```
Alg separatePosNNeg(A[0…..n-1]) // non_recursive
    for i ⟵ 0 to n-1
        do if (A[i]>0 && A[i+1]<0)
            then tmp=A[i]
                A[i]=A[i+1]
                A[i+1]= tmp
                i=i-2
```

$\longrightarrow$ for loop: $\sum_{i=0}^{n-1} 1$ = n-1-0+1 = n

~~Θ~~(n)

```
Func separatePosNNeg(A, l, r){ // recursive
    if(l==r){
        return l;          ⟶ c/1
    }
    else{
        separatePosNNeg(A, l, floor(l+r)/2) ⟶ T(n/2)
        separatePosNNeg(A, floor(l+r)/2+1, r) ⟶ T(n/2)
        for i ⟵ 0 to n-1
            do if (A[i]>0 && A[i+1]<
                tmp=A[i]
                A[i]=A[i+1]
                A[i+1]= tmp
                i=i-2
    }
```

$\longrightarrow$ for loop: $\sum_{i=0}^{n-1} 1$ = n-1-0+1 = n

}

If_else = max(c,2T(n/2)+n)= 2T(n/2)+n (master method)

$n^{\log_2 2} = n \longrightarrow \Theta(n)$

By comparison:

time complexity of non-recursive algorithm(Θ(n))

= time complexity of recursive algorithm(Θ(n))