|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Type | Difficulty | Problem | Confidence | Solution |
| 1 | #️ | 🔴 | 128-Longest-Consecutive-Sequence | Low | Check for elements on left, increase if there exists an adjacent number to the right |
| 2 | #️ | 🟡 | 36-Valid-Sudoku | High | Hashmap each row, col and (row//3, col//3) for values in grid |
| 3 | # | 🟡 | 49-group-anagram | Medium | Iterate elementwise, sort the string, if it isn’t inside the hashmap, include it, otherwise, append the values that are equal to the sorted string |
| 4 | #️ | 🟡 | 238-Product-of-Array-Except-Self | Medium | Create prefix, postfix arrays.  To create prefix array:   * start iterator at i=1; then go prefix[i-1] \* nums[i-1]   To create postfix array:   * start iterator at len(nums)-2 (second last element); then go postfix[i+1] \* nums[i+1]   The interesting thing here is that you can use the value in nums and the updated version of postfix, prefix to get the elements you need. |
| 5 | # | 🟡 | 347-top-k-frequent | Medium | Create a 2D array depending on the length+1, count each element, add each element into the 2D array depending on frequency, iterate backwards, then iterate through each subarray, then append each value to the result array |
| 6 | # | 🟡 | 659-Encode-and-Decode-String | Low | Encode the string, with a prefix containing number and symbol before each word in the array.  Decode: iterate through the encoded string, while the element from the encoded string is != to symbol, j+=1, the decoded string will be the length from j+1 till j+1+length |
| 7 | #️ | 🟡 | secret-santa | Medium | Create two arrays, one for giver, one for receiver. While the length of the giver array is > 0, randomise the giver array, create a giver from giver\_array[0], if this is inside the receiver array, remove it from the receiver array, select a random person from receiver array, the add the giver back into the giver array. Append each value to a result array. Remove the giver and receiver each time. |
| 8 | # | 🟢 | 1-Two-Sum (return index) | Medium | Create quotient checker, where key=quotient, value=index. If the element in nums corresponds to a key in the array then return the value and index |
| 9 | #️ | 🟢 | 217-Contains-Duplicate-hashmap | High | Create checker hashmap, if value > 1 return False |
| 10 | #️ | 🟢 | 217-Contains-Duplicate-hashset | High | Create checker hashset, add each value inside the set if it’s not in there, otherwise, return False |
| 11 | #️ | 🟢 | 242-Valid-Anagrams | High | Create checker for each string, if the lengths are different return false, if the keys are different return false, if the values are different return False |
| 12 | #️ | 🟢 | Intersection-of–Two-Arrays-II | High | Create checker hashmap for array1 and array2. Iterate through the smaller array, if the values in the longer\_array are > 0 then append that value, also reduce the count of the checker array by -=1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 13 | ⏮ | 🟡 | 79-Word-Search | Low | Create depth first search for grid. |
| 14 | 🌲 | 🟢 | Trees-104-Maximum-Depth-of-Binary-Tree-BFS | Low |  |
| 15 | 🌲 | 🟢 | Trees-104-Maximum-Depth-of-Binary-Tree-DFS | Medium | Recursively call the function so that it adds +=1 each time it traverses the maximum value for the tree |
| 16 | 🌲 | 🟢 | Trees-226-Invert-Binary-Tree | Medium | Recursively call the function so that each right and left tree are swapped |
| 17 | 🎹 | 🟡 | 56-Merge-Intervals | Medium | Sort the array, create a result array with initial value from input array. If the start of the current iterable is >= the result[0][-1] then update result[0][-1] = max(end, result[0][-1]) , otherwise append the other values to the result array |
| 18 | 🎹 | 🟢 | 920-Meeting-Rooms | Medium | Sort the input array. i=1, While i < len(input\_array) we want to know if the beforeMeeting.end > subsequentMeeting.start, if it is return false.  **Note: this is helpful if you want to check between two array values without going out of bounds.** |
| 19 | 👆👇 | 🟡 | 1b-Two-Sum-II-input-Array-Is-Sorted | High | Create a sum = array[r] + array[l]. If it’s sorted then we can assume that if sum > target, r-=1, elif sum < target l +=1, otherwise return [r+1, l+1] |
| 20 | 👆👇 | 🟡 | 11-Container-With-Most-Water | Low | Volume is equal to the two pointers (r-l) \* min height of the value (array[l], array[r]). If array[l] < array[r] then l+=1 otherwise r-=1 |
