AlgoNinjas Contest 1 - IIT, Bombay
Date: 29th June 2024

Topics:

- Linked List
- Binary Search
- Standard Algorithms

Question 1: Standard Algorithms (Easy level)
[24706]

https://www.naukri.com/code360/problems/front-back_7574153

Question 2: Linked List (Easy level)
[20251]
https://www.naukri.com/code360/problems/remove-loop_4609669

Question 3: Standard Algorithm (Medium level)
[23295]
https://www.naukri.com/code360/problems/ninja-and-the-magic-piles_6680367

Question 4: Linked List (Medium Level)
[17603]
https://www.naukri.com/code360/problems/fold-and-merge-linked-list_3929252

Question 5: Binary Search (Hard Level)
[29893]
https://www.naukri.com/code360/problems/card-game_10929469

Question 6: Binary Search (Hard Level)
[30774]
https://www.naukri.com/code360/problems/sum-of-digits_11507656

SOLUTIONS:

ANSWER 1: [24706]

LANGUAGE: C++

```
/*
   Time Complexity: O(N)
   Space Complexity: O(N)

   where 'N' is the length of the array 'A'.
*/

int frontBack(vector<int> &a)
{
   int n = a.size();

   // Creating an unordered map 'mp' of key-value pair as [int, int].
   unordered_map<int, int> mp;

   // Initializing 'j' with 0 and 'ans' with 1e9.
   int j = 0, ans = 1e9;

   // Calculating answer.
   for (int i = 0; i < n; i++)
   {
      mp[a[i]]++;
      while (mp[a[i]] > 1)
      {
         mp[a[j]]--;
         j++;
      }
      ans = min(ans, j + n - 1 - i + min(j, n - 1 - i));
   }

   // We are returning the answer here.
   return ans;
}
```

LANGUAGE: JAVA

```
/*
   Time Complexity: O(N)
   Space Complexity: O(N)

   where 'N' is the length of the array 'A'.
*/
import java.util.*;

public class Solution {
   static int frontBack(int []a) {
      int n = a.length;

      // Creating an unordered map 'mp' of key-value pair as [int, int].
      HashMap<Integer, Integer> mp = new HashMap<>();

      // Initializing 'j' with 0 and 'ans' with 1e9.
      int j = 0, ans = 1000000000;

      // Calculating answer.
      for (int i = 0; i < n; i++) {

         // Update frequency of 'A[i]'.
         mp.put(a[i], mp.get(a[i]) == null ? 1 : mp.get(a[i]) + 1);

         // Slide pointer 'j' while the frequency of 'A[i]' is greater than 1.
         while (mp.get(a[i]) > 1) {
            mp.put(a[j], mp.get(a[j]) - 1);
            j++;
         }
         ans = Math.min(ans, j + n - 1 - i + Math.min(j, n - 1 - i));
      }

      // We are returning the answer here.
      return ans;
   }
}
```

LANGUAGE: PYTHON

```
"""
    Time complexity: O(N)
    Space complexity: O(N)

    Where 'N' is the size of the array 'A'.
"""

from typing import *

def frontBack(a: List[int]) -> int:
    n = len(a)

    # Creating a dictionary 'mp' of key-value pair as [int, int].
    mp = {}

    # Initializing 'j' with 0 and 'ans' with 1e9.
    j = 0
    ans = 10**9

    # Calculating answer.
    for i in range(n):
        mp[a[i]] = mp.get(a[i], 0) + 1
        while mp[a[i]] > 1:
            mp[a[j]] -= 1
            j += 1
        ans = min(ans, j + n - 1 - i + min(j, n - 1 - i))

    # We are returning the answer here.
    return ans
```

ANSWER 2: [20251]

LANGUAGE: C++

```
/*********************************************

  Following is the linkedList class structure:

  class List {
   public:
      int data;
      List *next;
      List() {};
      List(int v){
         data = v;
         next = NULL;
      };
  };

*********************************************/

/*
   Time Complexity : O( N )
   Space Complexity : O( 1 )

   where 'N' are the number of nodes in LinkedList.
*/

List* noLoop(List *head , int k) {

  // If 'K' is '0' whole list is in loop we will return 'NULL'.
  if( k == 0 ){
    return NULL;
  }

  // Else we will find 'K-1'th node in 'TEMP' and set its 'NEXT' to 'NULL'.
  List *temp = head;
  k -= 1;
  while(k--) {
    temp = temp->next;
  }
  temp->next = NULL;

  // Return 'HEAD'.
  return head;
}
```

