For i=0 to n If attrib_val1 = A[i].attrib_val1 && attrib_val2 = A[i].attrib_val2 Insert the record into “Temporary” array, say T[i] and increment ‘counter’ to 1 Sort “Temporary” array on the basis of priority For i=0 to count For j=i+1 to count	n 1 n 1 n'' n''

<table>
<tr><td></td><td></td><td>If T[i].Pr>T[j].Pr
Exchange</td><td>1</td></tr>
<tr><td>Expected Output:
Consider previous case.</td><td></td><td>Expected Output:<table><tr><th>Name</th><th>Dept</th><th>Desig.</th><th>Exp</th><th>Pr.</th></tr><tr><td>Shiny</td><td>EE</td><td>Prof.</td><td>10</td><td>1</td></tr></table></td><td></td></tr>
<tr><td>Time Complexity, T(n)$\approx$O(n^2)</td><td></td><td>Time Complexity, T(n) $\approx$ O(n"^{2}), where n"<n. hence time complexity reduced.</td><td></td></tr>
<tr><td>Space Complexity, S(n)=1, since it takes 1 temporary variable to exchange the records. So S(n)=O(1).</td><td></td><td>Space Complexity, S(n)=n"+1, where n" indicates size of temporary array and 1 indicates temporary variable to exchange the records. So S(n)=O(n"+1).</td><td></td></tr>
<tr><td colspan="4">Note: The time complexity for remaining records becomes O(n^2-n"^{2}) and space complexity for remaining record becomes O(n^2-n"^{2}).</td></tr>
</table>

CONCLUSION

The researchers always try to reduce the time complexity by proposing new methods. So we have implemented proposed sorting algorithm which display only required records on the basis of priority for multiple attributes to save time by not displaying all records. So, we are saving time in terms of sorting as well as searching operations.

ACKNOWLEDGEMENT

I would like to express my cordial thanks to my supervisor Mr. B.L. Pal, HoD, Department of Computer Science & Engineering, Mewar University, Gangrar for his valuable advices and suggestion so that I could complete my work comfortable.

REFERENCES

[1] Udit Agarwal, *"Data Structure Using C"*, 2nd Edition, S.K. Kataria & Sons Publications, July 2011
[2] M.S. Garai Canaan.C and M. Daya, "*Popular sorting algorithms*", World Applied Programming, 1:62–71.
[3] A.D. Mishra and D. Garg, "*Selection of Best Sorting Algorithm*", International Journal of Intelligent Processing, 2(2):363–368.
[4] Pankaj Sareen, "*Comparison of sorting algorithms (on the basis of average case)*", IJARCSSE, 3:522–532.
[5] Donald E.Knuth, "*The Art of Computer Programming Second Edition"*, volume 3. ADDISON-WESLEY.
[6] Aditya Dev Mishra, *"Selection of Best Sorting Algorithm for a Particular Problem"*, 2009
[7] Tim Bell, Bengt Aspvall, "*Sorting Algorithms as Special Cases of a Priority Queue Sort*", SIGCSE'11, March 9–12, 2011, Dallas, Texas, USA. Copyright 2011 ACM 978-1-4503-0500.
[8] Ahmed M. Aliyu, Dr. P. B. Zirra, "*A Comparative Analysis of Sorting Algorithms on Integer and Character Arrays*", The International Journal Of Engineering And Science (IJES) ||Volume||2 ||Issue|| 7 ||Pages|| 25-30||2013|| ISSN(e): 2319 – 1813 ISSN(p): 2319 – 1805.
[9] Ghiath Al Aqel, *"A New Priority-Sort Based Optimization Algorithm Integrated Process Planning and Scheduling"*, International Journal of Modeling and Optimization, Vol. 3, No. 2, April 2013.
[10] Jehad Hammad, *"A Comparative Study between Various Sorting Algorithms",* IJCSNS International Journal of Computer Science and Network Security, VOL.1 5 No.3, March 2015.
[11] Thomas H. Cormen, *"Introduction to Algorithms"*, Third Edition, MIT Press, July 2009.

