**What is Stack Data Structure?**

|  |
| --- |
| Stack is a **linear data structure** based on **LIFO(Last In First Out)** principle in which the insertion of a new element and removal of an existing element takes place at the same end represented as the **top** of the stack.  To implement the stack, it is required to maintain the **pointer to the top of the stack** , which is the last element to be inserted because **we can access the elements only on the top of the stack.**  **LIFO(Last In First Out) Principle in Stack Data Structure:**  This strategy states that the element that is **inserted last will come out first**. You can take a pile of plates kept on top of each other as a real-life **example.**  The plate which we put last is on the top and since we remove the plate that is at the top, we can say that the plate that **was put last comes out first.** |

Representation of Stack Data Structure:

|  |
| --- |
|  |

**Types of Stack Data Structure:**

|  |
| --- |
| **Fixed Size Stack :** As the name suggests, a fixed size stack has a fixed size and cannot grow or shrink dynamically. If the stack is full and an attempt is made to add an element to it, an overflow error occurs. If the stack is empty and an attempt is made to remove an element from it, an underflow error occurs. |
| **Dynamic Size Stack :** A dynamic size stack can grow or shrink dynamically. When the stack is full, it automatically increases its size to accommodate the new element, and when the stack is empty, it decreases its size. This type of stack is implemented using a linked list, as it allows for easy resizing of the stack. |

**Basic Operations on Stack Data Structure:**

|  |
| --- |
| there are certain operations provided to us.  **push()** to insert an element into the stack  **pop()** to remove an element from the stack  **top()** Returns the top element of the stack.  **isEmpty()** returns true if stack is empty else false.  **isFull()** returns true if the stack is full else false. |
| **Push Operation in Stack Data Structure:**  Adds an item to the stack. If the stack is full, then it is said to be an Overflow condition. |
| **Algorithm for Push Operation:**  Before pushing the element to the stack, we check if the stack is full .  If the stack is full (top == capacity-1) , then Stack Overflows and we cannot insert the element to the stack.  Otherwise, we increment the value of top by 1 (**top = top + 1**) and the new value is inserted at top position .  The elements can be pushed into the stack till we reach the capacity of the stack. |

|  |
| --- |
|  |

|  |
| --- |
| **Pop Operation in Stack Data Structure:**  Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an **Underflow condition.** |
| **Algorithm for Pop Operation:**  Before popping the element from the stack, we check if the stack is empty .  If the stack is empty **(top == -1),** then Stack Underflows and we cannot remove any element from the stack.  Otherwise, we store the value at top, decrement the value of top by 1 (**top = top – 1**) and return the stored top value. |

|  |
| --- |
|  |

|  |
| --- |
| **Top or Peek Operation in Stack Data Structure:**  Returns the top element of the stack. |
| **Algorithm for Top Operation:**  Before returning the top element from the stack, we check if the stack is empty.  If the stack is empty **(top == -1),** we simply print “**Stack is empty**”.  Otherwise, we return the element stored **at index = top** . |

|  |
| --- |
|  |

|  |
| --- |
| **isEmpty Operation in Stack Data Structure:**  Returns true if the stack is empty, else false. |
| **Algorithm for isEmpty Operation:**  Check for the value of top in stack.  If **(top == -1)** , then the stack is empty so return true .  Otherwise, the stack is **not empty so return false** . |

|  |
| --- |
|  |

|  |
| --- |
| **isFull Operation in Stack Data Structure:**  Returns true if the stack is full, else false. |
| **Algorithm for isFull Operation:**  Check for the value of top in stack.  If (top == capacity-1), then the stack is full so return true.  Otherwise, the stack is not full so return false. |