LANGUAGE: JAVA

```
/*************************
 * Following is the linked list node class
   class List {

      int data;
      List next;

      List(int val) {
         this.data = val;
         next = null;
      }
   }

 ***************/

/*
   Time Complexity : O( N )
   Space Complexity : O( 1 )

   where 'N' are the number of nodes in LinkedList.
*/
public class Solution {
        public static List noLoop(List head , int k) {

                // If 'K' is '0' whole list is in loop we will return 'NULL'.
                if( k == 0 ){
                  return null;
                }

                // Else we will find 'K-1'th node in 'TEMP' and set its 'NEXT' to 'NULL'.
                List temp = head;
                k -= 1;
                while(k-- > 0) {
                  temp = temp.next;
                }
                temp.next = null;

                // Return 'HEAD'.
                return head;
        }

}
```

LANGUAGE: PYTHON

```
/*************************
 * Following is the linked list node class
   class List {

      int data;
      List next;

      List(int val) {
         this.data = val;
         next = null;
      }
   }

 ***************/

/*
   Time Complexity : O( N )
   Space Complexity : O( 1 )

   where 'N' are the number of nodes in LinkedList.
*/
public class Solution {
        public static List noLoop(List head , int k) {

                // If 'K' is '0' whole list is in loop we will return 'NULL'.
                if( k == 0 ){
                  return null;
                }

                // Else we will find 'K-1'th node in 'TEMP' and set its 'NEXT' to 'NULL'.
                List temp = head;
                k -= 1;
                while(k-- > 0) {
                  temp = temp.next;
                }
                temp.next = null;

                // Return 'HEAD'.
                return head;
        }

}
```

ANSWER 3: [23295]

LANGUAGE: C++

```
/*
   Time complexity: O(N)
   Space complexity: O(1)

   Where 'N' is the length of an input array 'A'.
*/

int minimumOperations(int n, vector<int> &a) {
   // Initializing the driver variables.
   long long left = a[0], right = a[n - 1];
   int i = 0, j = n - 1;
   int ans = 0;

   // Iterating till the current pointer is less than second.
   while (i < j) {
      // If 'left < right' then we need to increment 'i' and the left pointer.
      if (left < right) {
         i += 1;
         left += a[i];
         ans += 1;
      } else if (left > right) {
         // Here we need to decrement the 'j' and update the 'right' pointer.
         j -= 1;
         right += a[j];
         ans += 1;
      } else {
         // Case when both the pointers are equal. Increment 'i' and decrement 'j'.
         i += 1;
         j -= 1;
         left += a[i];
         right += a[j];
      }
   }

   // Returning the answer.
   return ans;
}
```

LANGUAGE: JAVA

```
/*
   Time complexity: O(N)
   Space complexity: O(1)

   Where 'N' is the length of an input array 'A'.
*/

public class Solution {
   static int minimumOperations(int n, int []a) {

      // Initializing the driver variables.
      long left = a[0], right = a[n - 1];
      int i = 0, j = n - 1;
      int ans = 0;

      // Iterating till the current pointer is less than second.
      while (i < j) {
         // If 'left < right' then we need to increment 'i' and the left pointer.
         if (left < right) {
            i += 1;
            left += a[i];
            ans += 1;
         } else if (left > right) {
            // Here we need to decrement the 'j' and update the 'right' pointer.
            j -= 1;
            right += a[j];
            ans += 1;
         } else {
            // Case when both the pointers are equal. Increment 'i' and decrement 'j'.
            i += 1;
            j -= 1;
            left += a[i];
            right += a[j];
         }
      }

      // Returning the answer.
      return ans;
   }
}
```

LANGUAGE: PYTHON

```
"""
    Time complexity: O(N)
    Space complexity: O(1)

    Where 'N' is the length of an input array 'A'.
"""



def minimumOperations(n: int, a: list) -> int:
    # Initializing the driver variables.
    left = a[0]
    right = a[n-1]
    i = 0
    j = n-1
    ans = 0

    # Iterating till the current pointer is less than second.
    while i < j:
        # If 'left < right' then we need to increment 'i' and the left pointer.
        if left < right:
            i += 1
            left += a[i]
            ans += 1
        elif left > right:
            # Here we need to decrement the 'j' and update the 'right' pointer.
            j -= 1
            right += a[j]
            ans += 1
        else:
            # Case when both the pointers are equal. Increment 'i' and decrement 'j'.
            i += 1
            j -= 1
            left += a[i]
            right += a[j]

    # Returning the answer.
    return ans
```

