Array

Wednesday, 25 January 2023 7:10 AM

1. Array:
2. **Hard:**
3. https://www.geeksforgeeks.org/count-of-subarrays-having-exactly-k-distinct-elements/
   a. https://www.junhaow.com/lc/problems/array/sliding-window/992_subarrays-with-k-different-integers.html
4. https://www.geeksforgeeks.org/maximum-profit-by-buying-and-selling-a-share-at-most-twi
   a. https://www.techiedelight.com/find-maximum-profit-earned-from-at-most-two-stock-transactions/

```
publicclassSolution{wo
   publicintmaxProfit(int[] prices) {
      // Start typing your Java solution below
      // DO NOT write main() function
      if(prices== null|| prices.length== 0) return0;
      intmin= prices[0];
      intmax= Integer.MIN_VALUE;

      for(inti= 0; i< prices.length; i++){
         if(prices[i] < min) min= prices[i];
        intmoney= prices[i] - min;
        if(money> max) max= money;
      }
      returnmax;

   }
}
```

Input: prices = [7,1,5,3,6,4]
Output: 5

5. https://www.geeksforgeeks.org/median-of-two-sorted-arrays-of-different-sizes/
6. https://www.geeksforgeeks.org/median-of-two-sorted-arrays/

7. Medium:
   a. https://www.geeksforgeeks.org/number-subarrays-sum-exactly-equal-k/

b. https://www.geeksforgeeks.org/count-pairs-difference-equal-k/
c. https://www.geeksforgeeks.org/minimum-number-of-jumps-to-reach-end-of-a-given-
    i. https://www.geeksforgeeks.org/minimum-number-jumps-reach-endset-2on-sol
d. https://www.geeksforgeeks.org/find-maximum-product-of-a-triplet-in-array/
    i. https://afteracademy.com/blog/maximum-product-of-three-numbers/
    ii. Approaches
        i. Sort the data, fetch last 3 and first 2 to take max of product
        ii. Instead of sorting, identify last 3 and first 2 by o(n) traversal and perform operations with TC of O(N) using single traversal

- https://www.geeksforgeeks.org/given-an-array-a-and-a-number-x-check-for-pair-in-a-sum-as-x/
    - Sort the array, and then two pointer technique
    - Sort the array, and then iterate through the array, and perform binary search fo arr[i] in O(N logn)
    - Using hashing technique(without sorting) in O(n)
- https://www.geeksforgeeks.org/find-a-triplet-that-sum-to-a-given-value/
    - 1 iteration is required followed by the solution of previous question
- https://takeuforward.org/data-structure/3-sum-find-triplets-that-add-up-to-a-zero/
    - 1 iteration is required followed by two pointer technique to identify the best combination sum up to 0
- Book Allocation
    - https://www.geeksforgeeks.org/allocate-minimum-number-pages/
    - Youtube -> Allocate Minimum Number of Pages | Binary Search
- https://www.geeksforgeeks.org/kth-largest-element-in-a-stream/
- Minimum swaps to sort
    - https://www.geeksforgeeks.org/minimum-number-swaps-required-sort-array/
    - Sort the array using pair concept(value, original index)
    - Pick the elements iteratively and swap the element by moving it to its original in before sorting.
    - Keep doing until all will move to its original position and calculate swapCount.
- https://takeuforward.org/data-structure/find-the-majority-element-that-occurs-more n-2-times/
    - https://www.techiedelight.com/find-majority-element-in-an-array-boyer-moore majority-vote-algorithm/
    - Boyer-moore voting algorithm
    - Only 1 candidate is possible to have more conunt than total_length/2
    - Consider Candidate, count.
    - Reset count whenever the candidate count becomes zero and assign next eleme new candidate

- https://takeuforward.org/data-structure/majority-elementsn-3-times-find-the-elemen

