CHAPTER III

BRUTE FORCE

Topics:

1. Brute Force Technique
2. Bubble Sort
3. Selection Sort
4. Sequential or Linear Search
5. Brute force string matching
6. Exhaustive Search
   a. Traveling salesman problem
   b. Knapsack Problem
   c. Assignment Problem.

A. M. PRASAD
Assistant Professor
Department of Compuier Science & Engineering
Dayananda Sagar College of Engineering
Bangalore - 560 078

I BRUTE FORCE TECHNIQUE

There are various approaches to solve a given problem. Brute force technique is one of the approach to solve the given problem. Brute force technique can be described as

"Brute Force is a straight forward approach to solving a problem directly based on problem statement and definition of the concept involved"

Some of the problems based on Brute force technique are

* Algorithm to find GCD of two nos
* Matrix operations such as addition, Subtraction
* Sorting such as bubble sort, Selection Sort
* Searching like linear search
* Exhaustive search, Traveling Salesman Problem, Knapsack problem, Assignment problem

The advantages of Brute force technique are

1. Simple and easy to write algorithm
2. Can be used to judge more efficient alternative approach to solve a problem

The disadvantages of Brute force technique are

1. Usually algorithms are slow.
2. Very rarely yeilds best solution

II BUBBLE SORT

Bubble sort is the Brute force method of arranging numbers in ascending order. The bubble sort method compares adjacent elements, if necessary arranges them in order by swapping them. This process is repeated in different cycles or passes till all elements are arranged in ascending order.

A. M. PRASAD
Assistant Professor
Department of Compuier Science & Engineer'ng
Dayananda Sagar College of Engineering
Bangalore - 560 078

Illustration of bubble sort

Consider the elements 9, 7, 5, 4, 2. The elements are sorted using bubble sort technique as shown below.

		7<9	5<9	4<9	2<9
I cycle/pass.	9	7	7	7	7
	7 ↷	9 ↖	5	5	5
	5 ↷	5 ↷	9 ↖	4	4
	4 ↷	4	4 ↷	9 ↖	2
	2 ↷	2	2	2 ↷	9

At the end of the first cycle, the first biggest element will come to or sink to its proper position. Since the largest element will sink to bottom of the array Bubble sort is also called as SINK SORT

II cycle/pass.

5<7	4<7	2<9
5	5	5
7	4	4
4	7	2
2	2	7
9	9	9

At the end of second cycle, the second biggest element will goto or sink to its proper position.

III cycle/pass.

4<5	2<5
4	4
5	2
2	5
7	7
9	9

At the end of third cycle, the third biggest element will be stored in its proper position. Since elements are not in ascending order, the o/p of third cycle becomes input to fourth cycle.

2<4
2
4
5
7
9

At the end of fourth cycle/pass all the five elements are arranged in ascending order. If 'n' is

the number of elements then bubble sort requires (n-1) cycles to arrange numbers in Ascending order.

The comparisions that are performed on each cycle/pass on elements stored in array are

I cycle	II cycle	III cycle	IV cycle
a[2] < a[1]	a[2] < a[1]	a[2] < a[1]	a[2] < a[1]
a[3] < a[2]	a[3] < a[2]	a[3] < a[2]	
a[4] < a[3]	a[4] < a[3]		
a[5] < a[4]			

A. M. PRASAD
Assistant Professor
Department of Compuier Science & Engineering
Dayananda Sagar College of Engineering
Bangalore - 560 078

The brute force algorithm can be written based on the above comparisions.

