Analysis of N-ary Search

Upasana Ghosh

Roll no: UE143110
U.I.E.T, Panjab University
Email: ghoshupasana05@gmail.com

Aim

This paper presents the performance analysis of n-ary search algorithm with different value of n (partitions) which is based on divide and conquer strategy.

Approach

In n-ary search algorithm, array is partitioned into n equal parts recursively until the required key is obtained. For the analysis, array of a certain size is taken and it is filled with random integer values generated by random function. All the array elements are then sorted. The array is then divided into n number of partitions where n begins from two and increases consecutively till it reaches the size of the array. N-ary search algorithm is used and the average time taken to search a random element in the array is calculated.

Experiment

An array of size 100 is taken and is filled with 100 random values. The array is then sorted using bubble sort algorithm. A key is generated randomly from the array elements and it is then searched by n-ary search algorithm with different portioning values. The number of partitions begins from 2 and it keeps on increasing till it reaches the size of array that is 100. For every partitioning value, n, the array elements and the key are generated 5,00,000 times and searching operation is performed on it. The average time for searching in each partition is calculated. The result (number of partitions and the average time taken for searching) is stored in a text file and a graph is plotted from the result. The platform used for compilation and execution is Ubuntu 15.10 and the graph is plotted using octave-3.6.4

I. Result

Table 1 and figure 1 depicts the results of n-ary search. Table 1 shows the partitioning number (n) and the corresponding time taken (average) in seconds for searching the key. The result is depicted graphically in figure 1.

Discussion

In n-ary search, initially time taken to search the key decreases with increase in the number of partitions. But after certain partitioning value, the time taken to search increases with increase in the number of partitions. As the result is performed on random values, there may be difference in the number of worst cases, average cases and best cases

TABLE I

No. of partitions(n)	time(s)	No. of partitions(n)	time(s)
2	0.062835	52	0.086741
4	0.06283	54	0.087356
6	0.063375	56	0.088629
8	0.05815	58	0.088545
10	0.056426	60	0.089961
12	0.057658	62	0.091025
14	0.059488	64	0.092841
16	0.059838	66	0.092921
18	0.0608	68	0.094459
20	0.061996	70	0.096317
22	0.063729	72	0.096498
24	0.065304	74	0.097636
26	0.065306	76	0.097555
28	0.066779	78	0.099354
30	0.068496	80	0.098516
32	0.069387	82	0.100209
34	0.073956	84	0.101498
36	0.075221	86	0.099844
38	0.07585	88	0.099682
40	0.076543	90	0.100819
42	0.077323	92	0.102121
44	0.079435	94	0.10334
46	0.078702	96	0.101076
48	0.079377	98	0.102603
50	0.079914	100	0.104633

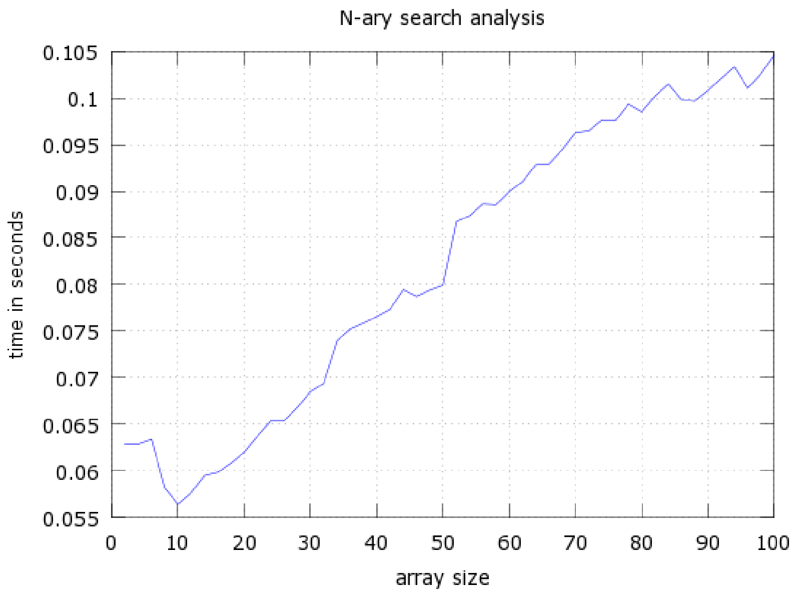

Fig. 1.

with different partitioning value. In this experiment, initially the time taken decreases with the increase in number of partitions.At n=4, time taken to search a key is almost similar to that for n=2 because the number of worst cases dominates the number of average cases. At n=10, the performance of n-ary search algorithm is maximum as the average time taken to search is 0.056426s, which is the global minima in the graph 1. But after this, the performance decreases with the increase in number of partitions, although there are some local minima and maxima in certain points in the graph.

Conclusion

The performance of n-ary search initially increases with increase in number of partitions. After a certain partitioning value, n, the performance decreases and the time to search increases. Hence, for most efficient search, use the partitioning value which yields the maximum performance.