| 21 | 👆👇 | 🟡⛔️ | 3Sum | Low | Iterate through each value, to avoid duplicates, skip if current value == previous value. Loop through each subarray, change pointers according to current\_sum, within the subarray, while the nums[left\_pointer] == nums[left\_pointer-1] push the left pointer forward by one. |
| 22 | 👆👇 | 🟢 | 125-valid-palindrome | High | Create substring, check if each value is an ascii letter or number. With two pointers, if check\_string[l] != check\_string[r] then return False |
| 23 | 👆👇 | 🟢 | 283-move-zeroes | Medium | Iterate through array, have a spare pointer zeroIndex, if an array[j] != 0 then swap with value of zeroIndex |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 24 | 📈📉 | 🟡 | 5-Longest-Palindromic-Substring | Low | Have a resultLength = 0, iterate through the string with a for loop, then begin with a while loop, account for even and odd strings by having r,l = i,i (odd), r,l = i,i+1 (even). Then update resultLength and resultString = s[l : r +1] |
| 25 | 📈📉 | 🟡 | 647-Palindromic-Substrings | Medium | Basically the same above, result += 1 for when there is a palindrome in a substring |
| 26 | 📈📉 | 🟡 | 91-Decode-Ways | Low |  |
| 27 | 📈📉 | 🟢 | 70-Climbing-Stairs | Medium | Fibonacci’s sequence |
| 28 | 🔟 | 🟢 | 704-binary-search | Medium | Start in the middle (r+l)//2 , if nums[middle] > target then m = r-1, if nums[middle] < target then m = l+1, each time |
| 29 | 🖇 | 🟢 | 21-Merge-Two-Sorted-List | Low |  |
| 30 | 🖇 | 🟢 | 141-Linked-List-Cycle | Low | Hair and tortoise algorithm |
| 31 | 🖇 | 🟢 | 206-Reverse-Linked-List-iterative | Medium |  |
|  |  |  |  |  |  |
| 32 | 🤑 | 🟡 | 55-Jump-Game | Low | Start at the end of the array, make your way down, if the nums[i] >= distance to the end=len(nums)-1-I, but the goal also changes, so goal = i at the end of the iteration |
| 33 | 🤑 | 🟢 | 53-Maximum-Subarray | Medium | Start at the beginning, if the total<0 then total=0, total += num, maxTotal = max(maxTotal, total) |
| 34 | 🥞 | 🟢 | 20-Valid-Parentheses | Low | Create hasmap with keys: },],) and corresponding values. Iterate through the string, if the stack is not empty and the hashmap[char] == stack[-1] then pop the element, otherwise append it to the list |
| 35 | 🥞 | 🟢 | 155-Min-Stack🤷‍♂️ | Low |  |
| 36 | 🧮 | 🟢 | 66-plus-one-list🤷‍♂️ | Low | Reverse the list, while keepIterating: if i<len(nums) then if the nums[i] == 9 then nums[i] == 0, once i>len(nums) (else) then append 1 to list. Return re-reversed list |
| 37 | 🧮 | 🟢 | 202-happy-number | Low | Reduce the number |
| 38 | 🪟 | 🔴 | 76-Minimum-Window-Substring | Low AF |  |
| 39 | 🪟 | 🟡 | 3-Longest-Substring-Without-Repeating-Characters | Low | Iterate through string, then while s[r] is in the set, remove that char from the set, increase left pointer by 1. After while loop, add s[l] to the set |
| 40 | 🪟 | 🟡 | 424-Longest-Repeating-Character-Replacement | Low AF | Iterate through string with right pointer, add s[r] to hashmap, if the distance between the pointers (r-l+1) – max(checker.values()) > k (where k is the amount of replacements possible), reduce the checker[s[l]]-=1, increase l+=1, the result is max(result, r-l+1) |
| 41 | 🪟 | 🟡 | 424-Longest-Repeating-Character-Replacement | Low AF | Create two arrays -> arr1, arr2: [0]\*26 and fill each one +=1 according to the letter of the char. Then iterate through the arrays, if arr1[i] == arr2[i] increment match +=1. Then sliding window: iterate with a for r in range(len(s1), len(s2)): index = ord(s[r])-ord(‘a’), countS2[index] += 1 if countS1[index] == countS2[index] then match +=1, if countS2[index]+1 == countS1[index] then match -=1. Do the reverse with the left pointer (de-increment by 1, then check if equal). |
| 42 | 🪟 | 🟢 | 121-Best-Time-To-Buy-and-Sell-Stock | High | Start with two pointers, l=0, r=1, while r < len(nums) if nums[r] < nums[l] then l=r, max\_sum = max(max\_sum, nums[r]-nums[l]) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 43 | 🟢 |  | 28-Implement-strStr() | Medium | Iterate through string (for i in range(len(s2)), if s2[i: i+len(s1)] then return i |
| 44 | 🟢 |  | longest\_common\_prefix | Medium | Iterate through first element with i (for i in range(len(strs[0])), then iterate elementwise in strs (for s in strs), if i==len(s) or s[i] != strs[0][i] return result, outside of that result += s[i] |

Move Zeroes