Counting Inversions

- ex: music site tries to match your song preferences with others

- separate list into two pieces
- count inversions in each half recursively
- count inversions in different halves, and return the sum of the three quantities
  - assume each half is sorted
  - count inversions where a_i & a_j are in different halves
  - merge two sorted halves into sorted whole

run time: $T(n) \leq T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + O(n) = O(n \log n)$

Pre-condition: A & B are sorted
Post-condition: L is sorted
sortAndCount(L) {
  if list L has one element
    return 0 & list L
  Divide the list in two halves A & B
  (r_A, A) = sort & count (A)
  (r_B, B) = sort & count (B)
  (r, L) = merge & count (A, B)
return $r = r_A + r_B + r$ and sorted list L
}