BLG 335E – Analysis of Algorithms I

Ekin ÇELEBİ

December 10, 2020

1 Part 2. Report

1.1 Asymptotic Bounds

1.1.1 Worst Case

The worst case happens when the pivot chosen by the partition function is always either the smallest or the greatest element. As a result one of the partitions is empty and the other contains n-1 elements. Since side of partition always has no elements.

$$T(n) = T(0) + T(n-1) + \theta(n)$$

The first call takes cn time, the second call takes c(n-1) time, the third call takes c(n-2) time and recurses untill n=2. We sum up all of the partitioning times, as a result of arithmetic series we observe:

$$c((n+1)(n/2) - 1)$$

. In big-O notation we eliminate the low-order terms and the constant coefficients. We get the solution of the recurrence as:

$$O(n^2)$$

1.1.2 Best Case

If we're lucky,partition are evenly balanced as possible. If the subarray has odd number of elements each partition has n/2 elements. If the subarray has even number of elements one of the partition has (n-1)/2 elements and the other one has n/2 elements.In both cased the partitions has at most n/2 elements. The recurrence for the best case is:

$$T(n) = 2T(n/2) + \theta(n)$$

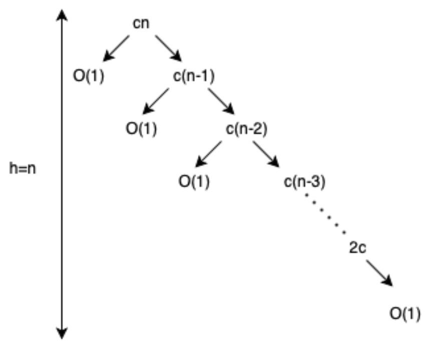


Figure 1: recursion tree for the worst case subproblems

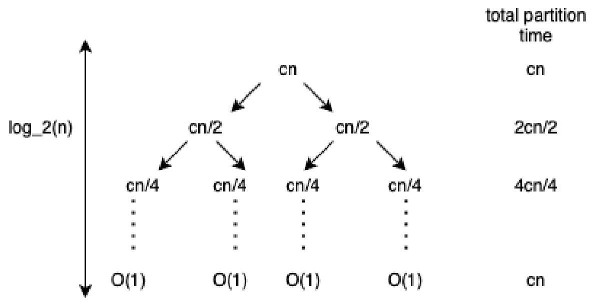


Figure 2: recursion tree for the best case subproblems

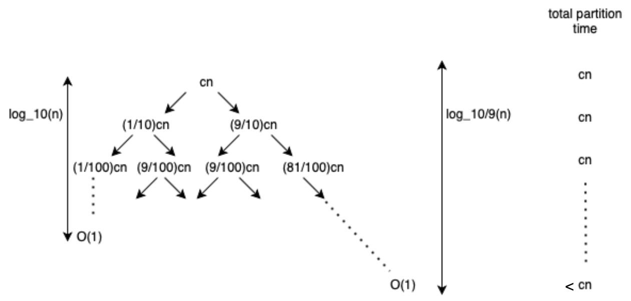


Figure 3: recursion tree for the average case subproblems

From the code the outer loop runs n times. The inner loop iterates in multiples of two, therefore the inner loop runs $log_2 n$) times. Therefore big-O notation gives us the result:

$$O(nlogn)$$

1.1.3 Average Case

For the rest of the cases we don't get balanced partitions. I consider I always get 1/10 to 9/10 partitions. The recurrence for the average case becomes:

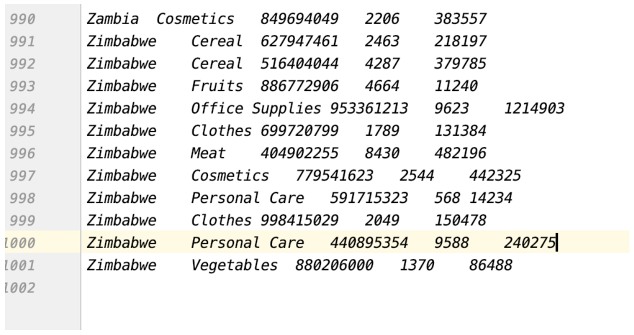
$$T(n) = T(n/9) + T(9n/10) + \theta(n)$$

We get a smaller partition on the left side at each iteration therefore it root downs faster than other levels(takes $log_{10/9} n$ levels). However on the right side the longest path takes $log_{10/9} n$ levels to root down. Total partitioning time until the level $log_{10/9} n$ is cn after that level we get partitioning times becomes smaller than cn. At total we end up with $log_{10/9} n$ levels. $log_{10/9} n$ is $log_2 n$ divided by $log_2(10/9)$ which is a constant since constants are discarded in the notation we end up with $O(nlogn)$ run time.

