Algorithms

@ellasstudy

SELECTION SORT

★ BEST CASE: $O(n^2)$ WORST CASE: $O(n^2)$ ★

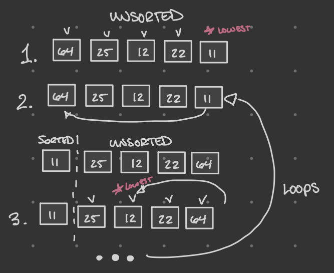
- SIMPLE + SLOW → BEST FOR SMALL
- AN ARRAY IS CONSIDERED INTO 2 PARTS (ALL UNSORTED)

UNSORTED SORTED

SELECTION 1. SELECTS THE LOWEST ELEMENT IN THE REMAINING ARRAY

SWAPPING 2. BRINGS IT TO THE STARTING POSITION

COUNTER SHIFT 3. CHANGE COUNTER FOR UNSORTED ARRAY BY 1

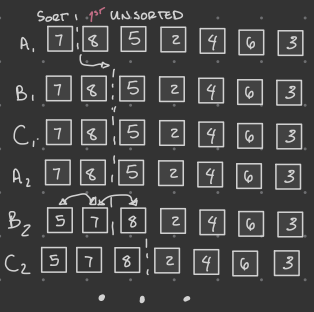


INSERTION SORT

★ BEST CASE: $O(n)$, WORST CASE: $O(n^2)$ ★

- BEST FOR SMALL + ALMOST SORTED
- SORTS ELEMENTS ONE AT A TIME BY COMPARING THE NEXT UNSORTED ELEMENT TO THE SORTED

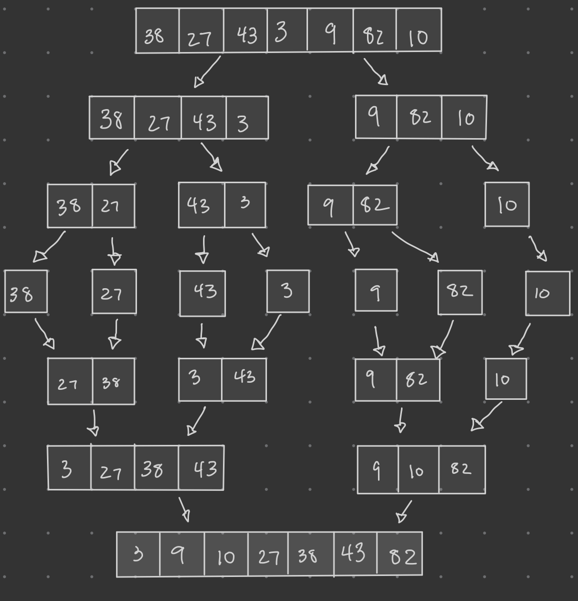
1. SETS A MARKER FOR SORTED SECTION AFTER FIRST ELEMENT
2. REPEAT FOLLOWING UNTIL SORTED
   A. SELECT FIRST UNSORTED
   B. SWAP OTHER ELEMENTS TO THE RIGHT TO CREATE THE CORRECT POSITION + SHIFT UNSORTED ELEMENT.
   C. ADVANCE THE MARKER TO THE RIGHT ONE ELEMENT



MERGE SORT

★ BEST CASE: $O(\log n)$ WORST CASE: $O(\log n)$

- BEST FOR LARGE + UNSORTED
- DIVIDES ARRAY IN HALF THEN SORTS TWO HALFS



BIG O NOTATION

◁ FROM "Big O Notation" VIDEO BY HackerRank ON YouTube

- HOW RUNTIME SCALES

$O(1)$ = SAME TIME REGARDLESS OF INPUT

$O(N)$ = LINEAR

PSEUDO CODE

```
Boolean contains(Array, x){
    for each element in array {
        if element == x {
            return true
        }
    }
}
```

⟹ $O(n)$

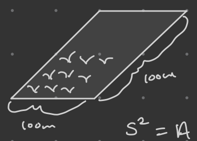
↳ n = SIZE OF ARRAY

PSEUDO CODE

```
Boolean contains(Array){
    for each x in array {
        for each y in array {
            print x, y
        }
    }
}
```

⟹ $O(N^2)$

↳ GOES THROUGH N 2X



NOW IMAGINE A FEILD

2 OPTIONS

$S^2 = A$

$O(A)$ or $O(s^2)$

└ = ┘

LOOKING FOR RELATIONSHIP

MULTISTEP

```
function(){
    doStep1() // O(a)
    doStep2() // O(b)
}
```

$\}$ = $O(a+b)$

- DROP CONSTANTS
- DIFF INPUTS = DIFF VARS
- DROP NON-DOM TERMS