

1. An *inversion* in an array $A[1..n]$ is a pair of indices (i, j) such that $i < j$ and $A[i] > A[j]$. The number of inversions in an n -element array is between 0 (if the array is sorted) and $\binom{n}{2}$ (if the array is sorted backward).

Describe and analyze an algorithm to count the number of inversions in an n -element array in $O(n \log n)$ time.

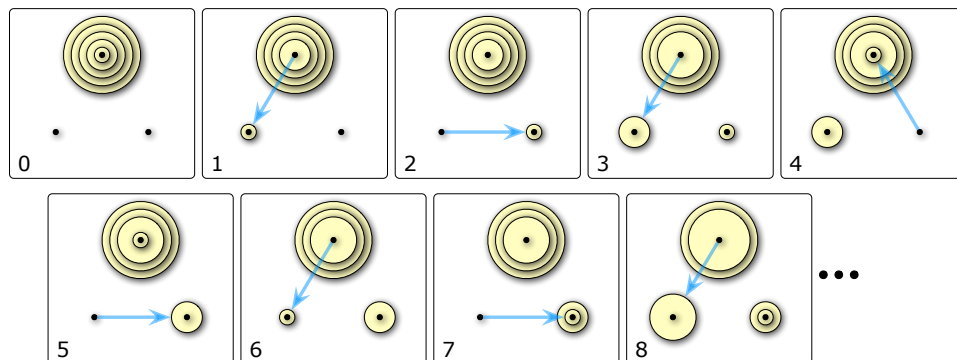
2. (a) Prove that the following algorithm actually sorts its input.

```

STOOGESORT(A[0..n-1]) :
  if n = 2 and A[0] > A[1]
    swap A[0] ↔ A[1]
  else if n > 2
    m = ⌈2n/3⌉
    STOOGESORT(A[0..m-1])
    STOOGESORT(A[n-m..n-1])
    STOOGESORT(A[0..m-1])
    
```

- (b) Would STOOGESORT still sort correctly if we replaced $m = \lceil 2n/3 \rceil$ with $m = \lfloor 2n/3 \rfloor$? Justify your answer.
 - (c) State a recurrence (including base case(s)) for the number of comparisons executed by STOOGESORT.
 - (d) Solve this recurrence. [Hint: Ignore the ceiling.]
 - (e) **To think about on your own:** Prove that the number of *swaps* executed by STOOGESORT is at most $\binom{n}{2}$.
3. Consider the following restricted variants of the Tower of Hanoi puzzle. In each problem, the needles are numbered 0, 1, and 2, and your task is to move a stack of n disks from needle 1 to needle 2.

- (a) Suppose you are forbidden to move any disk directly between needle 1 and needle 2; every move must involve needle 0. Describe an algorithm to solve this version of the puzzle using as few moves as possible. *Exactly* how many moves does your algorithm make?
- (b) Suppose you are only allowed to move disks from needle 0 to needle 2, from needle 2 to needle 1, or from needle 1 to needle 0. Equivalently, Suppose the needles are arranged in a circle and numbered in clockwise order, and you are only allowed to move disks counterclockwise. Describe an algorithm to solve this version of the puzzle using as few moves as possible. *Exactly* how many moves does your algorithm make?



The first eight moves in a counterclockwise Towers of Hanoi solution

- ★(c) Finally, suppose you are forbidden to move any disk directly from needle 1 to 2, but any other move is allowed. Describe an algorithm to solve this version of the puzzle using as few moves as possible. *Exactly* how many moves does your algorithm make?

*[Hint: This version is **considerably** harder than the other two.]*