|  |
| --- |
| **Advantages of Stack Data Structure:**  **Simplicity:** Stacks are a simple and easy-to-understand data structure, making them suitable for a wide range of applications.  **Efficiency:** Push and pop operations on a stack can be performed in constant time (O(1)) , providing efficient access to data.  **Last-in, First-out (LIFO):** Stacks follow the LIFO principle, ensuring that the last element added to the stack is the first one removed. This behavior is useful in many scenarios, such as function calls and expression evaluation.  **Limited memory usage:** Stacks only need to store the elements that have been pushed onto them, making them memory-efficient compared to other data structures.  **Disadvantages of Stack Data Structure:**  **Limited access:** Elements in a stack can only be accessed from the top, making it difficult to retrieve or modify elements in the middle of the stack.  **Potential for overflow:** If more elements are pushed onto a stack than it can hold, an overflow error will occur, resulting in a loss of data.  **Not suitable for random access:** Stack s do not allow for random access to elements, making them unsuitable for applications where elements need to be accessed in a specific order.  **Limited capacity:** Stacks have a fixed capacity, which can be a limitation if the number of elements that need to be stored is unknown or highly variable. |
| **Applications of Stack Data Structure:**   * Infix to Postfix /Prefix conversion * Redo-undo features at many places like editors, photoshop. * Forward and backward features in web browsers * In Memory management, any modern computer uses a stack as the primary management for a running purpose. Each program that is running in a computer system has its own memory allocations. * Stack also helps in implementing function call in computers. The last called function is always completed first. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Easy Interview Questions on Stack Data Structure** | | [Parenthesis Checker](https://www.geeksforgeeks.org/check-for-balanced-parentheses-in-an-expression/) | | [Reverse a String using Stack](https://www.geeksforgeeks.org/stack-set-3-reverse-string-using-stack/) | | [Why do we need Prefix and Postfix notations?](https://www.geeksforgeeks.org/postfix-prefix-conversion/) | | [Reverse an array using Stack](https://www.geeksforgeeks.org/reverse-an-array-using-stack/) | | [How would I modify my code to delete chars from the beginning?](https://www.geeksforgeeks.org/remove-characters-from-the-first-string-which-are-present-in-the-second-string/) | | [Check for balanced parentheses in an expression](https://www.geeksforgeeks.org/check-for-balanced-parentheses-in-an-expression/) | | [Reverse a Stack using queue](https://www.geeksforgeeks.org/reverse-a-stack-using-queue/) | | [Delete Middle element from stack](https://www.geeksforgeeks.org/delete-middle-element-stack/) | | [Reverse individual words](https://www.geeksforgeeks.org/reverse-individual-words/) | |  |  |  | | --- | | Medium Interview Questions on Stack Data Structure | | [How to implement a queue using two stacks?](https://www.geeksforgeeks.org/queue-using-stacks/#:~:text=Method%201%20(By%20making%20enQueue,everything%20from%20stack1%20to%20stack2.) | | [Implement Stack using Queues](https://www.geeksforgeeks.org/implement-stack-using-queue/) | | [Implement a stack using a single queue](https://www.geeksforgeeks.org/implement-a-stack-using-single-queue/) | | [Evaluate Postfix Expression](https://www.geeksforgeeks.org/stack-set-4-evaluation-postfix-expression/) | | [Next Greater Element](https://www.geeksforgeeks.org/next-greater-element/) | | [Nearest Smaller Element](https://www.geeksforgeeks.org/find-the-nearest-smaller-numbers-on-left-side-in-an-array/) | | [Find next Smaller of next Greater in an array](https://www.geeksforgeeks.org/find-next-smaller-next-greater-array/) | | [Sort a stack using a temporary stack.](https://www.geeksforgeeks.org/sort-stack-using-temporary-stack/) | | [Stock Span Problem](https://www.geeksforgeeks.org/the-stock-span-problem/) | | [Reverse a stack using recursion](https://www.geeksforgeeks.org/reverse-a-stack-using-recursion/) | | [Infix to Postfix Conversion using Stack](https://www.geeksforgeeks.org/stack-set-2-infix-to-postfix/) | | [Implement two stacks in an array](https://www.geeksforgeeks.org/implement-two-stacks-in-an-array/) | | [Delete consecutive same words in a sequence](https://www.geeksforgeeks.org/delete-consecutive-words-sequence/) | | [Lexicographically largest subsequence containing all distinct characters only once](https://www.geeksforgeeks.org/lexicographically-largest-subsequence-containing-all-distinct-characters-only-once/) | | [Iterative approach to check if a Binary Tree is BST or not](https://www.geeksforgeeks.org/iterative-approach-to-check-if-a-binary-tree-is-bst-or-not/) | | [Minimize length by removing subsequences forming valid parenthesis from a given string](https://www.geeksforgeeks.org/minimize-length-by-removing-subsequences-forming-valid-parenthesis-from-a-given-string/) | | [Special Stack](https://www.geeksforgeeks.org/design-and-implement-special-stack-data-structure/) | | [Clone a stack without using extra space | Set 2](https://www.geeksforgeeks.org/clone-a-stack-without-using-extra-space-set-2/) | | [Count of Subarrays whose first element is the minimum](https://www.geeksforgeeks.org/count-of-subarrays-whose-first-element-is-the-minimum/) | | [Length of the longest valid substring](https://www.geeksforgeeks.org/length-of-the-longest-valid-substring/) | | [Find index of closing bracket for a given opening bracket in an expression](https://www.geeksforgeeks.org/find-index-closing-bracket-given-opening-bracket-expression/) | | [Next Greater Frequency Element](https://www.geeksforgeeks.org/next-greater-frequency-element/) | | [Find maximum difference