Algorithm:

```
Algorithm BubbleSort(a,n)
// I/p: array a containing 'n' unsorted elements
// O/p: array a with 'n' sorted elements
for i ← 1 to n-1 do
    for j ← 1 to n-i do
        if (a[j+1] < a[j]) then
            swap(a[j+1], a[j])
        Endif
    End for
End for
```

Advantages of Bubble Sort

1. Simple and easy
2. Straight forward approach

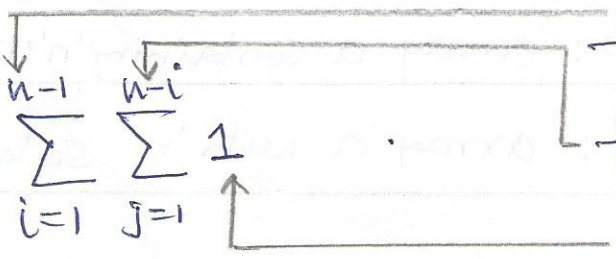
Disadvantages of Bubble Sort

1. Very slow and is not efficient
2. for best i/p also, it requires (n-1) cycles.

Time Efficiency

In the algorithm the key operation is comparision ie (a[j+1] < a[j]) and is only the statement. The number of times this statement will be executed depends upon two for-loops.

Let T(n) be the time taken by bubble sort algorithm to sort 'n' elements



$$T(n) = \sum_{i=1}^{n-1} \sum_{j=1}^{n-i} 1$$

```
for i← 1 to n-1 do
    for j← 1 to n-i do
        if (a[j+1] < a[j] do
            swap (a[j+1], a[j])
```

$$T(n) = \sum_{i=1}^{n-1} 1.((n-i)-1+1$$

UB-LB+1

$$T(n) = \sum_{i=1}^{n-1} n-i-1+1$$

$$T(n) = \sum_{i=1}^{n-1} n-i$$

Now substitute $i=1 \cdots n-1$ to expression $n-i$

$$T(n) = (n-1)+(n-2)+(n-3)+ \cdots\cdots +3+2+1$$

mathematically the above series can be written as

$$T(n) = \frac{n(n-1)}{2}$$

$$T(n) = \frac{n^2-n}{2}$$

A. M. PRASAD
Assistant Professor
Department of Compuier Science & Engineering
Dayananda Sagar College of Engineering
Bangalore - 560 078

Note:- While finding the time efficiency, constants cannot be considered

$$T(n) = n^2-n'$$

Note:- always consider the higher order term

So $T(n) = n^2$

Since n^2 is the time taken in average case or worst case, Θ or O notations can be used to represent time efficiency. Hence the time efficiency is

$$\boxed{T(n) = \Theta(n^2)}$$

III Selection Sort

Selection Sort is Brute force method of arranging numbers in Ascending order. The Selection Sort method selects first element of the list of 'n' numbers and finds the position of smallest element in the rest of the list and swaps first element and minimum element. So first smallest element will comes to its position. In the second cycle, algorithm selects second element and finds the position of minimum element among 3....n elements. Then minimum element is swapped with second position element. So second smallest element will come to second position. This process is repeated (n-1) times to arrange 'n' elements in ascending order.

Illustration of Selection Sort

Consider the elements 9, 7, 2, 5, 4. These elements are sorted using Selection Sort as described below

I cycle

[9] ← Selected		2 ✓
7		7
2 ← position of		9
5	smallest element	5
4	swap them	4

At the end of first cycle, first smallest element will go to first position

A. M. PRASAD
Assistant Professor
Department of Compuier Science & Engineering
Dayananda Sagar College of Engineering
Bangalore - 560 078

II cycle

2		
[7] ← Selected		2
9		4 ✓
5		9
4 ← position of smallest		5
element, swap them		7

At the end of second cycle, second smallest element will be stored in second position.

III cycle

2		2
4		4
[9] ← Selected		5 ✓
5 ← position of		9
7	minimum, swap them	7

IV cycle

2		2
4		4
5		5
[9] Selected		7 ✓
7 minimum element position		9

At the end of fourth cycle, all the five elements are sorted in order. If 'n' is the number of elements then the Selection Sort requires (n-1) cycles to arrange numbers in ascending order.

The basic idea while writing Brute force Selection Sort algorithm is selecting the element and finding the position of the smallest element among the Position of the element selected +1 to n.

