Hackerrank solutions

1 . Compare the Triplets

Sol : if alice > bob increment alice , else bob if equal continue;

```
ArrayList<Integer>alice=new ArrayList<Integer>();
 alice.add(a);
 alice.add(b);
 alice.add(c);
ArrayList<Integer>bob=new ArrayList<Integer>();
 bob.add(m);
 bob.add(n);
 bob.add(p);


int alice_score=0;
int bob_score=0;

for(int i=0;i<alice.size();i++){
    if(alice.get(i)>bob.get(i)){
        alice_score+=1;
    }else if(alice.get(i)<bob.get(i)){
        bob_score+=1;
    }
}
System.out.print(alice_score+" "+ bob_score);
```

2. A very Big Sum

Sol : Loop through Array and add , as it wont fit in integer due to constraint use long

```
long sum=0;
    for(int i=0;i<n;i++){
        long value=in.nextLong();
        sum+=value;
    }

    System.out.print(sum);
  }
```

3. Diagonal difference

Sol: pick up all elements , if it is diag , add to left or right based on position → (i,j)

```
System.out.println("Diff between summation of leftr_dig and right_dig");
     if(left_sum > right_sum){
         System.out.print(left_sum - right_sum);
     }else{
         System.out.print(right_sum - left_sum);
     }
```

4. Plus Minus

Sol : Loop through array , count positive , negative and zero

Divide by array length and print proportions

```
for(int i=0;i<arr.size();i++){
        if(arr.get(i)<0){
           neg++;
        }else if(arr.get(i)>0){
           pos++;
        }else{
           zero++;
        }
    }
    neg/=arr.size();
    pos/=arr.size();
    zero/=arr.size()
```

5. Staircase

Sol : take n , print spaces and # accordingly → use nested loops

```
for(int i=1;i<=n;i++){
    //   space
    for(int j=1;j<=n-i;j++){
       System.out.print(" ");
    }
    // hash
     for(int k=1;k<=i;k++){
       System.out.print("#");
    }
    System.out.println();
  }
```

6. MINI MAX SUM

Sol : check if sorted , total sum - first element and total sum - last element

Else total sum - a[i] repeat for all elements pick small and large sum

```
int sum=0;

  for(int i=0;i<arr.size();i++){
     sum+=arr.get(i);
  }
  int mn= sum-arr.get(arr.size()-1);
  int mx= sum-arr.get(0);
```

7. Birthday Cake Candles

Sol: i keep lowest value , loop through array , if a[i] > this i update and maintain count of this number , if i encounter another number which is greater update max and count updated accordingly .

Else : find greatest number , count the frequency

```
int mx=0;
int count=0;
for(int i=0;i<arr.size();i++){
    mx= Math.max(mx,arr.get(i));
}

for(int i=0;i<arr.size();i++){
    if(mx==arr.get(i)){
        count++;
    }
}
System.out.println(count);
```

8. Time conversion

Sol: look at pm or am , if pm make it 12 and add 12 for other in pm , if am make it 00 and remaining as it is

```
if (s.charAt(8) == 'P') {
    String portion = s.substring(0, 8); // Extracts HH:MM:SS
    String[] time = portion.split(":"); // Splits time into [HH, MM, SS]
    if (time[0].equals("12")) { // Check for noon
        System.out.println(time[0] + ":" + time[1] + ":" + time[2]);
    } else { // Convert to 24-hour format
        time[0] = String.valueOf(Integer.parseInt(time[0]) + 12);
        System.out.println(time[0] + ":" + time[1] + ":" + time[2]);
    }
} else { // AM case
    String portion = s.substring(0, 8); // Extracts HH:MM:SS
    String[] time = portion.split(":"); // Splits time into [HH, MM, SS]
    if (time[0].equals("12")) { // Check for midnight
        time[0] = "00";
        System.out.println(time[0] + ":" + time[1] + ":" + time[2]);
    } else { // No conversion needed
        System.out.println(portion);
    }
}
```

9. Grading Students

Sol : check if it is 37 or less than 37 → fail , else % 5 gives excess if this is less than 3 round to near by value

```
for(int i=0;i<n;i++){
        int num=grade.get(i);
        if(num>=38 && num%5>=3){
          num=num+5-num%5;
        }
        ans.add(num);
    }
```

10. Apple and Orranges

Sol; if apple and orrange lie in between house print them how add start point + d for apples , for oranges add d of orange , we get all points , if there exists a point such that apple is >= house start and <= house end similarly orange count <=house end and >= house start

```
        int apple_count=0;
    int orange_count=0;
    for(int i=0;i<apple_d.size();i++){
        int dist=a+apple_d.get(i);
        if(dist>=s && dist<=t){
            apple_count++;
        }
    }
    for(int i=0;i<orange_d.size();i++){
        int dist=b+orange_d.get(i);
        if(dist>=s && dist<=t){
            orange_count++;
        }
    }
```

