ADVANCED ALGORITHMS LAB

ASSIGNMENT - 3

QUICK SORT

Name - AMAR PARAMESWARAN

Class - TYCS A

PRN - 19070122016

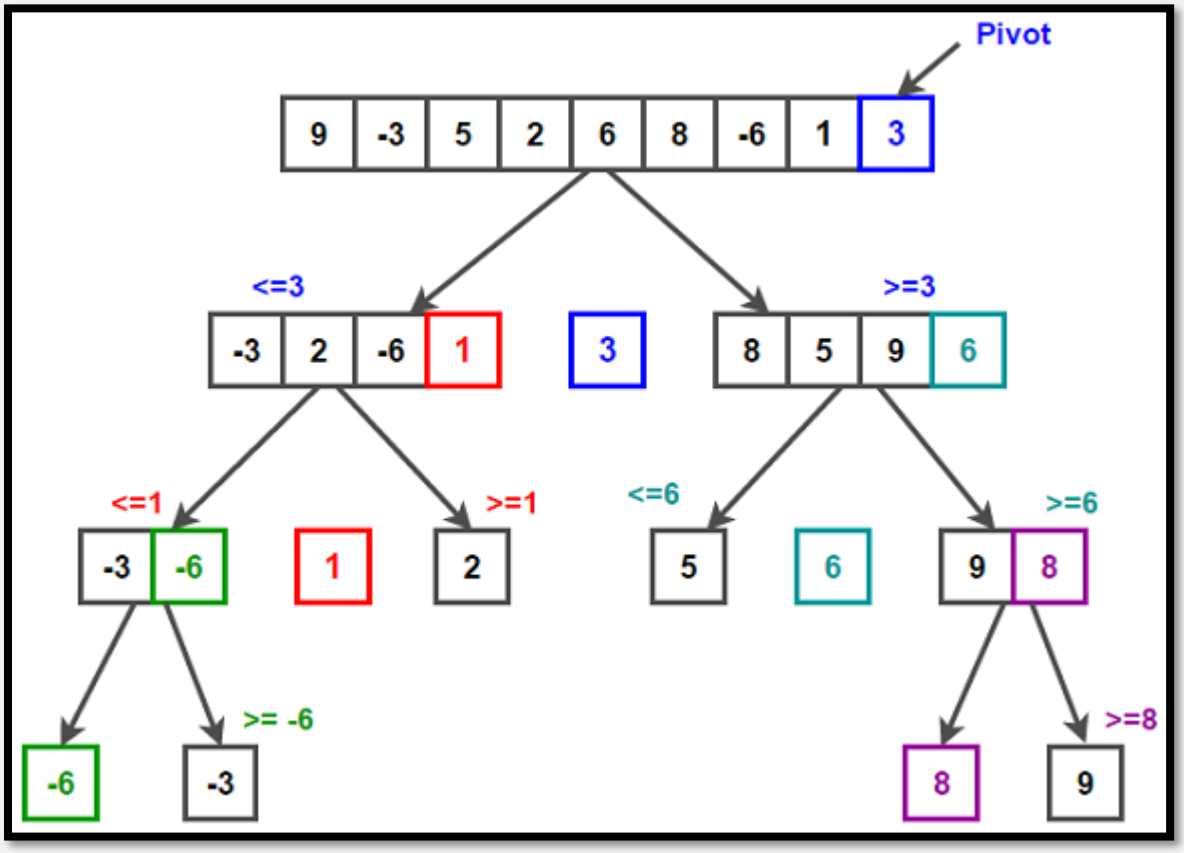
Advanced Algorithm Lab Assignment No: 3

Name of the Student	PRN	Batch	TY CS/IT Division
Amar Parameswaran	19070122016	TY CS	A

WORKING MECHANISM

The array is divided into 2 parts depending on the pivot element which basically means elements lesser than pivot element is present in the left subarray and elements bigger than pivot element are in the right subarray.

EXAMPLE:



Pseudo Code:

```
Pseudo Code
quick_sort(array, start, end)
{
  if(start<end)
  {
    pivot_index = partition(array, start, end);
    quick_sort(array, start, pivot_index-1);
    quick_sort(array, pivot_index+1, end);
  }
}
```

ITERATIVE

Code Snippet:

```
void QuickSort_Iterative(int s,int e) //Iterative
{
    //Stack to store the indexes of array
    int stack[MAX];
    int top=-1;
    stack[++top]=s;
    stack[++top]=e;

    while(top>=0)
    {
        e=stack[top--];
        s=stack[top--];

        if(s<e)
        {
            int p=Divide(s,e);
            stack[++top]=s;
            stack[++top]=p-1;
            stack[++top]=p+1;
            stack[++top]=e;
        }
    }
}
```

