TECHNICAL REPORT

Operation research project part 2

NAME: RAMY MAMDOUH
ID: 118371

Contents

1) What is the basic philosophy behind OR and the so-called “OR”.

First let’s start by defining the Object research concept to understand the philosophy of object oriented subject:

Briefly The Operation research Concern about reaching the optimum or near optimum solution for the problem by developing Specific methods with the least amount of steps, it also helps in improvement of decision making by analyzing

Definition related to the C.S major:

In OR, a problem is first clearly defined and represented (modeled) as a set of mathematical equations. It is then subjected to rigorous computer analysis to yield a solution (or a better solution) which is tested and re-tested against real-life situations until an optimum solution is found. OR applies different approaches to different types of problems: dynamic programming, linear programming, and critical path method are used in handling complex information in allocation of resources, inventory control, and in determining economic reorder quantity; forecasting and simulation techniques such as Monte Carlo method are used in situations of high uncertainty such as market trends, next period's sales revenue, and traffic patterns. Also called decision science, management science, or operational research.

The development, analysis, solution, and delivery of Operations Research/Management Science models has changed dramatically due to the phenomenal improvements in computer technology. At the same time, disciplines from computer science have benefitted from the advances achieved by operations researchers

In operations research, problems are broken down into basic components and then solved in defined steps by mathematical analysis.

Definition in U.K: is a discipline that deals with the application of advanced analytical methods to help make better decisions. So what is advanced analytical methods: Advanced analytics refers to future-oriented analyses that can be used to help drive changes and improvements in business practices .example in our course: increase in profit problems and how to get the maximum profit by using the minimum resources. And what are those analytical methods? Analytical methods used in OR include mathematical logic, simulation, network analysis, queuing theory, and game theory.

Field: mathematics.

When we must use OR?

At complex decision-making problems example: Operations research is often concerned with determining the maximum (of profit, performance, or yield) or minimum (of loss, risk, or cost) of some real-world objective.

So after Understanding the real meaning of The OR what is the philosophy behind OR:

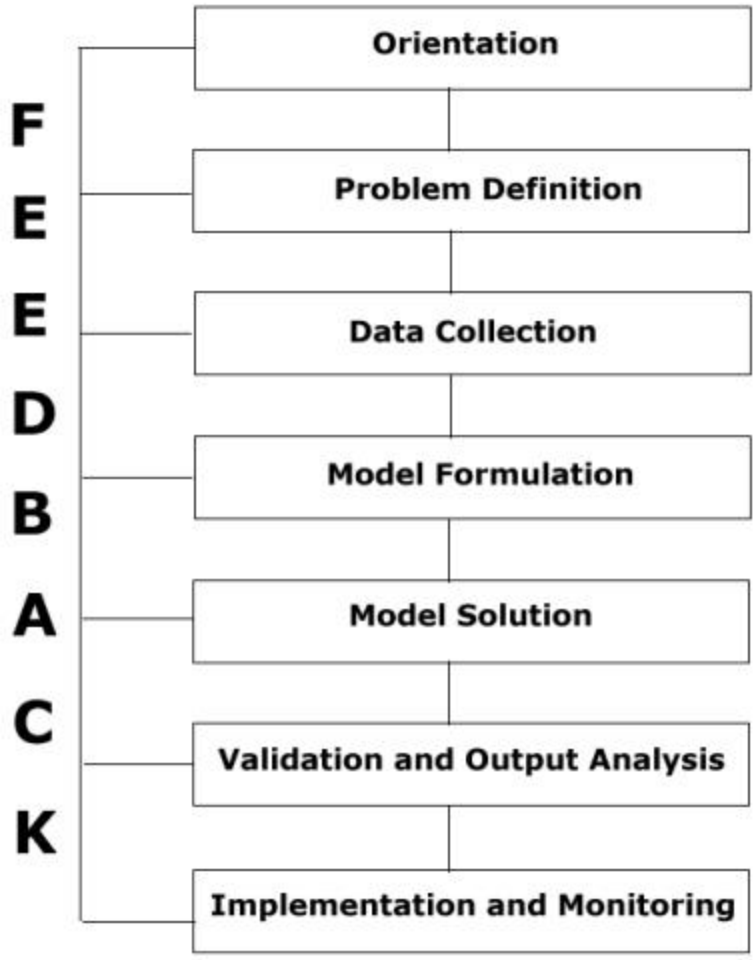
Operations Research (OR) is the professional discipline that deals with the application of information technology for informed decision-making. OR professionals aim to provide rational bases for decision making by seeking to understand and structure complex situations and to use this understanding to predict system behavior and improve system performance. Much of this work is done using analytical and numerical techniques to develop and manipulate mathematical and computer models of organizational systems composed of people, machines, and procedures.

