Analysis of Algorithms | Set 2 (Worst, Average and Best Cases) – By GeeksForGeeks

In the set, we discussed how Asymptotic analysis overcomes the problems of naive way of analyzing algorithms. In this post, we will take an example of Linear Search and analyze it using Asymptotic analysis.
We can have three cases to analyze an algorithm:
1) Worst Case
2) Average Case
3) Best Case

Let us consider the following implementation of Linear Search.

```
#include <stdio.h>

// Linearly search x in arr[].  If x is present then return the index,
// otherwise return -1
int search(int arr[], int n, int x)
{
    int i;
    for (i=0; i<n; i++)
    {
       if (arr[i] == x)
         return i;
    }
    return -1;
}

/* Driver program to test above functions*/
int main()
{
    int arr[] = {1, 10, 30, 15};
    int x = 30;
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("%d is present at index %d", x, search(arr, n, x));

    getchar();
    return 0;
}
```

Worst Case Analysis (Usually Done)

In the worst case analysis, we calculate upper bound on running time of an algorithm. We must know the case that causes maximum number of operations to be executed. For Linear Search, the worst case happens when the element to be searched (x in the above code) is not present in the array. When x is not present, the search() functions compares it with all the elements of arr[] one by one. Therefore, the worst case time complexity of linear search would be Θ(n).

Average Case Analysis (Sometimes done)

In average case analysis, we take all possible inputs and calculate computing time for all of the inputs. Sum all the calculated values and divide the sum by total number of inputs. We must know (or predict)

distribution of cases. For the linear search problem, let us assume that all cases are uniformly distributed (including the case of x not being present in array). So we sum all the cases and divide the sum by (n+1). Following is the value of average case time complexity.

$$\text{Average Case Time} = \frac{\sum_{i=1}^{n+1} \theta(i)}{(n+1)}$$

$$= \frac{\theta((n+1)*(n+2)/2)}{(n+1)}$$

$$= \Theta(n)$$

Best Case Analysis (Bogus)

In the best case analysis, we calculate lower bound on running time of an algorithm. We must know the case that causes minimum number of operations to be executed. In the linear search problem, the best case occurs when x is present at the first location. The number of operations in the best case is constant (not dependent on n). So time complexity in the best case would be Θ(1)
Most of the times, we do worst case analysis to analyze algorithms. In the worst analysis, we guarantee an upper bound on the running time of an algorithm which is good information.
The average case analysis is not easy to do in most of the practical cases and it is rarely done. In the average case analysis, we must know (or predict) the mathematical distribution of all possible inputs.
The Best Case analysis is bogus. Guaranteeing a lower bound on an algorithm doesn't provide any information as in the worst case, an algorithm may take years to run.
For some algorithms, all the cases are asymptotically same, i.e., there are no worst and best cases. For example, Merge Sort. Merge Sort does Θ(nLogn) operations in all cases. Most of the other sorting algorithms have worst and best cases. For example, in the typical implementation of Quick Sort (where pivot is chosen as a corner element), the worst occurs when the input array is already sorted and the best occur when the pivot elements always divide array in two halves. For insertion sort, the worst case occurs when the array is reverse sorted and the best case occurs when the array is sorted in the same order as output.

References:

MIT's Video lecture 1 on Introduction to Algorithms.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.