Performance Analysis Vs Performance Measurement

Performance of an algorithm is measured in terms of time complexity and space complexity

Time complexity Analysis

There are two approaches to determine time complexity

- Theoretical Analysis
- Experimental study/Empirical Analysis

Theoretical Analysis

Theoretical Analysis is evaluation of an Algorithm prior to its implementation on the actual machine so it is machine independent and it helps to compare algorithms irrespective of the machine configuration on which the algorithm is intended to run

Experimental Analysis

This is posterior evaluation of an algorithm. The algorithm is implemented and run on actual machine for different inputs to understand the relation between execution time and input size. This method for determining the performance of an algorithm is machine dependent

Performance Analysis of Sequential Search algorithm.

ALGORITHM SequentialSearch(A[0..n-1], K)

//Searches for a given value in a given array by sequential search

//Input: An array A[0..n-1] and a search key K

//Output: Returns the index of the first element of A that matches K or –1 if there are no matching elements

```
    i ←0
    while i < n and A[i] ǂ K do
        i ← i + 1
    if i < n
        return i
    else
        return -1
```

Basic operation A[i] ǂ K

Basic operation count n (for worst case)

$\mathbf{T(n)_{worst\ case} \in O(n)}$

Performance Measurement for sequential search algorithm

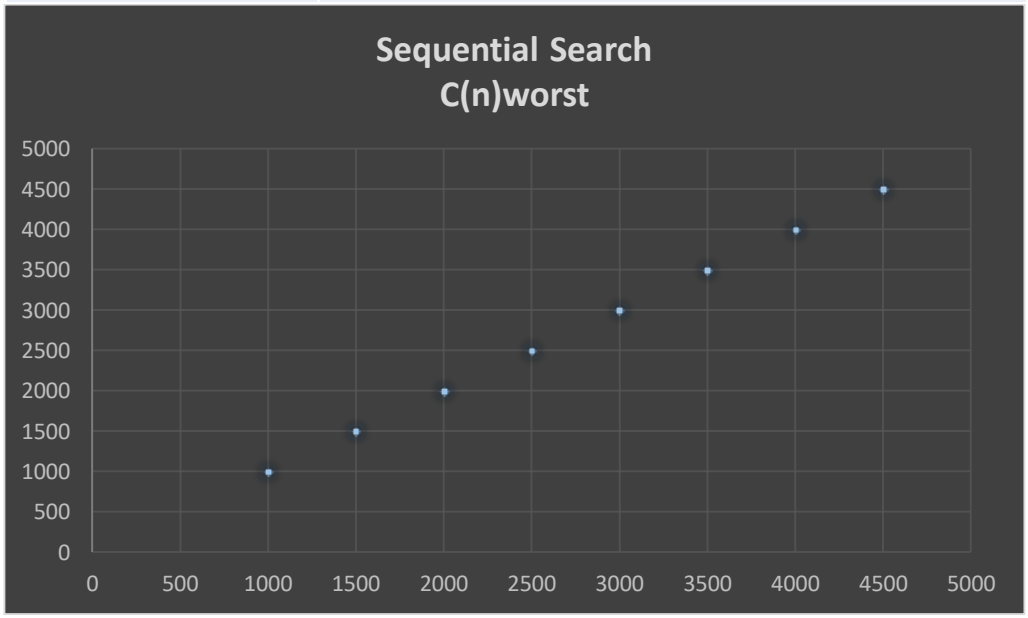
```
#include<stdio.h>
#include<stdlib.h>
#include<sys/time.h>
int getrand(int a[], int n)
{
        int i;
         for(i=0;i<n;i++)
         {
                 a[i]=rand()%10000
         }
}
int search(int arr[], int n,int x,int *count)
{
        int i;
        for(i=0;i<n;i++)
        {
     count=count+1;
                if(arr[i]==x)
                        return i;
        }
        return -1;
}
```

```
int main()
{
	int a[10000],i,res,count;
	double elapse,start,end;
	struct timeval tv;
	FILE *fp1,*fp2;
	fp1=fopen("seqtime.txt","w");
	fp2=fopen("seqcount.txt","w");
	int key;
	for(i=500;i<=10000;i+=500)// size of the array to be created
	{
		getrand(a,i);
		key=a[i-1];
		count=0;
		gettimeofday(&tv,NULL);
		start=tv.tv_sec+ tv_usec/100000//start time
		res=search(a,i,key,&count);
		gettimeofday(&tv,NULL);
		end=tv.tv_sec+ tv_usec/100000//end time
		elapse=(end-start)*1000
		fprintf(fp1,"%d\t%lf\n",i,elapse);
		fprintf(fp2,"%d\t%d\n",i,count);
	}
	fclose(fp1);
	fclose(fp2);
	return 0;
```

}

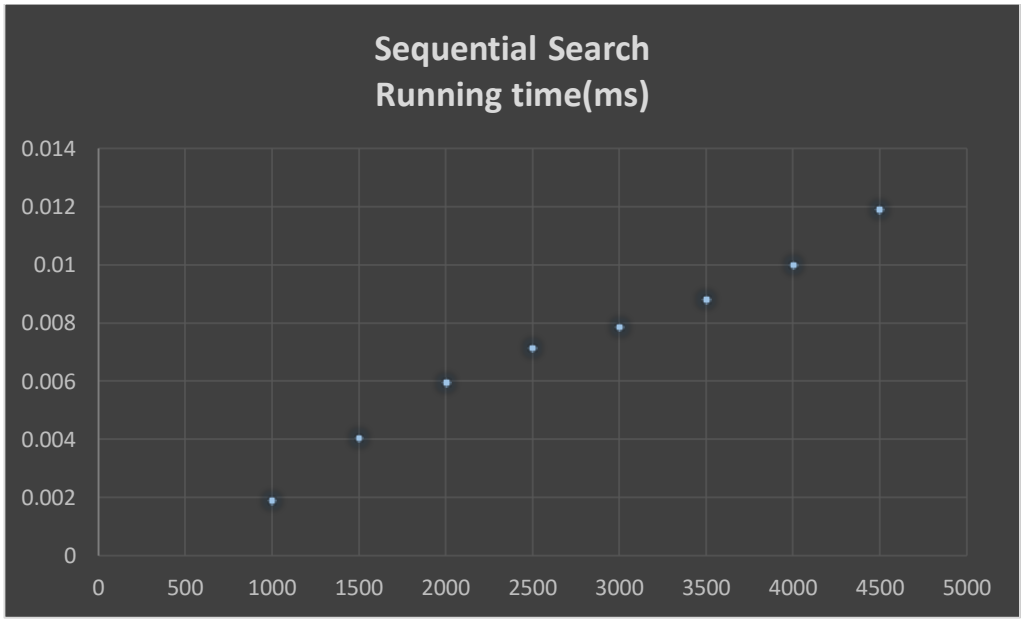
Number of key comparisons for inputs of different sizes

Input Size	Sequential Search C(n)worst
1000	1000
1500	1500
2000	2000
2500	2500
3000	3000
3500	3500
4000	4000
4500	4500



Actual running time for inputs of different sizes

Input Size	Sequential Search Actual Running Time(ms)
1000	0.001907
1500	0.004053
2000	0.00596
2500	0.007153
3000	0.007868
3500	0.008821
4000	0.010014
4500	0.011921



Time complexity (worst case) of sequential search is ***linear*** in terms of length of the list of elements