11. Kangaroo jump

Sol : check velocity of second as it far from first . if great than first one velocity print No , else check if difference between start points can be covered by difference between velocities

```
if(v1<v2){
        System.out.println("No");
        return;
    }
    if((x2-x1)%(v1-v2)==0){
       System.out.println("Yes");
    }else{
        System.out.println("No");
    }
  }
```

12. Breaking Records

Sol : initial game point is max and min , if great change max , update max count , if equal continue , if less than initial update min and count min

```
 int mx=arr.get(0);
     int mn=arr.get(0);

     int mx_count=0;
     int mn_count=0;
     for(int i=1;i<arr.size();i++){
        if(arr.get(i)>mx){
           mx=arr.get(i);
           mx_count++;
        }else if(arr.get(i)<mn){
           mn=arr.get(i);
           mn_count++;
        }
     }
```

13. Subarray division

Start from first square , look for how many squares you need and subtract a[i] , look if value and sum you need matches with next element , if yes increment count and start from second box from now.

```
for(int i=0;i<arr.size();i++){
        int sum=0;
        int length_count=0;
        for(int j=i;j<j+m;j++){
           if(j>=n || sum>=d){
              break;
           }
           sum+=arr.get(j);
           length_count++;
        }
        if(sum==d && length_count==m){
           count++;
        }
     }
     System.out.println(count);
```

14. Divisible sum pairs

Sol ; 2 loops inner + outer % k increment count

```
int count=0;
     for(int i=0;i<arr.size();i++){
        for(int j=i+1;j<arr.size();j++){
           int sum=arr.get(i)+arr.get(j);
           if(sum%k==0){
              count++;
           }
        }
     }
     System.out.println(count);
```

15. Migratory Birds

Sol: 5 types of birds , count their frequencies and if equal freq pick which is lowest

```
HashMap<Integer, Integer> mp = new HashMap<>();
for (int i = 0; i < arr.size(); i++) {
   int num = arr.get(i);
   mp.put(num, mp.getOrDefault(num, 0) + 1); // Build frequency map
}

int mx = 0; // Maximum frequency
int ans = Integer.MAX_VALUE; // Initialize with a large value to find the smallest key

for (Integer key : mp.keySet()) {
   int value = mp.get(key); // Get frequency of the current key
   if (value > mx) {
      // If this key has a higher frequency, update max frequency and answer
      mx = value;
      ans = key;
   } else if (value == mx) {
      // If frequencies are the same, choose the smaller key
      if (key < ans) {
         ans = key;
      }
   }
}

System.out.println(ans);
```

16. Bill division

Sol : calculate total sum , subtract one item and divide / 2 . check bill , if excess refund balance

```
int sum=0;
      for(int i=0;i<arr.size();i++){
         sum+=arr.get(i);
      }
      sum=sum-arr.get(ind);
      sum=sum/2;
      if(b>sum){
          System.out.println(b-sum);
      }else{
          System.out.println("Bon Apetite");
      }
```

17. Sales by match

Sol: freq of all socks , loop through hashmap , add socn/2 -> no of pairs

```
HashMap<Integer,Integer>mp=new HashMap<>();
    for(int i=0;i<n;i++){
        int color=arr.get(i);
        if(mp.containsKey(color)){
            mp.put(color,mp.get(color)+1);
        }else{
            mp.put(color,1);
        }
    }

    int pair=0;
    for(Integer key:mp.keySet()){
        int value=mp.get(key);
        pair=pair+(value/2);
    }
    System.out.println(pair);
```

18. Drawing Book

Sol: total number / 2 = total flips , total - front flip = back , take min front|back

Number /2 gives front flips

```
 int totalflip= n/2;
      int targetflipFront=p/2;
      int targetflipBack=totalflip-targetflipFront;
      int ans= Math.min(targetflipFront,targetflipBack);
      System.out.println(ans);
    }
```

