Data Structures TABA

Conor Kelly

21208140

Q.1 A stack is a data structure that follows the Last In, First Out (LIFO) principle, meaning that the most recently added element is the first one to be removed. Here are three real-world examples where you might implement a stack:

* Undo Function in Software Applications: Most software applications (such as text editors, image editors, or even spreadsheets) offer an undo function that allows users to reverse their most recent actions. This function can be implemented using a stack, where each user action is pushed onto the stack, and when the user wants to undo an action, the most recent one is popped off the stack.
* Web Browser History: When you browse the web, your browser keeps track of the pages you visit. This can be managed using a stack, where each new page you visit is pushed onto the stack. When you want to go back to the previous page, the browser can pop the top element (most recent page) from the stack to navigate back.
* Call Stack in Programming Languages: In programming languages, especially those that use recursion, a call stack is used to keep track of function calls and their execution state. When a function is called, its details are pushed onto the call stack, and when the function completes execution, its details are popped off the stack. This ensures proper sequencing of function calls and returns.

**Sequential Search:**

**Description:**

* In a sequential search (also known as linear search), the algorithm starts at the beginning of the collection and examines each element sequentially until it finds the target element or reaches the end of the collection.

**Advantages:**

* **Simple implementation:** It is easy to implement and understand.
* **Works with unsorted data:** Sequential search can be applied to both sorted and unsorted data.
* **Less overhead for small collections:** For small collections, the performance impact may be negligible.

**Disadvantages:**

* **Slow for large data sets:** In the worst case, it can take a long time to find an element, especially for large data sets, as it may need to examine every element.
* **No information about data structure:** The search does not take advantage of any particular organization or structure of the data.

**Binary Search:**

**Description:**

* Binary search is a more efficient search algorithm that works on sorted data. It begins by comparing the target element to the middle element of the collection. Depending on whether the target is greater or lesser, the search continues in either the left or right half of the collection, successively narrowing down the search range.

**Advantages:**

* **Much faster than sequential search:** The time complexity is O(log n), making it much quicker than sequential search, especially for large data sets.
* **Efficient use of data structure:** Binary search takes advantage of the sorted nature of the data to narrow down the search range.

**Disadvantages:**

* **Requires sorted data:** Binary search can only be applied to collections that are already sorted, which may require a preliminary sorting step.
* **More complex implementation:** Compared to sequential search, binary search can be slightly more complex to implement and understand.

**Conclusion:**

Sequential search is simpler and can handle unsorted data, but it can be slow for large collections. Binary search, on the other hand, is much faster for large collections but requires the data to be sorted beforehand. The choice of search algorithm depends on the data you are working with and the requirements of the application.