Algorithm

```
Algorithm SelectionSort(a, n)
// I/p : array a containing n unsorted nos
// O/p : a containing sorted elements
for i ← 1 to n-1 do
    begin
    min ← i
    for j ← i+1 to n do
        begin
        if (a[j] < a[min]) then
            min ← j
        Endif
    Endfor
    swap(a[i], a[min])
Endfor
```

Advantages of Selection Sort

1. Simple and easy
2. Straight forward approach

Disadvantages of Selection Sort

1. Very slow and not efficient
2. algorithm not constructive

A. M. PRASAD
Assistant Professor
Department of Computer Science & Engineering
Dayananda Sagar College of Engineering
Bangalore - 560 078

Time Efficiency

The key operation in Selection Sort algorithm is (A[j] < A[min]) and is only the statement. The number of times this statement will be executed depends upon two for loops.

Let T(n) be the time taken by Selection Sort algorithm to sort 'n' elements

```
for i ← 1 to n-1 do
    min ← i
    for j ← i+1 to n do
        if (A[j] < A[min])
            j ← min
```

$$T(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1$$

$$T(n) = \sum_{i=1}^{n-1} 1.(n-(i+1))+1$$

UB − LB + 1

$$T(n) = \sum_{i=1}^{n-1} n - i^0 - 1 + 1$$

$$T(n) = \sum_{i=1}^{n-1} n - i^0$$

by applying $i^0 = 1 \ldots n-1$ in equation $n - i^0$

$$T(n) = (n-1) + (n-2) + (n-3) + \cdots\cdots + 1$$

mathematically, series can be written as

$$T(n) = \frac{n(n-1)}{2}$$

$$T(n) = \frac{n^2 - n}{2}$$

Note: Constants cannot be considered

$$T(n) = n^2 - n$$

Note: Only higher order term should be considered

$$T(n) = n^2$$

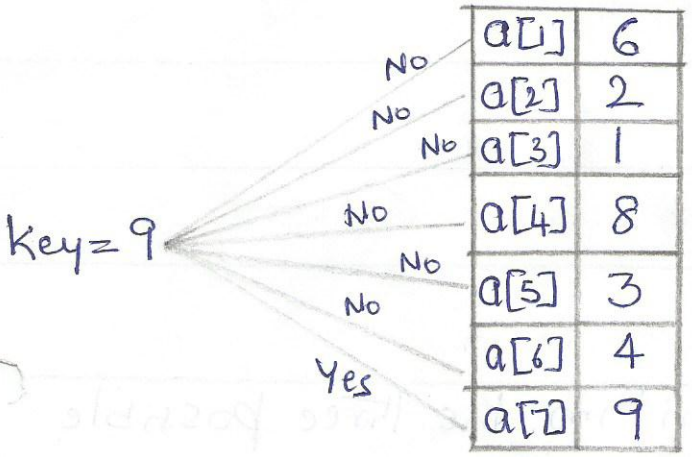
Since n^2 is the time taken in average or worst case, Θ or O notations can be used to represent time efficiency. So

$$\boxed{T(n) = \Theta(n^2)}$$

IV. LINEAR OR SEQUENTIAL SEARCH

Sequential or Linear Search is Bruteforce method of searching given key element among 'n' elements. The Sequential Search compares the given key element sequentially one after the other. If the key is found then Search is successful else unsuccessful.

Illustration of Sequential Search



A. M. PRASAD
Assistant Professor
Department of Compuier Science & Engineering
Dayananda Sagar College of Engineering
Bangalore - 560 078

Based on comparing key element with array elements one by one, the algorithm for sequential searer is given.

