Quick Sort

Implementation:

```
#include<bits/stdc++.h>

using namespace std;

int Partition(int arr[], int lb, int ub)
{
  int pivot = arr[lb];
  int start = lb;
  int ending = ub;

  while(start<ending)
  {
     while(arr[start]<=pivot){
      start++;
     }
     while(arr[ending]>pivot){
      ending--;
     }
     if(start<ending)
     {
        swap(arr[start],arr[ending]);
     }
  }
  swap(arr[lb],arr[ending]);
  return ending;
}
void QuickSort(int arr[], int lb, int ub)
{
   int loc;
   if(lb<ub){
      loc = Partition(arr,lb,ub);
      QuickSort(arr,lb,loc-1);
      QuickSort(arr,loc+1,ub);
   }
}

int main()
{
 int n;
 cin>>n;
 int arr[n];
 for(int i=0; i<n; i++)
```

Md. Foysal Ahmed
ID: 191-15-12486

```
{
   cin>>arr[i];
}
int lb,ub;
lb = 0;
ub = n-1;
QuickSort(arr,lb,ub);
for(int i=0; i<n; i++)
{
   cout<<arr[i]<<" ";
}
cout<<endl;
```
}**Analysis:**

We start analyzing the Insertion Sort procedure with the time "cost" of each statement and the number of times each statement is executed.

pseudo code of merge sort:

	MERGE_SORT(*A,lb,ub*)	**Cost**	**Time**
1	if(lb<ub)	c1	1
2	loc <- Partiton(A,lb,ub)	c2*n + c3	
3	Quicksort(A,lb,loc-1)	T(n/2)	
4	Quicksort(A,loc+1,ub)	T(n/2)	

The partition function will cost the same as linear function which is $an + b$ as the partition function is checking the pivot with other elements and swapping the will cost asymptotically $T(n) = c * n + c'$.

The running time of the algorithm is the sum of running times for each statement executed; a statement that takes c_i steps to execute and executes n times will contribute $c_i * n$ times to the total running time.

$$T(n) = \begin{cases} c, & if\ n = 1 \\ 2T\left(\frac{n}{2}\right) + (c1 + c3) + (c2) * n \end{cases}$$

This equation can be written as,

$$T(n) = 2T\left(\frac{n}{2}\right) + cn + c' \qquad \text{[where c and c' are some constant]}$$

When the value of n is grater then 1 then constant c' become negligible (n>1)

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

We can find out T(n/2) to substitute T(n),

$$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + c\left(\frac{n}{2}\right)$$

Substituting,

Md. Foysal Ahmed
ID: 191-15-12486

$$T(n) = 2\left\{2T\left(\frac{n}{4}\right) + c\left(\frac{n}{2}\right)\right\} + cn$$
$$T(n) = 4T\left(\frac{n}{4}\right) + 2c\left(\frac{n}{2}\right) + cn$$
$$T(n) = 4T\left(\frac{n}{4}\right) + cn + cn$$
$$T(n) = 4T\left(\frac{n}{4}\right) + 2cn$$

Again,

$T\left(\frac{n}{4}\right) = 4T\left(\frac{n}{16}\right) + 2c\left(\frac{n}{4}\right)$

Substituting,

$$T(n) = 4\left\{4T\left(\frac{n}{16}\right) + 2c\left(\frac{n}{4}\right)\right\} + 2cn$$
$$T(n) = 16T\left(\frac{n}{16}\right) + 8c\left(\frac{n}{4}\right) + 2cn$$
$$T(n) = 16T\left(\frac{n}{16}\right) + 2cn + 2cn$$
$$T(n) = 16T\left(\frac{n}{16}\right) + 4cn$$
$$\vdots$$
$$T(n) = 2^k\, T\left(\frac{n}{2^k}\right) + kcn$$

Assume,

$$\frac{n}{2^k} = 1$$
$$\Rightarrow 2^k = n$$
$$\Rightarrow k = \log_2 n$$

Therefore,

$$T(n) = 2^{\log_2 n}\, T(1) + \log_2 n * cn$$
$$= n * c + cn * \log_2 n \qquad [T(1) = c]$$

Therefore the Time function in best case would be $T(n) = n * c + cn * \log_2 n$

pivot

1	2	3	4	5	6	7

If in the Array an unbalanced partition occurs such that only larger values are positioned after pivot element or smaller values are positioned before pivot element and we can't divide the array into two parts in that case the worst case occurs and the time function will be,

$$T(n) = T(n-1) + c * n$$

We can find out T(n-1) to substitute T(n),

Md. Foysal Ahmed
ID: 191-15-12486

$$T(n-1) = T(n-2) + c(n-1)$$

Substituting,

$$T(n) = T(n-2) + c(n-1) + c * n$$
$$T(n) = T(n-2) + cn - c + cn$$
$$T(n) = T(n-2) + 2cn - c$$

Again,

$$T(n) = T(n-3) + c(n-2) + 2cn - c$$
$$T(n) = T(n-3) + cn - 2c + 2cn - c$$
$$T(n) = T(n-3) + 3cn - 3c$$

Again,

$$T(n) = T(n-4) + 4cn - 6c$$

.

.

.

$$T(n) = T(n-k) + kcn - \big(1 + 2 + 3 + \cdots + (k-1)\big)c$$

$$\big(1 + 2 + 3 + \cdots + (k-1)\big) = \frac{k(k-1)}{2}$$

Therefore,

$$T(n) = T(n-k) + kcn - \tfrac{k(k-1)}{2}\mathrm{c}$$

Let, $n - k = 1$

$n = k$

Therefore,

$$T(n) = T(1) + c^2 n - \frac{n(n-1)}{2}\mathrm{c}$$
$$= c + cn\left(n - \frac{n-1}{2}\right)$$
$$= c + cn\left(\frac{(2n - n + 1)}{2}\right)$$
$$= c + cn\left(\frac{n+1}{2}\right)$$

$$T(n) = \frac{cn^2}{2} + \frac{cn}{2} + \mathrm{c}$$

Md. Foysal Ahmed
ID: 191-15-12486

Time Complexity:

Worst Case: The time function we get from the analysis is $T(n) = \frac{cn^2}{2} + \frac{cn}{2} + \text{c}$

So the complexity of quick sort in worst case is $O(n^2)$

Best Case: The time function we get from the analysis is $T(n) = \ cn + cn \log_2 n$

So the complexity of quick sort in best case is $O(n \log n)$

Md. Foysal Ahmed
ID: 191-15-12486