Solution Code

```
def eraseOverlapIntervals(intervals):
    """
    Function to return the minimum number of intervals to remove
    to make the remaining intervals non-overlapping.

    Parameters:
    intervals (List[List[int]]): List of intervals where each interval is represented as [start, end].

    Returns:
    int: Minimum number of intervals to remove.
    """
    # Step 1: Sort intervals by their ending time
    intervals.sort(key=lambda x: x[1])

    # Step 2: Initialize variables
    # end holds the end time of the last added interval to the non-overlapping set
    end = float('-inf')
    removals = 0  # Counts the number of intervals to remove

    # Step 3: Iterate through intervals
    for interval in intervals:
        # If the start of the current interval is at least 'end', it doesn't overlap
        if interval[0] >= end:
```

```
            end = interval[1]  # Update end to the current interval's end
        else:
            # If it overlaps, increment removals count
            removals += 1

    return removals
```

```
def maxSum(A, B):
    MOD = 10**9 + 7
    i, j = 0, 0
    sumA, sumB = 0, 0  # Initialize sums for paths A and B

    # Traverse both arrays with two pointers
    while i < len(A) and j < len(B):
        if A[i] < B[j]:
            # Collect treasure from path A
            sumA += A[i]
            i += 1
        elif A[i] > B[j]:
            # Collect treasure from path B
            sumB += B[j]
            j += 1
        else:
```

```
        # Encounter a common stone, choose the max path and add the stone value
        max_sum = max(sumA, sumB) + A[i]
        sumA, sumB = max_sum, max_sum  # Update both sums to the same value
        i += 1
        j += 1

# Add any remaining treasures from path A or B
while i < len(A):
    sumA += A[i]
    i += 1

while j < len(B):
    sumB += B[j]
    j += 1

# The maximum score path
return max(sumA, sumB) % MOD
```