Output:

```
  ____        _      _        _____                _
 / __ \      (_)    | |      / ____|              | |
| |  | |_   _ _  ___| | __ _| (___   ___  _ __| |_
| |  | | | | | |/ __| |/ /  \___ \ / _ \| '__| __|
| |__| | |_| | | (__|   <   ____) | (_) | |  | |_
 \___\_\\__,_|_|\___|_|\_\ |_____/ \___/|_|   \__|

Enter the number of elements in the array : 9

Enter the elements of the array
Element 1: 9
Element 2: -3
Element 3: 5
Element 4: 2
Element 5: 6
Element 6: 8
Element 7: -6
Element 8: 1
Element 9: 3

The array is NOT sorted
9 -3 5 2 6 8 -6 1 3

Which sorting algorithm do you want to use? - (1)Iterative or (2)Recursive - 1

*** Iterative Quick Sort ***

Pass 1: -3 2 -6 1 3 8 5 9 6
Pass 2: -3 2 -6 1 3 5 6 9 8
Pass 3: -3 2 -6 1 3 5 6 8 9
Pass 4: -3 -6 1 2 3 5 6 8 9
Pass 5: -6 -3 1 2 3 5 6 8 9

The sorted array is : -6 -3 1 2 3 5 6 8 9

*****************************************
```

RECURSIVE

Code Snippet:

```
void QuickSort_Recursive(int s,int e) //Recursive
{
    if(s<e)
    {
        int p=Divide(s,e);
        QuickSort_Recursive(s,p-1);   //Sort 1st half
        QuickSort_Recursive(p+1,e);   //Sort 2nd half
    }
}
```

Output:

```
  ____        _      _       _____                _
 / __ \      (_)    | |     / ____|              | |
| |  | |_   _ _  ___| | __ | (___   ___  _ __| |_
| |  | | | | | |/ __| |/ /  \___ \ / _ \| '__| __|
| |__| | |_| | | (__|   <   ____) | (_) | |  | |_
 \___\_\\__,_|_|\___|_|\_\ |_____/ \___/|_|   \__|

Enter the number of elements in the array : 9

Enter the elements of the array
Element 1: 9
Element 2: -3
Element 3: 5
Element 4: 2
Element 5: 6
Element 6: 8
Element 7: -6
Element 8: 1
Element 9: 3

The array is NOT sorted
9 -3 5 2 6 8 -6 1 3

Which sorting algorithm do you want to use? - (1)Iterative or (2)Recursive - 2

*** Recursive Quick Sort ***

Pass 1: -3 2 -6 1 3 8 5 9 6
Pass 2: -3 -6 1 2 3 8 5 9 6
Pass 3: -6 -3 1 2 3 8 5 9 6
Pass 4: -6 -3 1 2 3 5 6 9 8
Pass 5: -6 -3 1 2 3 5 6 8 9

The sorted array is : -6 -3 1 2 3 5 6 8 9

*****************************************
```

Link for code: https://github.com/Amar1709/AdvancedAlgos_Practice.git

AMAR PARAMESWARAN
TYCSA
19070122016

QUICK SORT

★ TIME COMPLEXITY

- AVERAGE TIME COMPLEXITY ⇒ $O(N(\log(N))$

T(N) = Time complexity of Quick Sort for input size N

Also, the input size of N is broken into 2 parts

J and N-J

$T(N) = T(J) + T(N-J) + M(N)$

Here M(N) = Time complexity of finding pivot element for N elements

On solving T(N), we get $O(N \log N)$

- BEST CASE TIME COMPLEXITY ⇒ $O(N \log N)$

T(N) - Time complexity for best case

$T(N) = 2 * T(N/2) + c * N$ (where c is a constant)

Now,

$T(N) = 2 * (2 * T(N/4) + c * N) + c * N$

We can say that,

$T(N) = 2^k T(N/2^k) + k * c * N$ (where k, c → constants)

$\Rightarrow N = 2^k$

$\therefore k = \log_2 N$

Therefore, $T(N) = N * T(1) + N * \log N = O(N \log N)$

AMAR PARAMESWARAN
TYCSA
19070122016

- WORST CASE TIME COMPLEXITY ⇒ $O(N^2)$

$T(N)$ → Time complexity for worst case

$T(N) = T(N-1) + c*N$ where $c \rightarrow$ constant

Now,

$T(N) = T(N-2) + c*(N-1) + c*N$

$= T(N-2) + c*(2N-1)$

$= T(N-3) + c*(3N-3)$

We can say that,

$T(N) = T(N-k) + k*c*N - c*((k-1)+\ldots+2+1)$

$= T(N-k) + k*c*N - c*(k(k-1)/2)$

Put $n=k$, we get,

$T(k) = T(0) + c*N^2 - c*(N(N-1)/2)$

removing the constant terms,

$T(N) = N^2 - N(N-1)/2$

$\therefore T(N) = O(N^2)$

★ SPACE COMPLEXITY ⇒ $O(N)$

As no other container (array) is created other than the given array therefore the space complexity will be in the order of N i.e $O(N)$

AMAR PARAMESWARAN
TYCSA
19070122016

★ UNIQUE CHARACTERISTICS

- It is an unstable sort
- It is the best for sorting
- Time complexity is O(NlogN)
- Space complexity is O(N)

★ REAL LIFE APPLICATIONS

- Used in Operation Research & event driven simulations.
- Used in numerical computations due to its speedy nature.
- Commercial computing.

★ OPTIMIZATIONS

- Hand code smaller sorts
- Tune the pasting function /loop.
- Better pivot value
- Lots of equal values in the array.