HackerRank Format Challenges

# Longest Increasing Subsequence

\*\*Description:\*\* Return the length of the longest strictly increasing subsequence in a given list of integers.

\*\*Problem Statement:\*\*  
Given an array of integers `nums`, your task is to determine the length of the longest strictly increasing subsequence. The elements of the subsequence must maintain relative ordering but need not be contiguous.

\*\*Input Format:\*\*  
A single line containing space-separated integers representing the array `nums`.

\*\*Constraints:\*\*  
1 ≤ n ≤ 1000  
-10⁴ ≤ nums[i] ≤ 10⁴

\*\*Output Format:\*\*  
An integer representing the length of the longest increasing subsequence.

\*\*Tags:\*\* dynamic-programming, arrays, subsequence

\*\*Test Cases:\*\*

1. Input:  
   10 9 2 5 3 7 101 18  
   Output:  
   4
2. Input:  
   0 1 0 3 2 3  
   Output:  
   4
3. Input:  
   7 7 7 7 7 7 7  
   Output:  
   1
4. Input:  
   1 3 6 7 9 4 10 5 6  
   Output:  
   6
5. Input:  
   5 4 3 2 1  
   Output:  
   1

# Maximum Sum Circular Subarray

\*\*Description:\*\* Find the maximum possible sum of a non-empty subarray in a circular integer array.

\*\*Problem Statement:\*\*  
You are given a circular array of integers `nums`. A subarray is a contiguous part of the array. You may choose a subarray that wraps around the end to the beginning. Your task is to return the maximum possible sum of such a subarray.

\*\*Input Format:\*\*  
A single line containing space-separated integers representing the array `nums`.

\*\*Constraints:\*\*  
1 ≤ n ≤ 30,000  
-30,000 ≤ nums[i] ≤ 30,000

\*\*Output Format:\*\*  
An integer representing the maximum sum of any circular subarray.

\*\*Tags:\*\* kadane-algorithm, circular-array, arrays, maximum-subarray

\*\*Test Cases:\*\*

1. Input:  
   1 -2 3 -2  
   Output:  
   3
2. Input:  
   5 -3 5  
   Output:  
   10
3. Input:  
   -3 -2 -3  
   Output:  
   -2
4. Input:  
   3 -1 2 -1  
   Output:  
   4
5. Input:  
   8 -1 3 4  
   Output:  
   14

# Minimum Number of Perfect Squares

\*\*Description:\*\* Return the least number of perfect square numbers which sum to a given number `n`.

\*\*Problem Statement:\*\*  
Given a number `n`, return the minimum number of perfect square numbers (like 1, 4, 9, 16, ...) which sum to `n`. You can reuse square numbers multiple times.

\*\*Input Format:\*\*  
A single integer `n`.

\*\*Constraints:\*\*  
1 ≤ n ≤ 10,000

\*\*Output Format:\*\*  
An integer representing the minimum number of perfect square numbers that sum to `n`.

\*\*Tags:\*\* dynamic-programming, math, perfect-squares

\*\*Test Cases:\*\*

1. Input:  
   12  
   Output:  
   3
2. Input:  
   13  
   Output:  
   2
3. Input:  
   1  
   Output:  
   1
4. Input:  
   43  
   Output:  
   3
5. Input:  
   100  
   Output:  
   1