**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;
    }
}
```

**Recursion**

참조

**Sort**

**합병 정렬 (Merge Sort)**

참조

**Radix Sort (기수 정렬)**

참조

**Search**

**Binary Search (이진 탐색)**

참조

**Dynamic Programming**

```
    private TreeSet<File> getSourceFiles(File srcDir) throws
FileNotFoundException {
        if (!srcDir.isFile() && srcDir.listFiles() == null)
            throw new FileNotFoundException("Not found source directory.");

   // 파일 이름순 정렬
        TreeSet<File> files = new TreeSet<File>(new Comparator<File>() {
            public int compare(File o1, File o2) {
                return o1.getPath().compareToIgnoreCase(o2.getPath());
            }
        });

        Stack<File> directories = new Stack<File>();
        directories.push(srcDir);

        // DFS
        while(!directories.isEmpty()) {
            File directory = directories.pop();

            for(File composite : directory.listFiles()) {
                if(composite.isDirectory()) {
                    directories.push(composite);
                }
                else if(composite.isFile()) {
                    files.add(composite);
                }
            }
        }

        return files;
    }
```