ANSWER 4: [17603]

LANGUAGE: C++

```
/*
   Time Complexity : O(N)
   Space Complexity : O(1)

   Where 'N' is the number of nodes.
*/

/************************************************

  Following is the linkedList class structure:

  class List {
   public:
      int data;
      List *next;
      List() {};
      List(int v){
         data = v;
         next = NULL;
      };
  };

************************************************/

List* foldAndMerge(List *head) {

  List *fast = head , *prev = NULL , *next;
  while(fast->next){

    // 'FAST' pointer jumps twice while 'HEAD' goes only once to reach middle.
    fast = fast->next->next;

    // Reversing the first half of our 'LIST'.
    next = head->next;
    head->next = prev;
    prev = head;
    head = next;
  }
  // Keeping the 'ANS' pointer as 'HEAD'.
  List *ans = head;

  // Merging the folded part of the List.
  while(prev){
    head->data = head->data * prev->data;
```

```
    prev = prev->next;
    head = head->next;
  }

  // Returning 'ANS'.
  return ans;
}
```

LANGUAGE: JAVA

```
/*
   Time Complexity : O(N)
   Space Complexity : O(1)

   Where 'N' is the number of nodes.
*/


/************************************************************

   Following is the class structure of the Node class:

   class Node
   {
      public:
         int data;
         Node next;
         Node(int data)
         {
            this.data = data;
            this.next = null;
         }
   };

************************************************************/

class Solution {

        public static Node foldAndMerge(Node head) {
                Node fast = head , prev = null , next = null;

                while(fast != null && fast.next != null){

                  // 'FAST' pointer jumps twice while 'HEAD' goes only once to reach
middle.
                  fast = fast.next.next;
```

```
            // Reversing the first half of our 'LIST'.
            next = head.next;
            head.next = prev;
            prev = head;
            head = next;
        }
        // Keeping the 'ANS' pointer as 'HEAD'.
        Node ans = head;

        // Merging the folded part of the List.
        while(prev != null){
            head.data = head.data * prev.data;
            prev = prev.next;
            head = head.next;
        }

        // Returning 'ANS'.
        return ans;

    }
}
```

LANGUAGE: PYTHON

```
'''
    Time Complexity  : O( N )
    Space Complexity : O( 1 )

    Where, N is the number of nodes.
'''


class Node:
    def __init__(self,data):
        self.data = data
        self.next = None


def foldAndMerge(head):
    fast = head
    prev = Node(None)
    while (fast):

        # 'FAST' pointer jumps twice while 'HEAD' goes only once to reach middle.
        fast = fast.next.next

        # Reversing the first half of our 'LIST'.
```

```
        next = head.next
        head.next = prev
        prev = head
        head = next

    # Keeping the 'ANS' pointer as 'HEAD'.
    ans = head

    # Merging the folded part of the List.
    while (head):
        head.data = head.data * prev.data
        prev = prev.next
        head = head.next

    # Returning 'ANS'.
    return ans
```

ANSWER 5: [29893]

