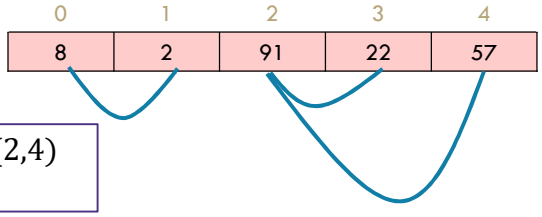
Counting Inversions

Given an array, A, determine how "unsorted" it is, by counting number of inversions:

Inversion: pair i, j such that $i < j$ but $A[i] > A[j]$



$(0,1)$, $(2,3)$, and $(2,4)$ are inversions

Intuitively, how many adjacent swaps to fully sort.

Why? Tell "how different" two lists are (e.g. tell if someone's opinion is an outlier, or if two people have similar preferences)

Pause

Lets get some intuition

$O(n \log n)$ is closest to which of these:

$O(n)$

$O(n^{1.1})$

$O(n\sqrt{n})$

$O(n^2)$

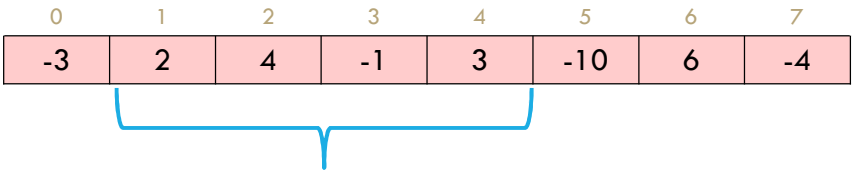
Almost there…

```
int CountInversions(A, int start, int stop)
  inversions = 0
  if(start >= stop)
    return 0
  int midpoint = (stop-start)/2 + start
  inversions += CountInversions(A, start, midpoint)
  inversions += CountInversions(A, midpoint+1, end)
  sort(A, midpoint+1, end)
  for(int i=start; i <= midpoint; i++)
    int k = binarySearch(A, midpoint+1, end, i)
    inversions += k-(midpoint+1)+1
  return inversions
```

Another divide and conquer

Maximum subarray sum

Given: an array of integers (positive and negative), find the indices that give the maximum contiguous subarray sum.



Sum: 8