Operations Research draws upon ideas from engineering, management, mathematics, and psychology to contribute to a wide variety of application domains; the field is closely related to several other fields in the "decision sciences" — applied mathematics, computer science, economics, industrial engineering, and systems engineering.

Operations Research is distinguished by its broad applicability and by the wide variety of career opportunities and work styles it embraces. Because the concepts and methods of OR are so pervasive, operations research offers highly flexible career paths.

THE OPERATIONS RESEARCH APPROACH

Given that O.R. represents an integrated framework to help make decisions, it is important to have a clear understanding of this framework so that it can be applied to a generic problem. To achieve this, the so-called *O.R. approach* is now detailed. This approach comprises the following seven sequential steps: (1) Orientation, (2) Problem Definition, (3) Data Collection, (4) Model Formulation, (5) Solution, (6) Model Validation and Output Analysis, and (7) Implementation and Monitoring. Tying each of these steps together is a mechanism for continuous feedback; Figure 1 shows this schematically.



O.R. IN THE REAL WORLD

Production Planning at Harris Corporation - Semiconductor Section

Gasoline Blending at Texaco

FMS Scheduling at Caterpillar

Fleet Assignment at Delta Airlines

KeyCorp Service Excellence Management System

What is the Running Time Analysis of the Quicksort algorithm?

Quicksort is a recursive sorting algorithm based on divide and conquer approach.

It picks an element from the list, called the pivot element (more on this later), and uses it to divide the list into two parts. The first part contains all of the elements that are smaller than the pivot element and the second part of the list contains all of the elements that are larger than the pivot element.

Quicksort is a very efficient sort on average, although its worst case performance is identical to insertion sort and bubble sort O (n2).

Quicksort algorithm to help us to solve the problem:

Quicksort (A, lo, hi) {

// A the list of elements to be sorted

// lo the index of the first element in the list to sort

// hi the index of the last element in the list to sort.

If lo < hi {

J -> Partition (A, lo, hi); // returns the final index of the pivot (j)

Quicksort (A, lo, j-1);

Quicksort (A, j+1, hi);

}

- Best case: If all splits happen in the middle of the corresponding subarrays, O (n log n)
- Worst case: if all the splits are skewed to the extreme (i.e. one of the subarrays is empty), O (n2)
- Average case: O (n log n)

Using recurrence we can get the analysis of quicksort:

Let us suppose that the pivot we just chose has divided the array into two parts - one of size k and the other of size n − k. Notice that both these parts still need to be sorted. This gives us the following relation:

T (n) = T (k) + T (n − k) + C*n

Where T (n) refers to the time taken by the algorithm to sort n elements.

C= Constant

WORST CASE ANALYSIS:

T (n) = T (n − 1) + C*n

= T (n − 2) + C*(n-1) + C*n

= T (n − 2) + 2*C*n -C

= T (n − 3) + 3*C*n -3*C

= T (n − 4) + + 3*C*n -6*C

Then

= T (n – k) +K*C*n- (k (k-1)/2)*C

So in case of T (1)

N-k=1 then n=k

SO

=T (1) +Cn^{2-}(n (n-1)/2)*C

T (n) =C1+(C*n*(n+1))/2

Therefore the answer is equal to:

=C*n^2/2+C*n/2+C

Then the time complexity is O (n^2)

This is the worst case of quick-sort, which happens when the pivot we picked turns out to be

The least element of the array to be sorted, in every step (i.e. in every recursive call). A similar

Situation will also occur if the pivot happens to be the largest element of the array to be sorted.

BEST CASE ANALYSIS:

The best case of quicksort occurs when the pivot we pick happens to divide the array into two

Exactly equal parts or almost equal, in every step. Thus we have k = n/2 and n − k = n/2 for the original array of

Size n.

Since T (n) = time taken to solve the problem

And

Quicksort (A, lo, j-1) --->takes T (n/2)

Quicksort (A, j+1, hi) --->takes T (n/2)

C = Constant

Consider, therefore, the recurrence:

T (n) = 2T (n/2) + C*n if n>1

C1, if n=1

Therefore T (1) =C

T (n) = 2T (n/2) + C*n

= 2 (2T (n/4) + C*(n/2)) + C*n

= 4T (n/4) + 2C*n (By simplifying and grouping terms together).

= 8T (n/8) + 3C*n

= 2^k T ($n/2^k$) + k*C*n (Continuing likewise till the kth step)

Notice that this recurrence will continue only until n = 2k

Therefore: $n/2^k$ =1 therefore k=$\log_2$ n

Until k = log n. Thus, by putting k = log n, we have the following equation:

T (n) = n*T (1) + C*n log n, which is O (n log n).

3) The mathematical tools and the specific techniques used to build

And analyze the sorting algorithms.