LANGUAGE: C++

```
/*
    Time Complexity: O(NlogN + K)
    Space Complexity: O(N + K)

    Where 'N' is the length of the array 'A', and 'K' is the given constant.
*/

vector<int> findMagicalNumbers(int n, int k, vector<int> a) {

    // Make an integer array 'prefixSum' of the length 'N'.
    vector<long long> prefixSum(n, 0LL);

    // For 'i' in the range '0' to 'N - 1':
    for (int i = 0; i < n; i++) {

        // Add 'A[i]' to the 'prefixSum[i]'.
        prefixSum[i] += a[i];

        // If 'i' is greater than '0', then add 'prefixSum[i - 1]' to the 'prefixSum[i]'.
        if (i > 0) {
            prefixSum[i] += prefixSum[i - 1];
        }
```

```
}

// Make an integer array 'differenceArray' of the length 'K + 2'.
vector<int> differenceArray(k + 2, 0);

// For 'i' in the range '0' to 'N - 1':
for (int i = 0; i < n; i++) {

   // If 'prefixSum[i]' is greater than K, then brsk the loop.
   if (prefixSum[i] > k) {
      break;
   }
   // Initialize an integer variable 'j' with 'N'.
   int j = n;

   // Initialize an integer variable 'L' with 'i', and 'R' with 'N - 1'.
   int l = i, r = n;

   // While 'L' is less than or equal to 'R':
   while (l <= r) {

      // Initialize an integer variable 'mid' with '(L + R) / 2'.
      int mid = (l + r) / 2;

      /*
         If 'prefixSum[mid]' is greater than or equal to '2 * prefixSum[i]':
            Set 'j' equal to 'mid', and 'R' equal to 'mid - 1'.
         Else:
            Set 'L' equal to 'mid + 1'.
      */
      if (prefixSum[mid] >= 2 * prefixSum[i]) {
         j = mid;
         r = mid - 1;
      }
      else {
         l = mid + 1;
      }
   }

   // If 'j' is equal to 'N' or 'prefixSum[j] - prefixSum[i]' is greater than 'K':
   if (j == n || prefixSum[j] - prefixSum[i] > k) {

      // Increase 'differenceArray[prefixSum[i]]' by one.
      differenceArray[prefixSum[i]]++;

      // Decrease 'differenceArray[K+1]' by one.
      differenceArray[k + 1]--;
```

```
            // Break the loop.
            break;
        }

        // Increase 'differenceArray[prefixSum[i]]' by one.
        differenceArray[prefixSum[i]]++;

        // Decrease 'differenceArray[prefixSum[j] - prefixSum[i]]' by one.
        differenceArray[prefixSum[j] - prefixSum[i]]--;
    }

    // Make one integer array 'answer'.
    vector<int> answer;

    // For 'i' in the range '1' to 'K':
    for (int i = 1; i <= k; i++) {

        // Add 'differenceArray[i - 1]' to the 'differenceArray[i]'.
        differenceArray[i] += differenceArray[i - 1];

        // If 'differenceArray[i]' is greater than '0', then Append 'i' to the back of the array
'answer'.
        if (differenceArray[i] > 0) {
            answer.push_back(i);
        }
    }

    // Return 'answer'.
    return answer;
}
```

LANGUAGE: JAVA

```
/*
    Time Complexity: O(NlogN + K)
    Space Complexity: O(N + K)

    Where 'N' is the length of the array 'A', and 'K' is the given constant.
*/

import java.util.*;

public class Solution {

    static ArrayList<Integer> findMagicalNumbers(int n, int k, ArrayList<Integer> a) {
        // Make an integer array 'prefixSum' of the length 'N'.
        long[] prefixSum = new long[n+1];
```

```
// For 'i' in the range '0' to 'N - 1':
for (int i = 0; i < n; i++) {
   // Add 'A[i]' to the 'prefixSum[i]'.
   prefixSum[i] += a.get(i);

   // If 'i' is greater than '0', then add 'prefixSum[i - 1]' to the 'prefixSum[i]'.
   if (i > 0) {
      prefixSum[i] += prefixSum[i - 1];
   }
}

// Make an integer array 'differenceArray' of the length 'K + 2'.
int[] differenceArray = new int[k + 2];

// For 'i' in the range '0' to 'N - 1':
for (int i = 0; i < n; i++) {
   // If 'prefixSum[i]' is greater than K, then break the loop.
   if (prefixSum[i] > k) {
      break;
   }
   // Initialize an integer variable 'j' with 'N'.
   int j = n;

   // Initialize an integer variable 'L' with 'i', and 'R' with 'N - 1'.
   int l = i, r = n;

   // While 'L' is less than or equal to 'R':
   while (l <= r) {
      // Initialize an integer variable 'mid' with '(L + R) / 2'.
      int mid = ((l + r)/2);

      /*
         If 'prefixSum[mid]' is greater than or equal to '2 * prefixSum[i]':
            Set 'j' equal to 'mid', and 'R' equal to 'mid - 1'.
         Else:
            Set 'L' equal to 'mid + 1'.
      */
      if (prefixSum[mid] >= 2 * prefixSum[i]) {
         j = mid;
         r = mid - 1;
      } else {
         l = mid + 1;
      }
   }

   // If 'j' is equal to 'N' or 'prefixSum[j] - prefixSum[i]' is greater than 'K':
   if (j == n || prefixSum[j] - prefixSum[i] > k) {
      // Increase 'differenceArray[prefixSum[i]]' by one.
```

```
                differenceArray[(int)prefixSum[i]]++;

                // Decrease 'differenceArray[K+1]' by one.
                differenceArray[k + 1]--;

                // Break the loop.
                break;
            }

            // Increase 'differenceArray[prefixSum[i]]' by one.
            differenceArray[(int)prefixSum[i]]++;

            // Decrease 'differenceArray[prefixSum[j] - prefixSum[i]]' by one.
            differenceArray[(int)(prefixSum[j] - prefixSum[i])]--;
        }

        // Make one integer array 'answer'.
        ArrayList<Integer> answer = new ArrayList<>();

        // For 'i' in the range '1' to 'K':
        for (int i = 1; i <= k; i++) {
            // Add 'differenceArray[i - 1]' to the 'differenceArray[i]'.
            differenceArray[i] += differenceArray[i - 1];

            // If 'differenceArray[i]' is greater than '0', then Append 'i' to the back of the
array 'answer'.
            if (differenceArray[i] > 0) {
                answer.add(i);
            }
        }

        // Return 'answer'.
        return answer;
    }
}
```

