2022 여름학기 동국대학교 SW역량강화캠프

12일차. 정렬과 분할정복

동국대학교 dongguk university

FUTURE TECH ACADEMY

● 정렬

▶ 일정 기준에 따라서 원소들을 순서대로 나열하는 과정

▶ 버블 정렬, 선택 정렬, 삽입 정렬, 힙 정렬, 병합 정렬, 퀵 정렬, 계수 정렬, 기수 정렬

▶ Java에서는 Array.sort() 나 Collection.sort() 와 같은 정렬 알고리즘이 내장

▶ 버블 정렬, 삽입 정렬, 병합 정렬

● 버블 정렬

▶ 인접한 두 원소끼리 비교하여 정렬하는 알고리즘

▶ 한 번 선형 탐색 할 때마다, 마지막 원소를 고정시킬 수 있다.

▶ (N−1)번 선형 탐색하여 정렬

▶ 시간복잡도 $O(N^2)$

- 버블 정렬

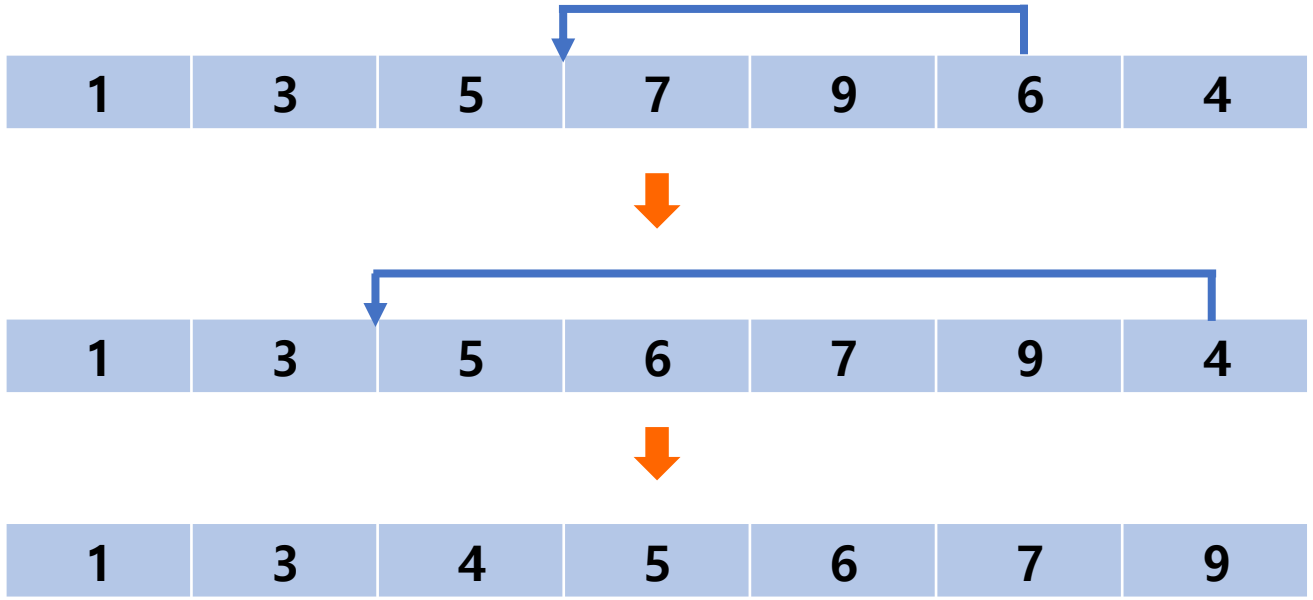
6 5 3 1 8 7 2 4

버블 정렬 코드 O(N²)

```
for(int i=0; i < N-1; i++){
    for(int j=1; j<N-i; j++){
        if(arr[j] < arr[j-1]) {
            int tmp = arr[j];
            arr[j] = arr[j-1];
            arr[j-1] = tmp;
        }
    }
}
```