-array/
lution/

max

-with-

r sum -

ndex

e-than-

e-

ent as

nts-that-

appears-more-than-n-3-times-in-the-array/

- Approach1
  - Brute force - take each elements and identify its count
  - TC would be O(n2)
  - Sort the array and calculate the count of each element using binary search O(nlogn)
- Approach2
  - Use hashing technique to store the frequency of each element
- Approach3
  - Only 2 elements are possible which can have frequency more than O(N)/2
    Use boyer-moore Voting algorithm to avoid the extra space of O(N) by line search
    So Take 2 counters in this case and do the same search for 2 bigger freque number here
    And validate whether the total/3 < count

- Rotten orange problem:
  - https://takeuforward.org/data-structure/rotten-oranges-min-time-to-rot-all-ora bfs/
  - BFS traverse approach
  - And check the queue size before processing the next set of neighbors
  - Once the set of neighbour processing is completed, increment the min count;
- https://takeuforward.org/data-structure/area-of-largest-rectangle-in-histogram/
  - It can be solved using next greater element variant(nge), i.e., left smaller elemer
    - Nge or lse can be solved using the stack datastructure.
    - Push the elements in linear order into the stack and remove it if it is great the current element in array
    - Pop it until the element in stack is smaller than the array element
    - That will be considered as last smaller element
  - Using one iteration
- https://takeuforward.org/data-structure/trapping-rainwater/
  - Maintain Prefix, suffix arrays values
  - Calculate lefMax, rightMax
- Count frequency of number in sorted array
  - https://www.geeksforgeeks.org/counting-frequencies-of-array-elements/
  - https://webrewrite.com/count-frequency-of-a-number-in-a-sorted-array/
  - https://www.geeksforgeeks.org/count-number-of-occurrences-or-frequency-in-sorted-array/

  - Using binary search logic to identify the boundaries of the repeated number and
  - Take a diff of it.

n. TC -

ear

ency

anges-

nt

er than

-a-

d

- Sliding window maximum subarray
  - https://www.geeksforgeeks.org/sliding-window-maximum-maximum-of-all-sub of-size-k/
  - Solve using max deque in decreasing order
  - removeLast(), removeFirst(), addLast(), peekLast(), peekFirst()
  - https://takeuforward.org/data-structure/sliding-window-maximum/
  - Approach:
    - Using deque, pop elements out if it is out of range from window size
    - Remove elements from back if it is smaller than current element
    - Push the element in to the stack
    - Take the peek element and consider it as maximum
- Cycle present in circular array or not -> https://thecodingsimplified.com/check-if-cycle present-in-circular-array/

Sliding window technique - https://www.geeksforgeeks.org/window-sliding-technique/

- https://www.geeksforgeeks.org/count-distinct-elements-in-every-window-of-size-k/
  - To identify distinct elements,
- https://www.geeksforgeeks.org/find-subarray-with-given-sum/
  - Sliding window technique since it is a contagious subarray to find
  - Subarray with given sum in O(N) complexity
    - Sliding window technique since it is contagious subarray
    - Using left & right pointers with current_sum
    - https://www.geeksforgeeks.org/find-subarray-with-given-sum/
    - https://www.codingninjas.com/codestudio/library/what-is-subarray-with sum

- Find The Longest Increasing Subsequence - DP
- Longest Common Subsequence
- Print Longest substring without repeating characters - sliding window technique
- Largest sum Contiguous Subarray - Kadane's algorithm
  - https://www.geeksforgeeks.org/largest-sum-contiguous-subarray/
- Kth Largest Sum Contagious SubArray - https://www.geeksforgeeks.org/k-th-largest-s contiguous-subarray/
  - Instead of sliding window, use prefix-sum to identify the largest sum.
  - Use 2 nested for loops to identify the different combinations of prefix sum
  - Insert it in min-heap with size <= K,
  - At the end fetch the top element, which will be Kth largest subarray.

- Longest Consecutive Subsequence

arrays-

e-in-

-given-

um-

- Longest Consecutive Subsequence
-