**Notes**

* Array MUST be pairwise sorted before algorithm is run
* Double ended selection sort must be run on an array with an even number of elements.
  + If an sorting an odd number of elements we would just add an extra element to the array either Integer.MAX\_VALUE or Integer.MIN\_VALUE
* After every swap we must check A[p] and A[n-p-1] and A[q] with A[n-q-1] to ensure the pairwise sort is maintained.
  + After each swap we know that A[i] and A[n-i-1] are in the proper place, but it may be that the elements at A[p] and A[q] and their corresponding elements may be in violation of the pairwise sort.
* The final step of the process is to check A[n/2-1] against A[n/2] and swap them if necessary.
  + These two elements are on opposite halves of the array and may be out of order once we’ve sorted each half independently.
* Solution to Recurrence relation:

.

* Recall the derivation of time complexity for selection sort.
* The first pass through the array we make comparisons, then on the second pass, etc up until we make 1 final comparison on the last pass…
* Therefore the double ended selection sort makes about half the number of data comparisons than does the traditional selection sort.
* The statistical analysis seems to show this has an effect on the running time.
* However, this algorithm is not better than the O(n logn) sorting algorithms such as merge sort or quicksort.