```python
# Counting Inversion

import sys
import argparse

def counting_inversion_brute(nums):
    swaps = []
    for i in xrange(len(nums)):
        for j in xrange(i, len(nums)):
            if nums[i] > nums[j]:
                swaps.append((nums[i], nums[j]))
                nums[j], nums[i] = nums[i], nums[j]
    return swaps

def counting_inversion_divide_and_conq(nums):
    def merge_and_count(a, b):
        i = j = 0
        c = []
        count = 0

        while i < len(a) and j < len(b):
            c.append(min(a[i], b[j]))
            if b[j] < a[i]:
                count += len(a) - i
                j += 1
            else:
                i += 1

        # append the list of b onto a
        if (i >= len(a)) and j < len(b):
            c += b[j:]
        elif (j >= len(b)) and i < len(a):
            c += a[i:]
        return count, c

    def sort_and_count(l):
        if len(l) == 1:
            return 0, l
        else:
            # divide l into a and b
            mid = len(l)/2
            a = l[:mid]
            b = l[mid:]

            r_a, a = sort_and_count(a)
            r_b, b = sort_and_count(b)
            r, l = merge_and_count(a, b)

            return r_a + r_b + r, l

    return sort_and_count(nums)[0]

def parse_numbers_file(filename):
    nums = []
    with open(filename) as f:
        for l in f:
            try:
                nums.append(int(l))
            except ValueError:
                pass
        return nums

def main():
    # parse command line arguments
    parser = argparse.ArgumentParser(
        description="The number of inversions to get a sorted sequence.")
    parser.add_argument('filename', metavar='FILE', type=str,
                        help='The file to parse the numbers out of. Expects a '+\
                             'file of on number on each line.')
    args = parser.parse_args()

    # open the file and run the algorithm
    nums = parse_numbers_file(args.filename)

    # swaps = counting_inversion_brute(nums[:])
    # # print "Brute force #:  ", len(swaps)
    # print "Sequence:  <%s>" % (', '.join(map(str, nums)),)
    # print "Swaps (found with brute force):"
    # for swap in swaps:
    #     print "Swap:  %s" % (repr(tuple(swap)),)
    # print

    num_swaps = counting_inversion_divide_and_conq(nums[:])
    print "# Swaps (Found with divide and conquer):", num_swaps

if __name__ == "__main__":
    main()
```