Algorithm: Algorithm SequentialSearch (a, n, key)

// I/p: array a with n elements

// key: is the element to be searched

// O/p: successful if present. So algorithm returns position of element else -1

```
for i ← 1 to n do
    begin   if (key = a[i]) then
                return i
            Endif
    End for
return -1
```

Time Efficiency

In search algorithm, the three possible time efficiencies are

i) **Best case**: if the key is found on first comparision. ie 1, can be represented as $\Omega(1)$

ii) **Worst case**: if the key is found on n^{th} comparision ie n, can be represented as $O(n)$

(iii) Average case: The key may be present in any position of the list. So total number of comparisons at any of the position is given by

$$= 1+2+3+4+\ldots\ldots+n$$

The average number of comparisons is given by

$$T(n) = \frac{1+2+3+\ldots\ldots\ldots+n}{n}$$

Mathematically, $T(n) = \frac{n(n+1)}{2n} = \frac{n+1}{2}$

Note:- Constants cannot be considered while finding the time efficiency

$$T(n) = n$$

A. M. PRASAD
Assistant Professor
Department of Computer Science & Engineering
Dayananda Sagar College of Engineering
Bangalore - 560 078

Since it is average case, time efficiency can be represented using Θ notation. So

$$T(n) = \Theta(n)$$

Advantages: 1. Suitable for unsorted list
2. simple and easy.

Disadvantages: 1. not Efficient

V. BRUTE FORCE STRING MATCHING

The string matching can be defined as

"Given string called text with n characters and another string called pattern with m characters, it is required to search for the string pattern in the string text"

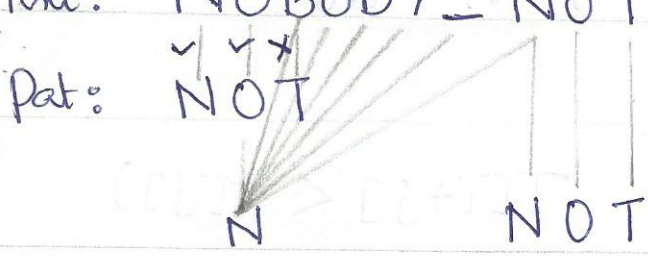
To start with first characters of both the strings are compared, if mismatch, compare the second character of the text string with first character of pattern string and so on. If both matches then shift one character right in both the string and compare the characters. If the number of matches are equal to length of pattern string 'm' then string is found otherwise not found.

For example, consider text string.

Text: NOBODY_NOTICED_HIM

Pattern String

Pat: NOT



A. M. PRASAD
Assistant Professor
Department of Compuier Science & Engineering
Dayananda Sagar College of Engineering
Bangalore - 560 078

In the above example first two charcters of both arrays are 'N' so shift right by one charauter and compare the charauters, both one 'O' then again shift right by one clarauter and compare them both are not matched.

Now compare the number of Corrent matches, its 2 not equal to m so compare the Second charauter of text and first charauter of pattren. and repeat the same.

Algorithm: Algorithm BF Stringmatching (T, P, m, n)

// T is the array contains text String

// P is the array containg pattren string

// m is the length of pattren string

// n is the length of Text string

```
for i ← 1 to n-m do
begin
    j ← 0
    while (j<m and T[i+j] < P[j])
        j ← j+1
        if (j=m) return i
    endwhile
    End for
    return -1
```

Best Case: The Best case occures if the pattern string is present in the begining. So the number of comparisions are m. The time complexity is given by $T(m) = \Omega(m)$

Worst Case: The basic operation is $T[i+j] = P[j]$, the number of times it is executed depends upon for and while loop

for $i \leftarrow 0$ to $n-m$ do

for $j \leftarrow 0$ to $m-1$ do

if (pat[j] = text[i+j])

$$T(n) = \sum_{i=0}^{n-m} \sum_{j=0}^{m-1} 1$$

$$= \sum_{i=0}^{n-m} m-1-0+1$$

$$= \sum_{i=0}^{n-m} m$$

$$= m \sum_{i=0}^{n-m} 1$$

UB − LB + 1

$$= m(n-m-0+1)$$

$$= mn - m^2 + m$$

$$= mn$$

mn is higher compared m^2 because normally n is greater than m

Since it is worst case, O-notation is used

$$T(n) = O(mn)$$

A. M. PRASAD
Assistant Professor
Department of Compuier Science & Engg.
Dayananda Sagar College of Engineering
Bangalore - 560 078

VI

Exhaustive Search

The Brute force method can be used to solve combinatorial problems called exhaustive search. The solution for these problems consumes too much time. The goal of exhaustive search is to search all possible solutions and obtain optimal solution out of all possible solutions.