2 Question 2

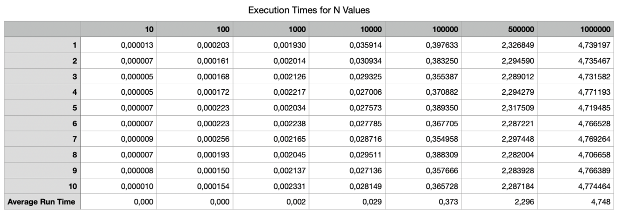
2.1 a

Sorting the data by the total profits gives us a reasonable output. However sorting the data again by the country names does not give us the desired output. During our second sort we only check for the country



```
Zambia  Cosmetics   849694049   2206    383557
Zimbabwe    Cereal  627947461   2463    218197
Zimbabwe    Cereal  516404044   4287    379785
Zimbabwe    Fruits  886772906   4664    11240
Zimbabwe    Office Supplies 953361213   9623    1214903
Zimbabwe    Clothes 699720799   1789    131384
Zimbabwe    Meat    404902255   8430    482196
Zimbabwe    Cosmetics   779541623   2544    442325
Zimbabwe    Personal Care   591715323   568 14234
Zimbabwe    Clothes 998415029   2049    150478
Zimbabwe    Personal Care   440895354   9588    240275
Zimbabwe    Vegetables  880206000   1370    86488
```

Figure 4: Question 2 simulation



Execution Times for N Values

	10	100	1000	10000	100000	500000	1000000
1	0,000013	0,000203	0,001930	0,035914	0,397633	2,326849	4,739197
2	0,000007	0,000161	0,002014	0,030934	0,383250	2,294590	4,735467
3	0,000005	0,000168	0,002126	0,029325	0,355387	2,289012	4,731582
4	0,000005	0,000172	0,002217	0,027006	0,370882	2,294279	4,771193
5	0,000007	0,000223	0,002034	0,027573	0,389350	2,317509	4,719485
6	0,000007	0,000223	0,002238	0,027785	0,367705	2,287221	4,766528
7	0,000009	0,000256	0,002165	0,028716	0,354958	2,297448	4,769264
8	0,000007	0,000193	0,002045	0,029511	0,388309	2,282004	4,706658
9	0,000008	0,000150	0,002137	0,027136	0,357666	2,283928	4,766389
10	0,000010	0,000154	0,002331	0,028149	0,365728	2,287184	4,774464
Average Run Time	0,000	0,000	0,002	0,029	0,373	2,296	4,748

Figure 5: Execution times for N values

names, any country having different total profit results are considered as equal values therefore there is no priorization applied for different values of the total profit on a same country. The results for the Zimbabwe is shared. The countries are sorted alphabetically, however the total profits are not in descending order.

2.2 b

The main problem in our case is the dublicated country names.If we are able keep the ordering of total profits stable while sorting the list for the country names we obtain successful result. We need a stable algorithm to maintain the relative ordering of duplicate elements in our data. We can use merge sort, insertion sort and bubble sort works for our case.

3 Question 3

Big-O notation is used to show the complexity regarding to the input size. It is more reasonable to have a rather small value for the increasing input size. If the value is increased drastically is it better to use alternative algorithms. In quicksort except for the worst case(in which the notation is nlogn), for the best and the average cases our notation is always nlogn. We can not make predictions about the exact growth rate however we would expect for the huge input sizes the algorithm to work much slower(we still have to consider the outer loop of the code running N times). The algorithm works very fast for the small input sizes.(Close to 0seconds up to N = 10000). Also for the big input sizes time increases but still works much faster than the algorithms having notation of n^2.

4 Question 4

4.1 Compare the results and explain the difference

The given inputs are sorted in this case. We pick the pivot from the head of the array each time and compare with the every item left in the array and we obtain the notation of n^2 coming from the worst case notation.Even for the very small size of inputs the average run time increases considerably because of the fact that the amount of computation increased a lot. The explanation lies in the comparision between the

