Divide and Conquer VS Standard Algorithms

Student’s Name

Institutional Affiliation

Divide and Conquer VS Standard Algorithms

For the case of this assignment, we select selection sort and insertion sort as 2 divide and conquer algorithms. These algorithms have worst case running times of 0 (n2). If the size of the array input is huge, these algorithms may take quite a long time to execute (Furat, 2016). When comparing these sorting algorithms with two other standard sorting algorithms such a merge sort and quick sort, their running times are much better (Khan, n.d). To be specific, merge sorting algorithms executes in 0 (n log n) Θ(nlgn)\Theta, left parenthesis, n \log n, right parenthesis time in all situations, and quick sort runs in 0 (n log n) \Theta(n \log n) 0(n log n)\Theta, left parenthesis, n \log n, in the right parenthesis time in the best case and on average, although its worst-case execution time is 0 (n 2) \Theta(n2) 0 (n 2)\Theta, left parenthesis n squared right parenthesis (Chow, Chen & Tse, 2013). The table below provides a comparison of the two divide and conquer algorithms in comparison with the two standard algorithms.

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithms | Worst Case Running Time | Best Case Running Time | Average Case Running Time |
| Selection Sort | 0 (n2) | 0 (n2) | 0 (n2) |
| Insertion Sort | 0 (n2) | 0 (n) | 0 (n2) |
| Merge Sorting | 0 (n log n) | 0 (n log n) | 0 (n log n) |
| Quick Soring | 0 (n2) | 0 (n log n) | 0 (n log n) |

Table 1: Divide and Conquer and Standard Sorting Algorithms,

Both selection and insertion sorting algorithms, which falls under the category of ivied and conquer algorithm, solves the subproblems recursively as compared to merge or quick sort algorithms. In terms of time complexities, insertion sorting algorithm has the 0 (n) running time which is in the best case. Similarly, selection sorting algorithm which is also in the best case run time complexity has 0 (n2). On the other side, standard algorithms such as merge and quick sorting algorithms are in the worst-case running time even though both of them do compete respectively.

Divide and conquer technique reduces the overall time complexity since this approach is used for optimal solution of a problem. In particular, the algorithm can be converted into simple loops which are referred to as ***decrease and conquer.*** The rational behind having divide an conquer approach to reduce the overall time complexity is because it has a mechanism of dividing difficult problem into sub problems such as the classic Tower of Hanoi puzzle, that reduces moving a tower of height *n* to moving a tower of height *n* − 1. Additionally, this paradigm helps in the discovery of efficient algorithms, thus reducing the overall time complexity (Chow, Chen & Tse, 2013). A good example of the application of divide and conquer is the optimization, whereby if a search space is reduced by a fixed factor at each step, the complexity of the overall algorithm will reduce with a constant factor. Now that divide and conquer algorithm ultimately reduces each problem instance to a huge number of base instances, these often has an implication to the overall cost of the algorithm, particularly when joining or splitting overhead is small. Notable, such considerations are not dependent on whether the recursion is implemented by the compiler or by specific stack. Divide and conquer algorithms reduces time complexity to 0 (n log (n)). The whole process takes recursive approach to formulate the solution of the problem at hand.

References

Chow, C., Chen, T. Y., & Tse, T. H. (2013, July). The art of divide and conquer: An innovative approach to improving the efficiency of adaptive random testing. In *2013 13th International Conference on Quality Software* (pp. 268-275). IEEE.

Furat, F. G. (2016). A Comparative Study of Selection Sort and Insertion Sort Algorithms. *International Research Journal of Engineering and Technology (IRJET). e-ISSN*, 2395-0056.

Khan, A. (n.d). Divide and conquer algorithms. Retrieved from https://www.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/divide-and-conquer-algorithms