between nearest left and right smaller elements](https://www.geeksforgeeks.org/find-maximum-difference-between-nearest-left-and-right-smaller-elements/) | | [Maximum product of indexes of next greater on left and right](https://www.geeksforgeeks.org/maximum-product-of-indexes-of-next-greater-on-left-and-right/) |  |  | | --- | | Hard Interview Questions on Stack Data Structure | | [Check if the given permutation is valid stack permutation or not](https://www.geeksforgeeks.org/stack-permutations-check-if-an-array-is-stack-permutation-of-other/) | | [Design a stack that supports getMin() in O(1) time and O(1) extra space](https://www.geeksforgeeks.org/design-a-stack-that-supports-getmin-in-o1-time-and-o1-extra-space/) | | [Design a data structure that supports insert(), delete(), getRandom() and getMin() in O(1) time complexity](https://www.geeksforgeeks.org/design-a-data-structure-that-supports-insert-delete-getrandom-in-o1-with-duplicates/) | | [Check if two expressions with brackets are same](https://www.geeksforgeeks.org/check-two-expressions-brackets/) | | [How to efficiently implement k stacks in a single array?](https://www.geeksforgeeks.org/efficiently-implement-k-stacks-single-array/) | | [Find the largest rectangular area possible in a given histogram](https://www.geeksforgeeks.org/largest-rectangle-under-histogram/) | | [Implementation of stack using Doubly Linked List](https://www.geeksforgeeks.org/implementation-of-stack-using-doubly-linked-list/) | | [The Celebrity Problem](https://www.geeksforgeeks.org/the-celebrity-problem/) | | [Design custom Browser History based on given operations](https://www.geeksforgeeks.org/design-custom-browser-history-based-on-given-operations/) | | [Maximum size rectangle binary sub-matrix with all 1s | Set 2](https://www.geeksforgeeks.org/maximum-size-rectangle-binary-sub-matrix-with-all-1s-set-2/) | | [Sort a stack using another stack](https://www.geeksforgeeks.org/sort-a-stack-using-recursion/) | | [Implement a stack that supports findMiddle() and deleteMiddle() operations in O(1) time complexity](https://www.geeksforgeeks.org/design-a-stack-with-find-middle-operation/) | | [Maximum people a person can see while standing in a line in both direction](https://www.geeksforgeeks.org/maximum-people-a-person-can-see-while-standing-in-a-line-in-both-direction/) | | [Count of distinct differences between two maximum elements of every Subarray](https://www.geeksforgeeks.org/count-of-distinct-differences-between-two-maximum-elements-of-every-subarray/) | | [Largest Rectangular Area in a Histogram](https://www.geeksforgeeks.org/largest-rectangle-under-histogram/) | | [Range Queries for Longest Correct Bracket Subsequence Set](https://www.geeksforgeeks.org/range-queries-longest-correct-bracket-subsequence-set-2/) | | [Find maximum of minimum for every window size in a given array](https://www.geeksforgeeks.org/find-the-maximum-of-minimums-for-every-window-size-in-a-given-array/) | | [Print ancestors of a given binary tree node without recursion](https://www.geeksforgeeks.org/print-ancestors-of-a-given-binary-tree-node-without-recursion/) | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Practice problems on Stack for Competitive Programming**   |  |  | | --- | --- | | Problem | Practice link | | [Largest Rectangular Area in a Histogram using Stack](https://www.geeksforgeeks.org/largest-rectangular-area-in-a-histogram-using-stack/) | [Solve](https://www.geeksforgeeks.org/problems/maximum-rectangular-area-in-a-histogram-1587115620/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=bottom_sticky_on_article) | | [Merge Overlapping Intervals](https://www.geeksforgeeks.org/merging-intervals/) | [Solve](https://practice.geeksforgeeks.org/problems/overlapping-intervals--170633/1) | | [Design a stack that supports getMin() in O(1) time and O(1) extra space](https://www.geeksforgeeks.org/design-a-stack-that-supports-getmin-in-o1-time-and-o1-extra-space/) | [Solve](https://www.geeksforgeeks.org/problems/get-minimum-element-from-stack/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=bottom_sticky_on_article) | | [Maximum size rectangle binary sub-matrix with all 1s](https://www.geeksforgeeks.org/maximum-size-rectangle-binary-sub-matrix-1s/) | [Solve](https://practice.geeksforgeeks.org/problems/max-rectangle/1?utm_source=gfg&utm_medium=article&utm_campaign=bottom_sticky_on_article) | | [The Stock Span Problem](https://www.geeksforgeeks.org/the-stock-span-problem/) | Solve | | [The Celebrity Problem](https://www.geeksforgeeks.org/the-celebrity-problem/) | [Solve](https://practice.geeksforgeeks.org/problems/the-celebrity-problem/1?utm_source=gfg&utm_medium=article&utm_campaign=bottom_sticky_on_article) | | [ZigZag Tree Traversal](https://www.geeksforgeeks.org/zigzag-tree-traversal/) | [Solve](https://www.geeksforgeeks.org/problems/zigzag-tree-traversal/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=bottom_sticky_on_article) | | [Length of the longest valid substring](https://www.geeksforgeeks.org/length-of-the-longest-valid-substring/) | [Solve](https://www.geeksforgeeks.org/problems/longest-valid-parentheses5657/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=bottom_sticky_on_article) | | [Reduce the string by removing K consecutive identical characters](https://www.geeksforgeeks.org/reduce-the-string-by-removing-k-consecutive-identical-characters/) | [Solve](https://www.geeksforgeeks.org/problems/restrictive-candy-crush--141631/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=bottom_sticky_on_article) | | [Minimum number of bracket reversals needed to make an expression balanced](https://www.geeksforgeeks.org/minimum-number-of-bracket-reversals-needed-to-make-an-expression-balanced/) | [Solve](https://www.geeksforgeeks.org/problems/count-the-reversals0401/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=bottom_sticky_on_article) | |