LANGUAGE: PYTHON

```
"""
Time Complexity: O(NlogN + K)
Space Complexity: O(N + K)

Where 'N' is the length of the array 'a', and 'K' is the given constant.
"""
from typing import List

def findMagicalNumbers(n: int, k: int, a: List[int]) -> List[int]:
    # Make a list 'prefixSum' of the length 'N'.
```

```
prefixSum = [0] * (n + 1)

# For 'i' in the range '0' to 'N - 1':
for i in range(n):

    # Add 'a[i]' to 'prefixSum[i]'.
    prefixSum[i] += a[i]

    # If 'i' is greater than '0', then add 'prefixSum[i - 1]' to 'prefixSum[i]'.
    if i > 0:
        prefixSum[i] += prefixSum[i - 1]

# Make a list 'differenceArray' of the length 'K + 2'.
differenceArray = [0] * (k + 2)

# For 'i' in the range '0' to 'N - 1':
for i in range(n):

    # If 'prefixSum[i]' is greater than K, then break the loop.
    if prefixSum[i] > k:
        break

    # Initialize an integer variable 'j' with 'N'.
    j = n

    # Initialize integer variables 'L' with 'i', and 'R' with 'N - 1'.
    l, r = i, n

    # While 'L' is less than or equal to 'R':
    while l <= r:

        # Initialize an integer variable 'mid' with '(L + R) // 2'.
        mid = (l + r) // 2

        # If 'prefixSum[mid]' is greater than or equal to '2 * prefixSum[i]':
        # Set 'j' equal to 'mid', and 'R' equal to 'mid - 1'.
        if prefixSum[mid] >= 2 * prefixSum[i]:
            j = mid
            r = mid - 1
        else:
            l = mid + 1

    # If 'j' is equal to 'N' or 'prefixSum[j] - prefixSum[i]' is greater than 'K':
    if j == n or prefixSum[j] - prefixSum[i] > k:
        # Increase 'differenceArray[prefixSum[i]]' by one.
        differenceArray[prefixSum[i]] += 1

        # Decrease 'differenceArray[K+1]' by one.
```

```
            differenceArray[k + 1] -= 1

            # Break the loop.
            break

        # Increase 'differenceArray[prefixSum[i]]' by one.
        differenceArray[prefixSum[i]] += 1

        # Decrease 'differenceArray[prefixSum[j] - prefixSum[i]]' by one.
        differenceArray[prefixSum[j] - prefixSum[i]] -= 1

    # Make a list 'answer'.
    answer = []

    # For 'i' in the range '1' to 'K':
    for i in range(1, k + 1):

        # Add 'differenceArray[i - 1]' to 'differenceArray[i]'.
        differenceArray[i] += differenceArray[i - 1]

        # If 'differenceArray[i]' is greater than '0':
        # Append 'i' to the back of the list 'answer'.
        if differenceArray[i] > 0:
            answer.append(i)

    # Return 'answer'.
    return answer
```

ANSWER 6: [30774]