19. Between 2 sets

Sol : in array check lcm

In range of last element from arr1 and first element of arr2 look for GCD

```
import java.util.*;

public class BetweenTwoSets {

    // Helper method to calculate GCD of two numbers
    public static int gcd(int a, int b) {
        if (b == 0) return a;
        return gcd(b, a % b);
    }

    // Helper method to calculate GCD of an array
    public static int gcdArray(int[] arr) {
        int result = arr[0];
        for (int i = 1; i < arr.length; i++) {
            result = gcd(result, arr[i]);
            if (result == 1) return 1; // Early exit if GCD is 1
        }
        return result;
    }

    // Helper method to calculate LCM of two numbers
    public static int lcm(int a, int b) {
        return (a * b) / gcd(a, b);
    }

    // Helper method to calculate LCM of an array
    public static int lcmArray(int[] arr) {
        int result = arr[0];
        for (int i = 1; i < arr.length; i++) {
            result = lcm(result, arr[i]);
        }
        return result;
    }

    public static int getTotalX(int[] A, int[] B) {
        // Step 1: Calculate LCM of A
        int lcmA = lcmArray(A);

        // Step 2: Calculate GCD of B
        int gcdB = gcdArray(B);

        // Step 3: Count numbers between LCM(A) and GCD(B) that satisfy conditions
        int count = 0;
        for (int i = lcmA; i <= gcdB; i += lcmA) { // Step through multiples of LCM(A)
            if (gcdB % i == 0) { // Check if GCD(B) is divisible by the number
sout(gcdB);
```

```
                count++;
            }
        }

        return count;
    }

    public static void main(String[] args) {
        int[] A = {2, 4};
        int[] B = {16, 32, 96};

        int result = getTotalX(A, B);
        System.out.println("Total numbers between the two sets: " + result);
    }
}
```

20. Cats and mouse

Check distance which is smaller , print that name

```
 int distA=Math.abs(x-z);
      int distB=Math.abs(y-z);
      if(distA<distB){
         System.out.println("Cat A");
      }else if(distB <distA){
         System.out.println("Cat B");
      }else{
         System.out.println("Mouce C");
      }
```

21. Day of programmer ‘

Sol : check years , < 1918 divisible by 4 , > 1918 400 or 4 and not 100, == 1918 . print day accordingly ‘ see if it is leap year

```
 if(year<1918){
         date+=(year%4==0)?"12.09."+ year :"13.09."+year;
      }else if(year==1918){
         date+="26.09."+year;
      }else{
         date+=((year%400==0) || (year%4==0 && year%100!=0))?"12.09."+year
:"13.09."+year;
      }
```

22. Counting valley

Sol : if it valley we move from down to climp up , while climb is finished check if we reached destination

```
int level=0;
      int valley_count=0;
      for(int i=0;i<v.length();i++){
          if(v.charAt(i)=='U'){
              Level++;
             if(level==0){
                  valley_count++;
              }
          }else if(v.charAt(i)=='D'){
                          level--;
          }
      }
      System.out.println(valley_count);
    }
```

23. Hurdle Race

Sol : Check max height in array , check height and max height print diff iif negative print 0

```
 int mx=0;
      for(int i=0;i<arr.size();i++){
          mx=Math.max(mx,arr.get(i));
      }

      if(mx>h){
          System.out.println(mx-h);
      }else{
          System.out.println(0);
      }
```

24. Designer PDF Viewer

Sol : convert word to char array , loop through each char subtract ascii use this to get value from freq array , after getting this array . loop through this get max and cal using formula

```
for(int i=0;i<n;i++){
          int value=in.nextInt();
          h[i]=value;
      }

      String word=in.nextLine();
      int mx=0;
      for(int i=0;i<word.length();i++){
          if(h[word.charAt(i)-97]>mx){
              mx=h[word.charAt(i)-97];
          }
      }
      System.out.println(mx*word.length());
```

25.Electronics shop

Optim sol : Sort array , loop through second array , for each value do binary search on array 1 and get expensive option , update max . see for global max option

Bruteforce :

```
int mx=-1;
    for(int i=0;i<keyboard.size();i++){
        for(int j=0;j<usb.size();j++){
            int pair=keyboard.get(i)+usb.get(j);
            if(pair<=b){
                mx=Math.max(mx,pair);
            }
        }
    }
    System.out.println(mx);
```

26. Utopian Tree

Sol : even add 1 , odd multiply by 2 . return height

```
int height=1;
    for(int i=1;i<=n;i++){
        if(i%2!=0){
            height*=2;
        }else{
            height+=1;
        }
    }
    System.out.println(height);
```

27 . Angry Prof

Sol ; check if val <= 0 , if yes increment count , count >= k no cancellation

```
int count=0;
    for(int i=0;i<arr.size();i++){
        if(arr.get(i)<=0){
            count++;
        }
    }
    if(count>=k){
        System.out.println("No");
    }else{
        System.out.println("Yes");
    }
```

28. Beautiful Tree

Sol : start to end , start - rev(start) % k ==0 increment count

```
int count=0;
    for(int i=s;i<=e;i++){
        int num=i;
        int rev=0;
        while(num!=0){
            int rem=num%10;
            rev=rev*10+rem;
            num=num/10;
        }
        int diff= Math.abs(i-rev);
        if(diff%k==0){
            count++;
        }
    }
    System.out.println(count);
```

29. Strange Advertise

Sol : initial 5 people , share ½ of people , add this share across days return accumulated sum of shares , people increase by multiply by 3 as share people chain increases

```
 int people= 5;
    int total_like=0;
    for(int i=1;i<=n;i++){
      int share= (people/2);
      total_like+=share;
      people= share*3;
    }
    System.out.println(total_like);
```