Some of the problem that involve exhaustive search are

1. Travelling Salesman Problem
2. knapsack Problem
3. Assignment Problem

1. Travelling Salesman Problem (TSP)

The Tsp can be defined as

" Given n cities, a salesperson start from specified called source and has to visit all the cities and comeback to same city. The objective of this problem is to find a route through the cities that minimizes the cost"

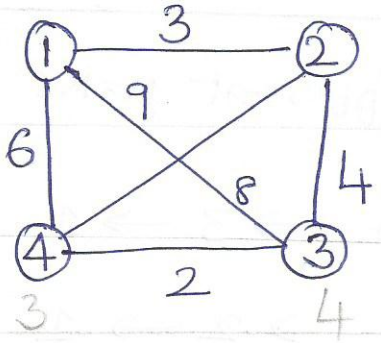
This problem can be modeled by weighted graph as shown below

* The Vertices of graph represent cities
* weight associated with edges represent the distances between two cities or cost involved while travelling from one city to other.

A. M. PRASAD
Assistant Professor
Department of Computer Science & Engineering
Dayananda Sagar College of Engineering
Bangalore - 560 078

The Brute force approach is find all possible paths from source to itself by visiting all cities and select the shortest among them. The disadvantage is, the execution time increases rapidly as the size of graph or Tsp problem increases. For example if number of cities are 16 then the number of possible routes are 1,307,674,368,000.

Example: Consider the graph given



The graph can be represented by cost adjacency matrix

	1	2	3	4
1	0	3	6	8
2	3	0	9	4
3	6	9	0	2
4	8	4	2	0

If salesperson starts from city 1, the various routes which he can visit all cities without repeating same city and returns back to start city 1 are

$1 \xrightarrow{3} 2 \xrightarrow{9} 3 \xrightarrow{2} 4 \xrightarrow{8} 1 = 22$

$1 \xrightarrow{3} 2 \xrightarrow{4} 4 \xrightarrow{2} 3 \xrightarrow{6} 1 = 15$

$1 \xrightarrow{6} 3 \xrightarrow{9} 2 \xrightarrow{4} 4 \xrightarrow{8} 1 = 27$

$1 \xrightarrow{6} 3 \xrightarrow{2} 4 \xrightarrow{4} 2 \xrightarrow{3} 1 = 15$

$1 \xrightarrow{8} 4 \xrightarrow{4} 2 \xrightarrow{9} 3 \xrightarrow{6} 1 = 27$

$1 \xrightarrow{8} 4 \xrightarrow{2} 3 \xrightarrow{9} 2 \xrightarrow{3} 1 = 22$

Now, we have to find out smallest among all costs.

It is 15 and two different paths ie

$1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$ and

$1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 1$

So travelling salesman can select any one among two.

Time Efficiency: In any complete graph the number of paths from source node to other all nodes and back to source node is $(n-1)!$.

For example if $n=4$, n is number of Vertices then different paths are Six.

if $n=3$ the different paths are 2

So, in general, for n cities, the number of routes are $(n-1)!$. Therefore complexity is given by

$$T(n) = \theta(n-1)! \quad \text{or}$$

$$T(n) = \theta(n!)$$

A. M. PRASAD
Assistant Professor
Department of Computer Science & Engineering
Dayananda Sagar College of Engineering
Bangalore - 560 078

2. Knapsack Problem

The Knapsack Problem can be defined as

" Given knapsack of capacity M and n objects of weights $w_1, w_2, \ldots w_n$ with profits $P_1, P_2 \ldots P_n$. The goal is to place objects into knapsack so that

maximum profit is obtained and total weight of objects inside knapsack should not exceed the capacity of knapsack".

