**DSA Assignment 1**

1. Explain the difference between a static array and a dynamic array. What are the pros and cons of each?
2. Describe how arrays are stored in memory and how their elements are accessed.
3. How can you implement an efficient algorithm to find the majority element in an array?
4. Discuss the concept of multi-dimensional arrays. How do you calculate the memory address of an element in a 2D array?
5. What are some common pitfalls when working with arrays, and how can they be avoided?
6. Explain the concept of array slicing. Provide scenarios where slicing can be useful.
7. How does array resizing work in a dynamic array? What is the amortized time complexity of appending an element to a dynamic array?
8. Discuss the trade-offs between using an array and a linked list for storing a sequence of elements.
9. Explain the difference between row-major and column-major order in a multi-dimensional array.
10. How does an array differ from other data structures like lists, stacks, and queues?
11. Explain how strings are represented in memory in different programming languages.
12. What are the various methods of comparing strings?
13. Discuss the concept of string interning. How does it optimize memory usage?
14. What is the difference between string concatenation and string interpolation? Which is more efficient and why?
15. How can you efficiently search for a substring within a string? Compare different algorithms used for this purpose.
16. Explain the process of dynamic memory allocation and deallocation. How does it differ from static memory allocation?
17. What are memory leaks, and how can they be prevented in programs that use dynamic memory allocation?
18. Discuss the role of memory management functions like malloc, calloc, realloc, and free in C.
19. How does a memory allocator work? Describe different memory allocation strategies like first-fit, best-fit, and worst-fit.
20. Describe the divide and conquer approach used in Merge Sort. How does it ensure that the array is sorted?
21. How does Quick Sort achieve its efficiency? Discuss the role of the pivot element.
22. What are the best-case, average-case, and worst-case time complexities of Quick Sort? When does the worst-case occur?
23. Explain the working of Bucket Sort. What types of data are most suitable for this sorting algorithm?
24. How does Radix Sort differ from other comparison-based sorting algorithms?
25. Why is Merge Sort preferred over Quick Sort in some scenarios, despite its higher space complexity?
26. Discuss the trade-offs between using an in-place sorting algorithm and one that requires additional memory.
27. Compare Linear Search and Binary Search in terms of efficiency. In what scenarios would you prefer one over the other?
28. Explain the conditions under which Binary Search can be applied. What are the consequences of applying it to an unsorted array?
29. How does Binary Search handle duplicate elements in a sorted array? Explain the process of finding the first and last occurrence of a target element.
30. How can Binary Search be applied to problems other than searching, such as finding the square root of a number?
31. Discuss the limitations of Binary Search in the context of large-scale data. How do modern algorithms overcome these limitations?
32. What are some real-world applications of searching algorithms? Discuss the importance of choosing the right algorithm for a given problem.