Algorithm

Algorithm

마방진 (Magic Square)

- 홀수만 처리 가능!

```
int size = 5;

int[][] arr = new int[size][size];
int middle = size/2;

int i=0, j=middle;
for(int num=1 ; num<=size*size ; num++) {
    arr[i][j] = num;
  //행 감소
    i--;
    if(i < 0)
        i = size-1;
  //열 증가
    j = (++j)%size;
  //아래 표현과 같다.
    /*j++;
    if(j >= size)
        j = 0;*/

   //배수이면 행은 1 증가, 열은 그대로
    if(num%size == 0) {
        i = (i+2)%size;
        j--;
        if(j < 0)
            j = size-1;
    }
}
```

Sort

합병 정렬 (Merge Sort)

참조

Radix Sort (기수 정렬)

참조

Search

이진 탐색 (Binary Search)

- 일반적인 구현

```
    public static int binarySearch(int[] arr, int target) {
        int left = 0;
        int right = arr.length - 1;
        int mid = 0;
        while(left <= right) {
            mid = (left + right) / 2;
            if(arr[mid] < target)
                left = mid + 1;
            else if (arr[mid] > target)
                right = mid - 1;
            else
                return mid;
        }
        return Integer.MIN_VALUE;
    }
```

- 재귀적 구현

```
    public static int binarySearchRecursive(int[] arr, int target, int left,
int right) {
        if(left > right)
            return Integer.MIN_VALUE;
        int mid = (left + right) / 2;
        if(arr[mid] < target)
            return binarySearchRecursive(arr, target, left + 1, right);
        else if (arr[mid] > target)
            return binarySearchRecursive(arr, target, left, right - 1);
        else
            return mid;
    }
```

- A □ X ≤ B 조건을 검색. (반환값이 index+1 이므로)

```
    public static int binarySearch(ArrayList<Integer> list, int target) {

        int left = 0;
        int right = list.size() - 1;
        int middle = 0;
        while(left <= right) {
```

```
            middle = (left + right) / 2;
            if(target < list.get(middle)) {
                right = middle - 1;
            }
            else {
                left = middle + 1;
            }

            if(list.get(middle) <= target && middle < list.size() - 1) {
                if(target < list.get(middle + 1)) {
                    return list.get(list.size() > middle + 1 ? middle + 1 :
list.size() - 1);
                }
            }
        }
        return -1;
    }
```

From:
https://ledyx.xyz/ - **Ledyx WIKI**

Permanent link:
https://ledyx.xyz/doku.php?id=algorithm:algorithm&rev=1465305120

Last update: **2026/10/24 07:03**