Consider the example, the capacity M=10 and n=4, weights are profits are

$w_1 = 7$	$P_1 = 42$
$w_2 = 3$	$P_2 = 12$
$w_3 = 4$	$P_3 = 40$
$w_4 = 5$	$P_4 = 25$

The Brute force technique is to place all the combi-nations of weights and take the sum of profits of weights placed. The different combinations are

Subset	Total weight	Total value
$\emptyset$	0	0
{1}	7	42
{2}	3	12
{3}	4	40
{4}	5	25
{1,2}	10	36

{1, 3}	11	not feasible
{1, 4}	12	not feasible
{2, 3}	7	52
{2, 4}	8	37
{3, 4}	9	65
{1, 2, 3}	14	not feasible
{1, 2, 4}	15	—11—
{1, 3, 4}	16	—11—
{2, 3, 4}	12	—11—
{1, 2, 3, 4}	19	—11—

Now, it is to find biggest among all Total values. The biggest is the Optimal Solution.

A. M. PRASAD
Assistant Professor
Department of Compuier Science & Engineering
Dayananda Sagar College of Engineering
Bangalore - 560 078

Time Efficiency

If 'n' is the number of objects, then total number of subsets are 2^n. So time complexity of knapsack problem is

$$T(n) = \Theta(2^n)$$

3. Assignment Problem

Assignment Problem can be defined as

" Given n jobs $\langle j_1, j_2, \ldots j_n \rangle$ and n-persons $\langle P_1, P_2, \ldots P_n \rangle$, it is required to assign all 'n' jobs to all 'n' persons and cost involved should be minimum".

The Bruteforce solution is to find all permutations of 'n' jobs to 'n' people. For each permutation obtain total cost and select minimum among all. That is the optimal solution.

Example: Consider the given Assignment Problem

	J1	J2	J3	J4
P1	10	3	18	9
P2	7	5	4	8
P3	6	9	12	9
P4	8	7	10	15

The various instances of problem's solution are

(1, 2, 3, 4) Cost = 10+5+12+15 = 42

(1, 2, 4, 3) Cost = 10+5+9+10 = 34

(1, 3, 2, 4) Cost = 10+4+9+15 = 38

(1, 3, 4, 2) cost = 10+4+9+7 = 30

(1, 4, 2, 3) cost = 10+8+9+10 = 37

(1, 4, 3, 2) cost = 10+8+12+7 = 37

Similarly it is required to calculate

(2,1,3,4), (2,1,4,3), (2,3,14) (2,3,4,1)

(2,4,1,3), (2,4,3,1)

(3,1,2,4), (3,1,4,2) (3,2,1,4), (3,2,4,1) (3,4,1,2)

(3,4,2,1)

(4,1,2,3), (4,1,3,2), (4,2,1,3) (4,2,3,1), (4,3,1,2)

(4,3,2,1)

A. M. PRASAD
Assistant Professor
Department of Compuier Science & Engineering
Dayananda Sagar College of Engineering
Bangalore - 560 078

Out of all above solutions, the solutions that takes least cost should be considered.

Time Efficiency: If there are 'n' jobs and 'n' people, the total number of combinations are n!

So time complexity of Assignment Problem is

$$T(n) = \Theta(n!)$$

Question Bank

1. Write a algorithm for selection sort method. show that its worst case time Efficiency is $O(n^2)$ [June-July 08]
2. Sort the following list using bubble sort 66, 55, 44, 33, 22, 11 in ascending order [June-July 08]
3. With an example, show how an exhaustive search can be applied for travelling salesman problem [June-July 08]
4. What is brute force method? Explain sequential search algorithm with an example. Analyze its time Efficiency [Dec 08-Jan 09]