# LintCode Check List

## Chapter 1 -  strStr & Coding Style

### Required:

* 13. strStr     *(E)*

### Optional:

* 17. Subset     *(M)*
* 18. Subsets II     *(M)*
* 594. strStr II     *(H, L)*

### Related:

* 15. Permutations     *(M)*
* 16. Permutations II  *(M)*

## Chapter 2 - Binary Search

### Required

* 459. Closest Number in Sorted Array     *(E, L)*
* 458. Last Position of Target     *(E, L)*
* 28. Search a 2D Matrix     *(E)*
* 585. Maximum Number in Mountain Sequence  *(M, L)*
* 447. Search in a Big Sorted Array     *(M, L)*
* 159. Find Minimum in Rotated Sorted Array     *(M)*
* 75. Find Peak Element     *(M)*
* 74 First Bad Version     *(M)*
* 62. Search in Rotated Sorted Array     *(M)*

### Optional

* 462. Total Occurrence of Target     *(E, L)*
* 254. Drop Eggs     *(E, L)*
* 14. First Position of Target     *(E)*
* 460. K Closest Numbers in Sorted Array     *(M, L)*
* 414. Divide Two Integers     *(M)*
* 61. Search for a Range     *(M)*
* 38. Search a 2D Matrix II     *(M)*
* 600. Smallest Rectangle Enclosing Black Pixels     *(H, L)*

### Related

* 457. Classical Binary Search     *(E)*
* 141. Sqrt(x)     *(E)*
* 617. Maximum Average Subarray     *(M)*
* 586. Sqrt(x) II     *(M, L)*
* 160. Find Minimum in Rotated Sorted Array II     *(M)*
* 63. Search in Rotated Array II     *(M)*
* 437. Copy Books     *(H)*
* 183. Wood Cut     *(H)*
* 39. Recover Rotated Sorted Array *(E)*
* 8. Rotate String *(E)*

## Chapter 3 - Binary Tree & Divide Conquer

### Required

* 597. Subtree with Maximum Average     *(E, L)*
* 596. Minimum Subtree     *(E, L)*
* 480. Binary Tree Paths     *(E)*
* 453. Flatten Binary Tree to Linked List     *(E)*
* 97. Maximum Depth of Binary Tree     *(E)*
* 93. Balanced Binary Tree     *(E)*
* 67. Binary Tree Inorder Traversal     *(E)*
* 66. Binary Tree Preorder Traversal     *(E)*
* 578. Lowest Common Ancestor III     *(M, L)*
* 95. Validate Binary Search Tree     *(M)*

### Optional

* 595. Binary Tree Longest Consecutive Sequence     *(E)*
* 376. Binary Tree Path Sum     *(E)*
* 474. Lowest Common Ancestor II     *(E, L)*
* 246. Binary Tree Path Sum II     *(E, L)*
* 68. Binary Tree Postorder Traversal     *(E)*
* 619. Binary Tree Longest Consecutive Sequence III     *(M, L)*
* 614. Binary Tree Longest Consecutive Sequence II     *(M)*
* 475. Binary Tree Maximum Path Sum II     *(M, L)*
* 448. Inorder Successor in Binary Search Tree     *(M, L)*
* 378. Convert Binary Search Tree to Doubly Linked List     *(M)*
* 472 Binary Tree Path Sum III     *(H, L)*

### Related

* 155. Minimum Depth of Binary Tree     *(E)*
* 88. Lowest Common Ancestor     *(M)*
* 175. Invert Binary Tree (*E*)
* 86. Binary Search Tree Iterator (H)
* 11. Search Range in Binary Search Tree (M)
* 85. Insert Node in a Binary Search Tree (E)
* 87. Remove Node in Binary Search Tree (H)

## Chapter 4 - Breadth First Search

### Required

* 433. Number of Islands   *(E)*
* 69. Binary Tree Level Order Traversal     *(E)*
* 616. Course Schedule II     *(M)*
* 618. Search Graph Nodes     *(M, L)*
* 611. Knight Shortest Path     *(M, L)*
* 598. Zombie in Matrix     *(M, L)*
* 178. Graph Valid Tree     *(M)*
* 137. Clone Graph     *(M)*
* 7. Binary Tree Serialization     *(M)*
* 573. Build Post Office II     *(H, L)*

### Optional

* 242. Convert Binary Tree to Linked Lists by Depth     *(E, L)*
* 624. Remove Substrings     *(M, L)*
* 605. Sequence Reconstruction     *(M, L)*
* 531. Six Degrees     *(M, L)*
* 127. Topological Sorting     *(M)*
* 120. Word Ladder     *(M)*

### Related

* 615. Course Schedule     *(M)*
* 431. Connected Component in Undirected Graph     *(M, L)*
* 71. Binary Tree Zigzag Level Order Traversal II     *(M)*
* 70. Binary Tree Level Order Traversal II     *(M)*
* 600. Smallest Rectangle Enclosing Black Pixels     *(H, L)*
* 574. Build Post Office     *(H, L)*
* 434. Number of Island II     *(H, L)*

## Chapter 5 - Depth First Search

### Required

* 136. Palindrome Partitioning     *(M)*
* 153. Combination Sum II     *(M)*
* 135. Combination Sum     *(M)*
* 18. Subsets II     *(M)*
* 16. Permutations II     *(M)*
* 15. Permutations     *(M)*
* 121 Word Ladder II     *(H)*

