## Array Rotation

## Input Array

0 1 2 3

| 10 | 5 | 27 | 2 |
| --- | --- | --- | --- |

Index

Start index : 0

End Index : 3

### Left Rotation By 1

| 5 | 27 | 2 | 10 |
| --- | --- | --- | --- |

### Left Rotation By 2

| 27 | 2 | 10 | 5 |
| --- | --- | --- | --- |

**{ 339, 19, 276, 780, 301, 711, 943}**

**{ 19, 276, 780, 301, 711, 943, 339}**

**{ 917, 403, 572, 729, 95, 696, 361}**

**{ 403, 572, 729, 95, 696, 361, 917}**

**Left Rotate Array by N times**

|  | 0 | 1 | 2 | 3 | 4 |
| --- | --- | --- | --- | --- | --- |
| Original | 186 | 482 | 336 | 515 | 74 |
| R-1 | 482 | 336 | 515 | 74 | 186 |
| R-2 | 336 | 515 | 74 | 186 | 482 |
| R-3 | 515 | 74 | 186 | 482 | 336 |
| R-4 | 74 | 186 | 482 | 336 | 515 |
| R-5 | 186 | 482 | 336 | 515 | 74 |
| R-6 |  |  |  |  |  |
| R-7 |  |  |  |  |  |
| R-8 |  |  |  |  |  |
| R-9 |  |  |  |  |  |
| R-10 | 186 | 482 | 336 | 515 | 74 |

**Example 1**

* 515 Element
  + Index - 3
  + Rotate left by 2
    - From index 3, shifted to index 1
      * 3 - 2 -> 1

Example 2

* 336 element
  + Index - 2
  + Rotate left by 2
    - Index 2 to index 0
      * 2 - 2 -> 0

Example 3

* 482 element
  + Index - 1
    - Rotate left by 2
      * Index 1 to Index 4
        + 1 - 2 -> -1

-1 + X -> 4

- 1 + 5 -> 4

(5 is the array length)

Example 4

* 186 element
  + Index - 0
    - Rotate left by 2
      * Index 0 to Index 3
        + 0 - 2 -> -2

-2 + X -> 3

-2 + 5 -> 3

(5 is the array length)

## Implementation points

* Single iteration of the whole array
* With the common pattern/formula identified
  + We handle the regular cases, where target index is +ve valid
* A special pattern/formula
  + We will handle the cases where target index value becomes -ve / invalid
* Backup array
  + Copy elements from source array to backup array
  + Retrieval of elements will be from backup array
    - We will be working with source array and making changes

{ 795, 319, 243, 736, 938}

Rotate left by 2

{ 243, 736, 938, 795, 319}

{ 900, 232, 617, 422, 7, 291, 380}

Rotate left by 5

{ 291, 380, 900, 232, 617, 422, 7}

# Binary Search

* Works on a sorted array
  + {243, 319, 736, 795, 938}
    - Length is 5, start index 0, end index 4
    - Search/key element
      * 736, 795, 319
* Identify the midpoint in the sorted array
  + Mid point index -> (beginIndex + endIndex) / 2
    - 0 + 4 / 2 -> 2
    - 2 is the mid point index
      * Get element at mid point index
* If element searched for, is at the mid point index
  + Return from code execution as we have found our element
* If element searched for, is not at the mid point index
  + Start dividing the array
  + Divide the array into sub sections or smaller portions
    - 100 elements
      * Range 0 to 99
        + Splitting into smaller index ranges

0 - 24, 25 - 49 and so on

* + In the divided segments of array,
    - We will search for our preferred element - key element

Case 1

Search element is 736

* Mid point element -> 736
  + Matches with key element
  + Return

Case 2

Search element is 795

* Mid point element -> 736
  + {243, 319, **736**, 795, 938}
    - On the right side of the midpoint
      * Start splitting the array into index ranges
      * {795,938}
        + {795}, {938}

Case 3

Search element is 319

* Mid point element -> 736
  + {243, 319, **736**, 795, 938}
    - Start splitting the array on the left side of the midpoint
      * {243, 319}
        + {243}, {319}

Test Cases:

Source Array

{75, 498, 1136, 1818, 1977, **2453**, 2486, 2898, 3049, 3113, 3870}

Search elements

{75, 3113, 498, 2453, 2898}

Element 75 present at index 0

--------------------

Element 3113 present at index 9

{2486, 2898, **3049**, 3113, 3870}

{3113, 3870}

--------------------

Element 498 present at index 1

--------------------

Element 2453 present at index 5

--------------------

Element 2898 present at index 7

# Merge Sort

* Visualization on hacker earth site
  + <https://www.hackerearth.com/practice/algorithms/sorting/merge-sort/visualize/>