<배열 (Array)>

- **Go에서 사이즈 구하기**

```
arr := []int{1, 2, 3, 4, 5, 6, 7}
n := len(arr)
```

1. 배열 회전 프로그램

배열을 2씩 회전하면서 배열된다.

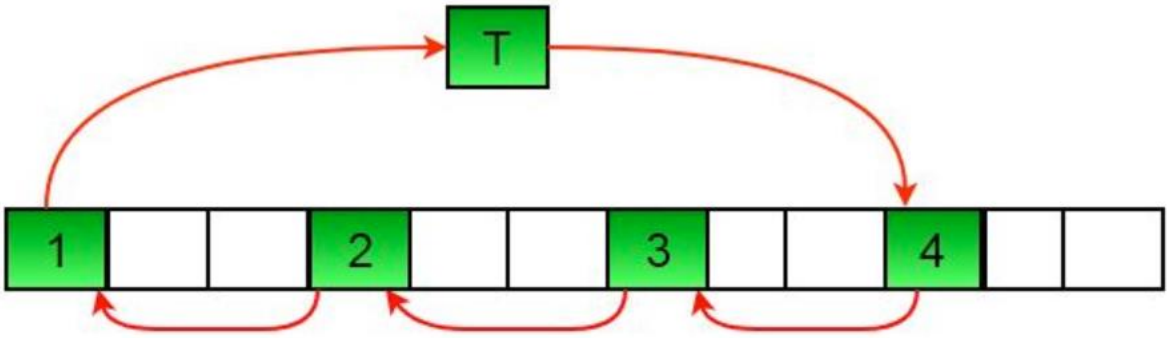
- **기본적인 회전 알고리즘 구현**
- **예제 코드:** https://github.com/limes22/algorithm/blob/main/go/DataStructure/Array/revers_array.go

Temp를 활용해서 첫번째 인덱스 값을 저장 후 arr[0]~arr[n-1]을 각각 arr[1]~arr[n]의 값을 주고, arr[n]에 temp를 넣어준다.

```go
func leftRotatebyOne(arr []int, n int) {
    i := 0
    temp := arr[i]
    for i = 0; i < n-1; i++ {
        arr[i] = arr[i+1]
    }
    arr[i] = temp
}
```

이 함수를 이용해 원하는 회전 수 만큼 for문을 돌려 구현 가능

- **저글링 알고리즘 구현**
- **예제 코드**: https://github.com/limes22/algorithm/blob/main/go/DataStructure/Array/juggling_array.go



최대공약수 gcd 를 이용해 집합을 나누어 여러 요소를 한꺼번에 이동시키는 것

- **역전 알고리즘 구현**
- **예제 코드:** https://github.com/limes22/algorithm/blob/main/go/DataStructure/Array/reversal_array.go

회전시키는 수에 대해 구간을 나누어 reverse 로 구현하는 방법

d = 2 이면, 1,2 / 3,4,5,6,7 로 구간을 나눈다.

첫번째 구간 reverse -> 2,1 , 두번째 구간 reverse -> 7,6,5,4,3

합치기 -> 2,1,7,6,5,4,3 , 합친 배열을 reverse -> **3,4,5,6,7,1,2**

- **Swap을 통한 reverse**

```go
func reverseArr(arr []int, start int, end int) {
    for start < end {
        temp := arr[start]
        arr[start] = arr[end]
        arr[end] = temp
        start++
        end--
    }
}
```

- **구간을 d로 나누었을 때 역전 알고리즘 구현**

```go
//d로 나눠서 역전 알고리즘 수행
func rotateLeft(arr []int, d int, n int) {
    reverseArr(arr, start: 0, d-1)
    reverseArr(arr, d, n-1)
    reverseArr(arr, start: 0, n-1)
}
```

2. 배열의 특정 최대 합 구하기

- **예제 코드:**
  https://github.com/limes22/algorithm/blob/main/go/DataStructure/Array/maxvalue_array.go

예시) arr[i]가 있을 때, i*arr[i]의 Sum 이 가장 클 때 그 값을 출력하기

(회전하면서 최대값을 찾아야한다.)

```
Input: arr[] = {1, 20, 2, 10}
Output: 72

2번 회전했을 때 아래와 같이 최대값이 나오게 된다.
{2, 10, 1, 20}
20*3 + 1*2 + 10*1 + 2*0 = 72

Input: arr[] = {10, 1, 2, 3, 4, 5, 6, 7, 8, 9};
Output: 330

9번 회전했을 때 아래와 같이 최대값이 나오게 된다.
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
0*1 + 1*2 + 2*3 ... 9*10 = 330
```

- **접근 방법**

arr[i]의 전체 합과 i*arr[i]의 전체 합을 저장할 변수 선언

최종 가장 큰 sum 값을 저장할 변수 선언

배열을 회전시키면서 i*arr[i]의 합의 값을 저장하고, 가장 큰 값을 저장해서 출력하면 된다.

- **해결법**

```go
func MaxVal(arr []int, n int) int {
    arrSum := 0 //arr[i]의 전체값
    curSum := 0 //i*arr[i]의 전체값
    for i := 0; i < n; i++ {
        arrSum = arrSum + arr[i]
        curSum = curSum + (i * arr[i])
    }
    maxSum := curSum
    for j := 1; j < n; j++ {
        curSum = curSum + arrSum - n*arr[n-j]
        if curSum > maxSum {
            maxSum = curSum
        }
    }
    return maxSum
}
```

<Reference>

1. https://github.com/gyoogle/tech-interview-for-developer/blob/master/Computer%20Science/Data%20Structure/Array.md