LANGUAGE: C++

```
/*
    Time Complexity: O(logN * log10(N)).
    Space Complexity: O(1).

    Where 'N' is the constant described in the statement.
*/
```

```
bool isGood(long long x, long long m) {

   // Initialize an integer variable 'temp' with 'X', and 'sum' with '0'.
   long long temp = x, sum = 0;

   // While 'temp' is greater than '0':
   while (temp > 0) {

      // Add 'temp % 10' into the 'sum'.
      sum += (temp % 10);

      // Set 'temp' equal to 'temp / 10'.
      temp = (temp / 10);
   }

   // If 'X - sum' is greater than 'M', then return 'false'.
   if (x - sum > m) {
      return false;
   }

   // Return 'true'.
   return true;
}

long long countGoodNumbers(long long n, long long m) {

   // Initialize an integer variable 'L' with '1', and 'R' with 'N'.
   long long l = 1, r = n;

   // Initialize an integer variable 'answer' with '0'.
   long long answer = 0;

   // While 'L' is not equal to 'R':
   while (l <= r) {

      // Initialize an integer variable 'mid' with '(L + R) / 2'.
      long long mid = (l + r) / 2;

      // If 'isGood(mid, M)' is equal to 'true':
      if (isGood(mid, m) == true) {

         // Set 'answer' equal to 'mid'.
         answer = mid;

         // Set 'L' equal to 'mid+1'.
         l = mid + 1;
      }
      else {
```

```
            // Set 'R' equal to 'mid-1'.
            r = mid - 1;
        }
    }

    // Return '(answer * (n - answer)) modulo (10^9 + 7)'.
    return (answer * (n - answer)) % (1000000007);
}
```

LANGUAGE: JAVA

```
/*
    Time Complexity: O(logN * log10(N))
    Space Complexity: O(1)

    Where 'N' is the constant described in the statement.
*/

import java.util.HashMap;

public class Solution {

    static boolean isGood(long x, long m) {
        // Initialize an integer variable 'temp' with 'X', and 'sum' with '0'.
        long temp = x, sum = 0;

        // While 'temp' is greater than '0':
        while (temp > 0) {
            // Add 'temp % 10' into the 'sum'.
            sum += (temp % 10);

            // Set 'temp' equal to 'temp / 10'.
            temp = (temp / 10);
        }

        // If 'X - sum' is greater than 'M', then return 'false'.
        return x - sum <= m;
    }

    static long countGoodNumbers(long n, long m) {
        // Initialize an integer variable 'L' with '1', and 'R' with 'N'.
        long l = 1, r = n;

        // Initialize an integer variable 'answer' with '0'.
        long answer = 0;
```

```
        // While 'L' is not equal to 'R':
        while (l <= r) {
            // Initialize an integer variable 'mid' with '(L + R) / 2'.
            long mid = (l + r) / 2;

            // If 'isGood(mid, M)' is equal to 'true':
            if (isGood(mid, m)) {
                // Set 'answer' equal to 'mid'.
                answer = mid;

                // Set 'L' equal to 'mid+1'.
                l = mid + 1;
            } else {
                // Set 'R' equal to 'mid-1'.
                r = mid - 1;
            }
        }

        // Return '(answer * (n - answer)) modulo (10^9 + 7)'.
        return (answer * (n - answer)) % (1000000007);
    }
}
```

LANGUAGE: PYTHON

```
'''
    Time complexity: O( N )
    Space complexity: O( 1 )

    where 'N' is the length of the string 's'.
'''

from typing import *

def is_good(x, m):

    # Initialize an integer variable 'temp' with 'x', and 'sum' with '0'.
    temp = x
    sum_val = 0

    # While 'temp' is greater than '0':
    while temp > 0:
        # Add 'temp % 10' into the 'sum'.
        sum_val += temp % 10

        # Set 'temp' equal to 'temp // 10'.
        temp //= 10
```

```
    # If 'x - sum' is greater than 'm', then return 'False'.
    if x - sum_val > m:
        return False

    # Return 'True'.
    return True

def count_good_numbers(n: int, m: int) -> int:

    # Initialize an integer variable 'l' with '1', and 'r' with 'n'.
    l = 1
    r = n

    # Initialize an integer variable 'answer' with '0'.
    answer = 0

    # While 'l' is not equal to 'r':
    while l <= r:
        # Initialize an integer variable 'mid' with '(l + r) // 2'.
        mid = (l + r) // 2

        # If 'is_good(mid, m)' is equal to 'True':
        if is_good(mid, m):
            # Set 'answer' equal to 'mid'.
            answer = mid

            # Set 'l' equal to 'mid + 1'.
            l = mid + 1
        else:
            # Set 'r' equal to 'mid - 1'.
            r = mid - 1

    # Return '(answer * (n - answer)) % (10^9 + 7)'.
    return (answer * (n - answer)) % (10**9 + 7)
```