```go
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

package main

type Item struct {
  ArrayIdx int
  Num      int
}

// O(nk) time | O(n + k) space - where where n is the total
// number of array elements and k is the number of arrays
func MergeSortedArrays(arrays [][]int) []int {
  sortedList := []int{}
  elementIdxs := make([]int, len(arrays))

  for {
    smallestItems := []Item{}
    for arrayIdx := 0; arrayIdx < len(arrays); arrayIdx++ {
      relevantArray := arrays[arrayIdx]
      elementIdx := elementIdxs[arrayIdx]
      if elementIdx == len(relevantArray) {
        continue
      }
      smallestItems = append(smallestItems, Item{
        ArrayIdx: arrayIdx,
        Num:      relevantArray[elementIdx],
      })
    }

    if len(smallestItems) == 0 {
      break
    }
    nextItem := getMinValue(smallestItems)
    sortedList = append(sortedList, nextItem.Num)
    elementIdxs[nextItem.ArrayIdx] += 1
  }
  return sortedList
}

func getMinValue(items []Item) Item {
  minValueItem := items[0]
  for i := 1; i < len(items); i++ {
    if items[i].Num < minValueItem.Num {
      minValueItem = items[i]
    }
  }
  return minValueItem
}

```