# Majority Element, Stock Market, and Inversion

Author: Rafael

# 1

If `x` is the majority element in `A`, it must appear in both halves of `A`. If not, `x` would appear ≤ $\frac{n}{2}$ times, contradicting its majority status.

Also, `x` must be the majority in at least one half of `A`. If `x` appears $\frac{n}{2} + 1$ times (minimum for majority), and is evenly split, it could $\frac{n}{4} + \frac{1}{2}$ times in each half. Since we cannot have half an item, that one extra `x` makes it a majority in one half.

This leads to the algorithm in the next page.

1. If array size is one, that element is the majority.
2. Recursively find majority elements in each half.
3. If both halves have the same majority, that is the majority.
4. Otherwise, count `left` and `right` in `A`.
5. If `leftCount` > n/2, `left` is the majority.
6. Else, `right` is the majority, as `A` is guaranteed to have one. In deeper recursion levels, `right` might not be the majority, but this does not matter as the majority element from the "left side" of the recursion will eventually be returned.

```
Procedure MajorityElement(A, n)
        If n == 1
                Return A[0]

        mid = n / 2
        left = MajorityElement(First half of A, n/2)
        right = MajorityElement(Second half of A, n - n/2)

        If left == right
                Return left

        leftCount = Count(A, left) # a function that counts the freq of left
in A
        rightCount = Count(A, right)

        If leftCount > mid
                Return left

        # do not have to worry about a half not containing the
        # majority element because A is said to be guaranteed
        # to have a majority element
        Else
                Return right
```

# 2

1. Initialize `current` to 0 and `longest` to `current`. `current` tracks the current streak and `longest` tracks the longest streak of non-decreasing days
2. Start loop from 0 to `n - 1`
    1. If current price at `day` is less than or equal to `day + 1`, sequence is non-decreasing so increment `current`
    2. Else, set `longest` to the max between `longest` and `current`, reset `current` to 0
3. Return `longest + 1` to account for the last day in the streak

This procedure correctly identifies the length of the longest streak of non-decreasing days because it checks every number in `A`. It will track the length of the current streak, and whenever the streak is broken, it will update `longest` accordingly and reset `current`. At the end, `longest` must contain the length of the longest streak.

```
Procedure StockMarket(A, n)
    Initialize current to 0
    Initialize longest to current
    For day from 0 to n - 1
        If A[day] <= A[day + 1]
            Increment current
        Else
                Set longest to max(longest, current)
            Set current to 0

    Return longest + 1
```

# 3

- Call `Inversions` with `count` starting at 0, this variable will store the number of inversions
- This algorithm is merge sort. The one modification is the counting during the merging of the two halves
- Whenever an item from the second half, `A2`, is inserted into `merged`, increment `count` by however many elements are currently in the first half of `A`, `A1`.
- The variable initially used for `count` now stores the number of inversions

This procedure accurately calculates the number of inversions. During the merge step, if a value from the second half of the array (`A2`) is appended to `merged`, it indicates that this value is larger than the first value in the first half (`A1`). Since `A1` is sorted, all its remaining values are also larger than its first value, and thus larger than the value from `A2` that was appended into `merged`. Moreover, all these values from `A1` had lower indexes in the original array compared to the index of the value from `A2`. Hence, this value from `A2` forms an inversion with all the values currently in `A1`.

Pseudocode available in the next page.

```
Procedure Inversions(A, n, count)
        If n == 1
                Return A

        Initialize A1 to Inversions(First half of A, n/2, count)
        Initialize A2 to Inversions(Second half of A, n - n/2, count)

        Initialize merged to []
        While A1 is not empty and A2 is not empty
                If A1[0] <= A2[0]
                        Append A1[0] to merged
                        Remove first item from A1

                Else
                        Append A2[0] to merged
                        Remove first item from A2
                        Increment count by current length of A1

                Increment appended

        While A1 is not empty
                Append A1[0] to merged
                Remove first item from A1

        While A2 is not empty
                Append A2[0] to merged
                Remove first item from A2

        Return merged
```

#homework