### Optional

* 120 Word Ladder     *(M)*
* 52. Next Permutation     *(M)*
* 51. Previous Permutation     *(M)*
* 582. Word Break II     *(H)*

### Related

* 211. String Permutation     *(E, L)*
* 197. Permutation Index     *(E)*
* 10. String Permutation II     *(M, L)*
* 190. Next Permutation II     *(M)*
* 198. Permutation Index II     *(M)*
* 107. Word Break     *(M)*
* 108. Palindrome Partitioning II     *(M)*

## Chapter 6 - Linked List & Array

### required

* 599. Insert into a Cyclic Sorted List     *(E, L)*
* 165. Merge Two Sorted Lists     *(E)*
* 138. Subarray Sum     *(E)*
* 41. Maximum Subarray    *(E)*
* 139. Subarray Sum Closest     *(M)*
* 105. Copy List with Random Pointer    *(M)*
* 102. Linked List Cycle    *(M)*
* 98. Sort List     *(M)*
* 450. Reverse Node in k-Group    *(H)*
* 65. Median of two Sorted Arrays     *(H)*

### Optional

* 548. Intersection of Two Array II     *(E)*
* 547 Intersection of Two Arrays    *(E)*
* 96. Partition List     *(E)*
* 64. Merge Sorted Array    *(E)*
* 6. Merge Two Sorted Arrays     *(E)*
* 103. Linked List Cycle II     *(H)*

### Related

* 41. Maximum Subarray    *(E)*
* 620. Maximum Subarray IV    *(M, L)*
* 617. Maximum Average Subarray    *(M)*
* 191. Maximum Product Subarray    *(M)*
* 45. Maximum Subarray Difference     *(M)*
* 42. Maximum Subarray II    *(M)*
* 621. Maximum Subarray V    *(H, L)*
* 43. Maximum Subarray III      *(H)*

## Chapter 7 - Two Pointers

### Required

* 607. Two Sum - Data structure design     *(E, L)*
* 521. Remove Duplicate Numbers in Array   *(E, L)*
* 608. Two Sum - Input array is sorted     *(M)*
* 609. Two Sum - Less than or equal to target     *(M, L)*
* 587. Two Sum - Unique pairs     *(M, L)*
* 533. Two Sum - Closest to target     (M, L)
* 148. Sort Colors     *(M)*
* 143. Sort Colors II     *(M)*
* 57. 3Sum    *(M)*
* 31. Partition Array     *(M)*

### Optional

* 604. Window Sum   *(E)*
* 539. Move Zeroes   *(E)*
* 415. Valid Palindrome   *(E)*
* 56. Two Sum   *(E)*
* 625. Partition Array II   *(M, L)*
* 610. Two Sum - Difference equals to target     *(M, L)*
* 443. Two Sum - Greater than target     *(M, L)*
* 461. Kth Smallest Numbers in Unsorted Array    *(M, L)*
* 382. Triangle Count     *(M, L)*
* 59. 3Sum Closest     *(M)*
* 58. 4Sum     *(M)*

### Related

* None

## Chapter 8 - Hash & Heap

### Required

* 128. Hash Function     *(E)*
* 613. High Five     *(M, L)*
* 612. K Closest Points    *(M, L)*
* 606. Kth Largest Element II    *(M, L)*
* 544. Top k Largest Numbers    *(M, L)*
* 129. Rehashing  *(M)*
* 104. Merge k Sorted Lists    *(M)*
* 4. Ugly Number II    *(M)*
* 594. strStr II     *(H, L)*
* 134. LRU Cache  *(H)*

### Optional

* 601. Flatten 2D Vector    *(M, L)*
* 545. Top k Largest Numbers II     *(M, L)*
* 486. Merge k Sorted Arrays    *(M, L)*
* 471. Top K Frequent Words    *(M, L)*
* 130. Heapify     *(M)*
* 124. Longest Consecutive Sequence  *(M)*

### Related

* 551. Nested List Weight Sum  *(E, L)*
* 494. Implement Stack by Two Queues  *(E, L)*
* 575. Expression Expand  *(M, L)*
* 541. Zigzag Iterator II    *(M, L)*
* 540. Zigzag Iterator     *(M, L)*
* 528. Flatten Nested List Iterator     *(M)*
* 24. LFU Cache    *(H)*

## Chapter 9 - Dynamic Programming

### Required

* 115. Unique Paths II    *(E)*
* 114. Unique Paths    *(E)*
* 111. Climbing Stairs     *(E)*
* 110. Minimum Path Sum    *(E)*
* 109. Triangle    *(E)*
* 603. Largest Divisible Subset    *(M)*
* 611. Knight Short Shortest Path    *(M)*
* 513. Perfect Squares    *(M)*
* 116. Jump Game    *(M)*
* 76. Longest Increasing Subsequence    *(M)*

### Optional

* 272. Climbing Stairs II    *(E, L)*
* 630. Knight Shortest Path II     *(M, L)*
* 117. Jump Game II  *(M)*
* 602. Russian Doll Envelopes  *(H)*
* 622.Frog Jump    *(H)*

### Related

* 254. Drop Eggs    *(E, L)*
* 584. Drop Eggs II*(M, L)*