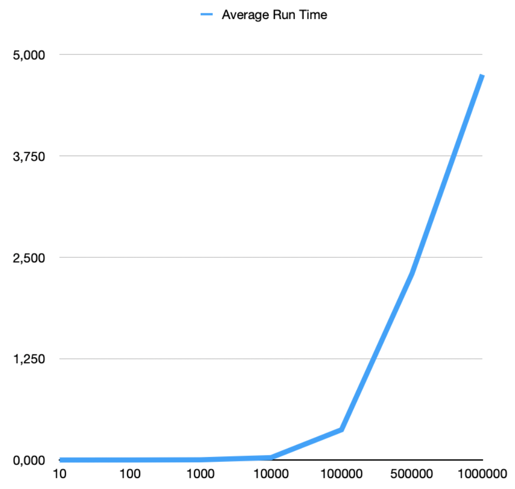


Figure 6: Average run time graph

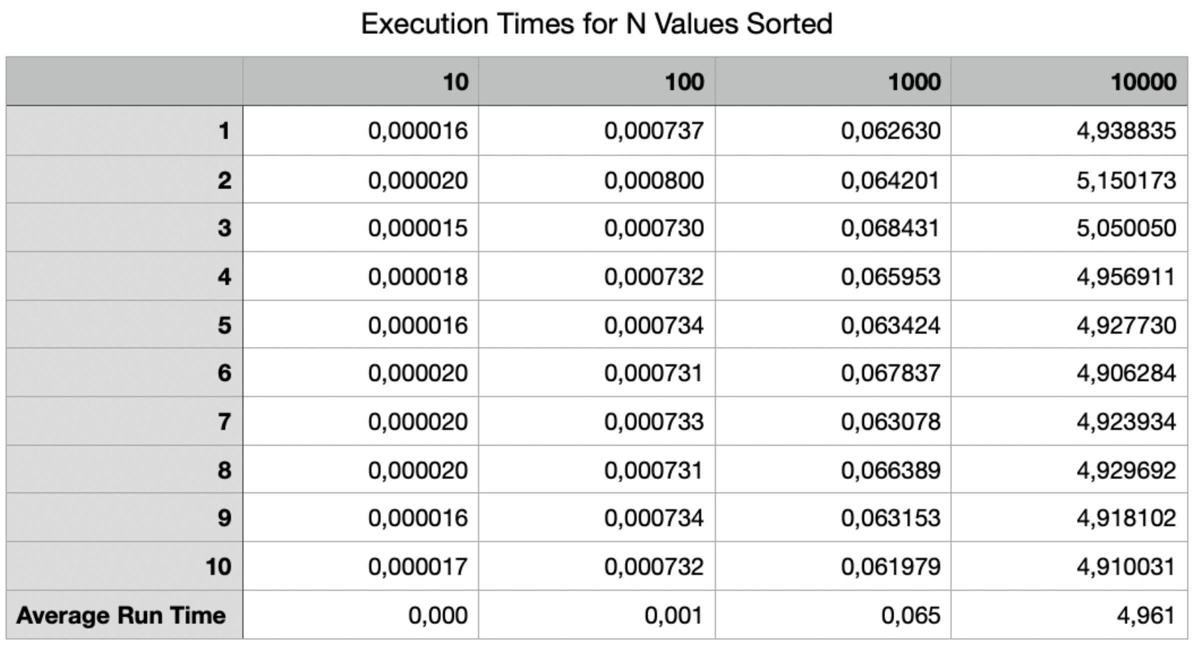
worst case notation and best, average case notation. Worst case situation(as a quadratic function) is more costly than the superlinear(best and average case) functions. For the larger input sizes the difference is even more.The run time when N=10K in the sorted array is similar to the result when I tried N=1M in the not sorted array. I was not even able to calculate the result for when N=100K. As a result it can be obtained that quicksort algorithm performs poorly for the sorted case.

4.2 Which other input cases would give us the similar results?

Reverse sorted,all values the same cases also give the similar results. Any case where one side of the partition is empty.

4.3 Propose a solution to this case.

At every partition if we choose a random pivot we would discard the case which we always pick the smallest or the greatest element of the array, if we randomize the scenario of having the worst case becomes extremely small by chance.



Execution Times for N Values Sorted

	10	100	1000	10000
1	0,000016	0,000737	0,062630	4,938835
2	0,000020	0,000800	0,064201	5,150173
3	0,000015	0,000730	0,068431	5,050050
4	0,000018	0,000732	0,065953	4,956911
5	0,000016	0,000734	0,063424	4,927730
6	0,000020	0,000731	0,067837	4,906284
7	0,000020	0,000733	0,063078	4,923934
8	0,000020	0,000731	0,066389	4,929692
9	0,000016	0,000734	0,063153	4,918102
10	0,000017	0,000732	0,061979	4,910031
Average Run Time	0,000	0,001	0,065	4,961

Figure 7: Execution times for N values sorted list

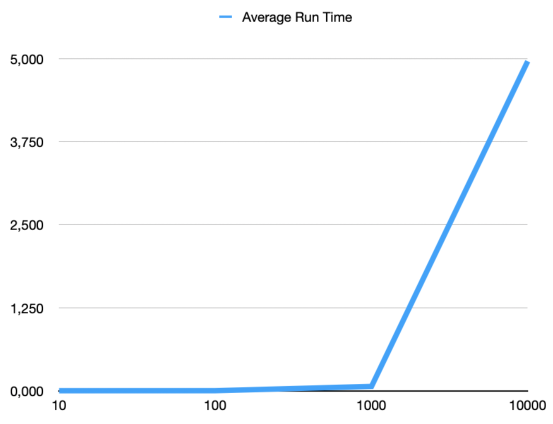


Figure 8: Average run time graph sorted list