Another Sorting Algorithm

Dexter keeps doing strange things with numbers.
Yesterday he found a new algorithm to sort a sequence of numbers & he implemented the following pseudocode to sort the list Seq of N numbers (0-based) in ascending order :

```
Seq = Given sequence of N numbers

swap(i,j)
{
   temp = Seq[i]
   Seq[i] =Seq[j]
   Seq[j] = temp
}



reverse(i,j)
{
   Do for k from i to (i+j-1)/2
      swap(k,i+j-k)
}

sort()
{
   Do for i from 0 to N-1
      Do for j from i+1 to N-1
         if(Seq[i]>Seq[j]) then reverse(i,j)

}
```

However unknown to Dexter, his sister Dee Dee added another loop inside the outer loop so that the changed sort function now looks like :

```
sort()
{
   Do for i from 0 to N-1
      Do for j from i+1 to N-1
         if(Seq[i]> Seq[j]) then reverse(i,j)

      Do for j from N-2 to i+2
         reverse(i+1,j)

}
```

When today Dexter tested his program he was frustrated to find that the program was sorting the numbers but it was taking more time than it should(DeeDee's plans always work !).
You have been asked to help esitmate the time taken. Given the sequence of numbers that

Dexter wants to sort, your job is to find the number of times the swap function has been called.

Input Format :

First line contains the number N, the size of the sequence to be sorted. N lines follow, each containing a single integer (the (i)th of these lines contains the value Seq[i])

Output Format:

Output a single number representing the number of times the swap function has been called.

Sample Input File:

5
1
2
3
4
5

Sample Output File:

4

Constraints:
1<=N<=4000
0<=Seq[i]<=1,000,000,000