Firstly the Specific techniques used to Build and design algorithms:

Algorithm design is identified and incorporated into many solution theories of operation research, such as dynamic programming and divide-and-conquer.

Techniques for designing and implementing algorithm designs are algorithm design patterns,[1] such as template method pattern and decorator pattern, and uses of data structures, and name and sort lists. Some current day uses of algorithm design can be found in internet retrieval processes of web crawling, packet routing and caching.

Examples of sorting algorithm design patterns:

Secondly the mathematical tools to Build and design algorithms:

Mainframe programming languages such As ALGOL (for *Algo*rithmic *l*anguage), FORTRAN, COBOL, MATLAB,PL/I, SAIL, and SNOBOL are computing tools to implement an "algorithm design"... but, an "algorithm design" (a/d) is not a language. An a/d can be a hand written process, e.g. set of equations, a series of mechanical processes done by hand, an analog piece of equipment, or a digital process and/or processor. One of the most important aspects of algorithm design is creating an algorithm that has an efficient run time, also known as its big Oh.

Steps in development of Algorithms

1. Problem definition
2. Development of a model
3. Specification of Algorithm
4. Designing an Algorithm
5. Checking the correctness of Algorithm
6. Analysis of Algorithm
7. Implementation of Algorithm
8. Program testing
9. Documentation Preparation

Thirdly the mathematical tools and the specific techniques used to analyze the non-recursive sorting algorithms.

Empirically

1. Implement the algorithm using any programming language.

2. Execute it with inputs of varying size and composition (different inputs of same size) to see how long it takes (you can use a method like System.currentTimeMillis (), or stopwatch class in java to get a measure of the running time.

3. Draw a curve illustrates the running time versus input data size.

Mathematically

Express the algorithm using code or pseudo code.

1. Decide on parameter n indicating input size
2. Identify algorithm's basic operation
3. Determine worst, average, and best cases for input of size n
4. Set up a sum for the number of times the basic operation is executed
5. Simplify the sum using standard formulas and rules

Asymptotic analysis

- Big Omega
- Big Theta
- Big O
- Little-O and Omega

The mathematical tools and the specific techniques used to analyze recursive sorting algorithms

1) **Substitution Method**

2) **Recurrence Tree Method**

3) **Master Method**

Discuss and compare the running time of some versions of sorting Algorithms (Recent versions; 2013-2015)

SELECTION SORT

Selection sort is one of the simplest sorting algorithms. It is actually an improvement in performance of bubble sort. This algorithm is called selection sort because it works by selecting a minimum element in each step of the sort. This algorithm, iterate through a list of n unsorted items, has a worst-case, average-case, and best-case run- time of $\Theta(n^2)$, assuming that comparisons can be done in constant time. The Selection sort spends most of its time trying to find the minimum element in the "unsorted" part of the array. The brief algorithm for selection sort is given below

Name: algorithm 1

SELECTION_SORT (A)		Cost of Each Step
1:	for i ← 0 to n-1 do	O (n)
2:	min ← i;	O (1)
3:	for i ← 0 to n-1 do	O (n^2)
4:	If A[j] < A[min] then	O (1)
5:	min←j	O (1)
6:	swap (A[i], A[min])	O (1)

Total run time Cost== $\Theta(n^2)$ asymptotically

It works as follows:

1. Find the smallest element using a linear scan.
2. Swap the element with the element in first position.
3. Find the second smallest element in the remaining array.
4. Then swap to the second position.
5. Continue the steps until a sorted list is obtained.

PERFORMANCE ANALYSIS OF SELECTION SORT

For each value of element until n it needs to compare and check whether there is any value smaller than it and needs (n-1) comparisons. Thus it can be easily seen that run time complexity of selection sort is Θ (n^2) from line No.3.But since it just requires the Θ (n) swaps means linear order writes to the memory which is optimal to any sorting algorithm. Table 1 and 2. Further clarifies the above statement. The running time is obtained by calculating how many times the line 4 of the above algorithm got executed on the specified array size.

Table 1. Running time of selection sort on integer

Number of Array Elements	Running Time (sec)
1000	0.364
1500	0.565
2000	0.763
2500	0.965
3000	1.064

Table 2. Running time of selection sort on String Array

Number of Array Elements	Running Time (sec)
1000	0.384
1500	0.762
2000	0.963
2500	1.156
3000	1.361

From the above table it is clear that this naive approach is very good for small number of elements.

But it requires the comparison of every element to every other element which will lead to a great deal of work for large data sets. The worst case occurs if the array is already sorted in descending order. Nonetheless, the time require by selection sort algorithm is not very sensitive to the original order of the array to be sorted: the test "if A[j] <A[min]" is executed exactly the same number of times in every case.