30.Cut the stick

Sol (opt) sort the array cal min , start from index 1 add count , subtract this val from array , count only a[i]!= 0 after sub . repeat

```
      // Step 1: Sort the sticks
      Arrays.sort(sticks);

      List<Integer> result = new ArrayList<>();
      int n = sticks.length;

      // Step 2: Iterate through the array
      for (int i = 0; i < n; i++) {
```

```
            if (i == 0 || sticks[i] != sticks[i - 1]) {
                result.add(n - i);              }
        }

        return result;
    }
```

31. Sequence eq ‘

Sol: maintain position array , y = Pos[pos[x]]

```
int n=in.nextInt();
    HashMap<Integer,Integer>mp=new HashMap<>();
    for(int i=1;i<=n;i++){
        int value=in.nextInt();
        mp.put(value,i);
    }
    for(int i=1;i<=n;i++){
        int x=mp.get(i);
        int y=mp.get(x);
        System.out.println(y);
    }
```

32. Find Digits

Sol : get rem and if rem ! = 0 and num should divide rem

```
Num = n;
int count=0;
    while(num!=0){
        int rem=num%10;
        if(rem!=0 && n%rem==0){
            count++;
        }
        num=num/10;
    }
    System.out.println(count);
```

33. Missing Number from 2 arrays

Sol : sort 2 arrays , original list freq , loop through first array , if element exist decrement 1 from freq . loop this map and if value ==1 print its key

```
 Collections.sort(arr1);
    Collections.sort(arr2);

    HashMap<Integer,Integer>mp=new HashMap<>();

    for(int i=0;i<arr2.size();i++){
        if(mp.containsKey(arr2.get(i))){
            mp.put(arr2.get(i),mp.get(arr2.get(i))+1);
```

```
        }else{
            mp.put(arr2.get(i),1);
        }
    }
    for(int i=0;i<arr1.size();i++){
        if(mp.containsKey(arr1.get(i))){
            mp.put(arr1.get(i),mp.get(arr1.get(i))-1);
        }
    }
    mp.forEach((k,v)->{
        if(v==1){
            System.out.print(k+" ");
        }
    });
```

34. Middle element -> median

Sol; sort and get the middle index , print element at that index

```
Collections.sort(arr);
    int mid_ind=arr.size()/2;
    System.out.println(arr.get(mid_ind));
  }
```

35. Rotate array k times

Sol; n%k - times rotation , pick last element , swap last elem with elem before , place last elem at front after rotation

Rotate first k elements , rotate rem elements , rotate entire array ‘

```
public class ArrayRotation {

  // Function to reverse a portion of an array
  public static void reverse(int[] arr, int start, int end) {
    while (start < end) {
      int temp = arr[start];
      arr[start] = arr[end];
      arr[end] = temp;
      start++;
      end--;
    }
  }

  public static void leftRotateInPlace(int[] arr, int k) {
    int n = arr.length;
    k = k % n; // Handle cases where k > n

    // Step 1: Reverse the first k elements
```

```
        reverse(arr, 0, k - 1);

        // Step 2: Reverse the remaining n-k elements
        reverse(arr, k, n - 1);

        // Step 3: Reverse the entire array
        reverse(arr, 0, n - 1);
    }
```

36. Closest Numbers ‘

Sol; get min diff in array , in array list add keys whose diff == min_diff

```
        Collections.sort(arr);
    HashMap<Integer,Integer>mp=new HashMap<>();
    int min_diff=Integer.MAX_VALUE;
    for(int i=0;i<n-1;i++){
        int diff=arr.get(i+1)-arr.get(i);
        mp.put(i,diff);
        min_diff=Math.min(min_diff,diff);
    }
    ArrayList<Integer>ans=new ArrayList<Integer>();
    for(Integer key:mp.keySet()){
        if(mp.get(key)==min_diff){
            ans.add(arr.get(key));
            ans.add(arr.get(key+1));
        }
    }

    for(int i=0;i<ans.size();i++){
        System.out.print(ans.get(i)+" ");
    }
```

37. Sherlock and array

Sol; total sum cal , start from index , remove value from total check if both sum are equal , sub from total .

```
int total_sum=0;
    for(int i=0;i<arr.size();i++){
        total_sum+=arr.get(i);
    }
    boolean flag=false;
    int curr_sum=0;
    for(int i=0;i<arr.size();i++){
        curr_sum+=arr.get(i);
        if(curr_sum==total_sum - arr.get(i){
```

```
        flag=true;
        break;
    }
    total_sum-=arr.get(i);
}
if(flag==true){
    System.out.println("Yes");
}else{
   System.out.println("No");
}
```