QUICK SORT

Quick sort is a comparison sort developed by Tony Hoare. Also, like merge sort, it is a divide and conquer algorithm, and just like merge sort, it uses recursion to sort the lists. It uses a pivot chosen by the programmer, and passes through the sorting list and on a certain condition, it sorts the data set. Quick sort algorithm can be depicted as follows:

Name: algorithm 2

QUICKSORT (A)

1: step←m;

2: while step > 0

3: for (i←0 to n with increment 1)

4: do temp←0;

5: do j←i;

6: for (k←j+step to n with increment step)

7: do temp←A[k];

8: do j←k-step;

9: while (j>=0 && A[j]>temp)

10: do A[j+step]=A[j];

11: do j←j-step;

12: do Array[j]+step←temp;

13: do step←step/2;

Run Time of Quick Sort

Best case is O (n*log n) Average case is O (n*log n) and the worst case is O (n^2)

Table 1. Running time of selection sort on integer

The main focus of the Quick sort lies in line 6. Line 6-12 is algorithm for insertion sort. But insertion sort is performed between the elements between some specified gaps. Since the sorted order of elements get increased with the completion of each step these lines gives a constant value less than the value of n. Thus in the best case of this is almost O(n).But there may be the worst case condition for large number of inputs where computational complexity where the gap insertion sort can give almost n steps resulting into running time of $\Theta(n^2)$.

PERFORMANCE ANALYSIS OF QUICK SORT

The following table gives the running time of Quick sort for both integer and string array .It is obtained by running the above algorithm in the different input elements. Running time is obtained by counting the number of times line 10 and 11 got executed

Table 3. Running time of Quick sort on integer

Number of Array Elements	Running Time (sec)
1000	0.836
1500	0.998
2000	1.032
2500	1.221
3000	1.398

Table 4. Running time of Quick sort on string

Number of Array Elements	Running Time (sec)
1000	1.002
1500	1.124
2000	1.232
2500	1.431
3000	1.634

Thus from the above table it is clear that it is not exactly $\Theta(n^2)$. The increment in the performance of Quick sort is because it tries to implement the best part of insertion sort. After each gap step the array elements will get partly sorted which makes the performance increase for the Quick sort.

PERFORMANCE COMPARISION

Selection sort and Quick sort are both comparisons based sorting algorithms which uses two different perspectives. One operates by selecting smallest element of the unsorted part of the list each time. Another operates by performing insertion sort in a gap sequence. The generalized performance of the two algorithms on both integer and string arrays is summarized in the table below:

Table 5 summary of run time on of integers and String Arrays for the two sorting techniques

	Selection Sort		Quick Sort	
	Integer	String	Integer	String
1000	0.364	0.384	0.836	1.002
1500	0.565	0.762	0.998	1.124
2000	0.763	0.963	1.032	1.232
2500	0.965	1.156	1.221	1.431
3000	1.064	1.361	1.398	1.634

DISCUSSION

From the above analysis also it is clear that performance of selection sort is better than Quick sort.

Selection Sort is fairly simple algorithm and can be easily implemented. The algorithm of Quick sort is rather complex and it will give very poor running time in case of large sets of data. But the performance of both algorithms is worse than the lower bound run time of comparison sort O (n*log n).These algorithms can be implemented for small amount of data but not for large amount of data.

CONCLUSION

Selection sort shows better performance than Quick sort but being the simple structure selection sort is more preferred and widely used. It is clear that both sorting techniques are not that popular for the large arrays because as the arrays size increases both timing is slower for integer and string, but generally the study indicate that integer array have faster CPU time than string arrays. Both have the upper bound running time O (n^2).

In your opinion what questions will be hot topics in this field during the next decade?

What will Operation research do to help in decision theory under different kind of uncertainties, Uncertainty modelling, Supply chain decisions and Agri-food supply chains?

Will Formal modeling and integrating organizational, sociological, and political theory be added into operations management and research?

Determine Behavior of Human Resources in process and mechanical operations

Will Operation research be able to model the social media networks?

Reference:

1. **https://www.informs.org/About-INFORMS/What-is-Operations-Research**
2. **http://www.springer.com/series/6375**
3. **http://whatis.techtarget.com/definition/operations-research-OR**
4. **http://www.businessdictionary.com/definition/operations-research-OR.html**
5. **http://www.cise.ufl.edu/class/cot3100fa07/quicksort_analysis.pdf**
6. **Dr Abeer Hamdy lectures-analysis of algorithm**
7. **http://www.smu.edu/Lyle/Departments/EMIS/Advising/Doctoral/PhDOR**
8. **http://www.pitt.edu/~jrclass/or/or-intro.html**
9. **http://en.wikipedia.org/wiki/Algorithm_design**
10. **http://www.theijes.com/papers/v2-i7/Part.3/D0273025030.pdf**