● 삽입 정렬

▶ 원소를 정렬된 배열에서 위치를 찾아 삽입하여 정렬하는 알고리즘



▶ 시간복잡도 $O(N^2)$

● 삽입 정렬

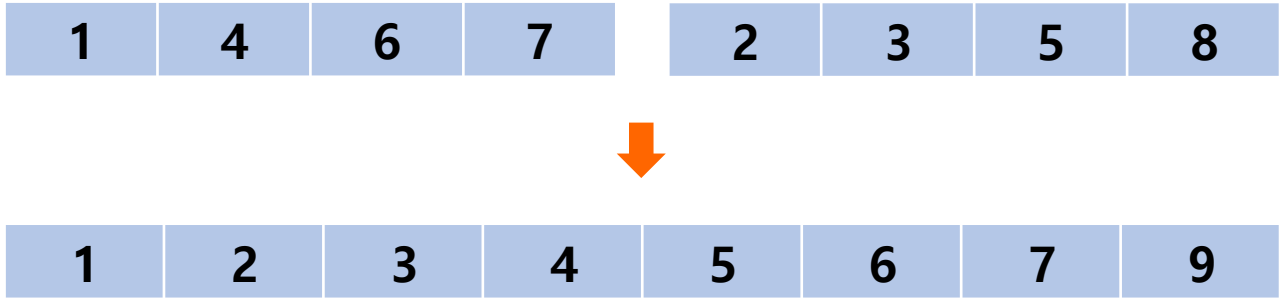
6 5 3 1 8 7 2 4

삽입 정렬 코드 $O(N^2)$

```
for(int i=1; i<N; i++) {
    int target = arr[i]; // arr[i]의 위치 정하기
    int j = i; // 초기 위치 j
    while(j>=1 && target < arr[j-1]) { // j의 한 칸 왼쪽이 target보다 크다면
        arr[j] = arr[j-1]; // arr[j-1]을 오른쪽으로 한 칸 옮기고
        j--; // target의 위치를 한 칸 왼쪽으로
    }
    arr[j] = target;
}
```

● 병합 정렬

▶ 배열을 2등분하여 각각 정렬하고, 정렬된 두 배열을 병합하는 것을 재귀적으로 진행하여 배열을 정렬하는 알고리즘



● 병합 정렬

6 5 3 1 8 7 2 4

● 병합 정렬

▶ 구간의 길이가 8이라면 병합 Layer = 3 (1 → 2 → 4 → 8)

▶ 구간의 길이가 16이라면 병합 Layer = 4 (1 → 2 → 4 → 8 → 16)

▶ 시간복잡도 = O (Layer 수 * N) = O(NlgN)

▶ 컴퓨터 과학에서 lgN = $\log_2 N$

병합 정렬 코드 O(NlgN)

```java
public static void mergesort(int start, int end) {
    if (start == end)
        return;
    int mid = (start + end) / 2;
    mergesort(start, mid); // 왼쪽 절반 정렬
    mergesort(mid + 1, end); // 오른쪽 절반 정렬
    //임시 배열 tmp에 arr[start~end]을 정렬한 결과를 저장
    int L = start; // 왼쪽 분할의 첫 인덱스
    int R = mid + 1; // 오른쪽 분할의 첫 인덱스
    for (int i = start; i <= end; i++) {
        if (L > mid) // 왼쪽 분할에 남은 원소가 없는 경우
            tmp[i] = arr[R++];
        else if (R > end) // 오른쪽 분할에 남은 원소가 없는 경우
            tmp[i] = arr[L++];
        else if (arr[L] < arr[R]) // 왼쪽 분할의 원소가 더 작은 경우
            tmp[i] = arr[L++];
        else // 오른쪽 분할의 원소가 더 작은 경우
            tmp[i] = arr[R++];
    }
    for (int i = start; i <= end; i++) { // 임시로 저장해둔 tmp의 값들을 arr로 복사
        arr[i] = tmp[i];
    }
}
```

● 하노이의 탑(5342)

세 개의 장대가 있고 첫 번째 장대에는 반경이 서로 다른 n개의 원판이 쌓여 있다. 각 원판은 반경이 큰 순서대로 쌓여있다. 이제 수도승들이 다음 규칙에 따라 첫 번째 장대에서 세 번째 장대로 옮기려 한다.

한 번에 한 개의 원판만을 다른 탑으로 옮길 수 있다.
쌓아 놓은 원판은 항상 위의 것이 아래의 것보다 작아야 한다.
이 작업을 수행하는데 필요한 이동 순서를 출력하는 프로그램을 작성하라. 단, 이동 횟수는 최소가 되어야 한다.

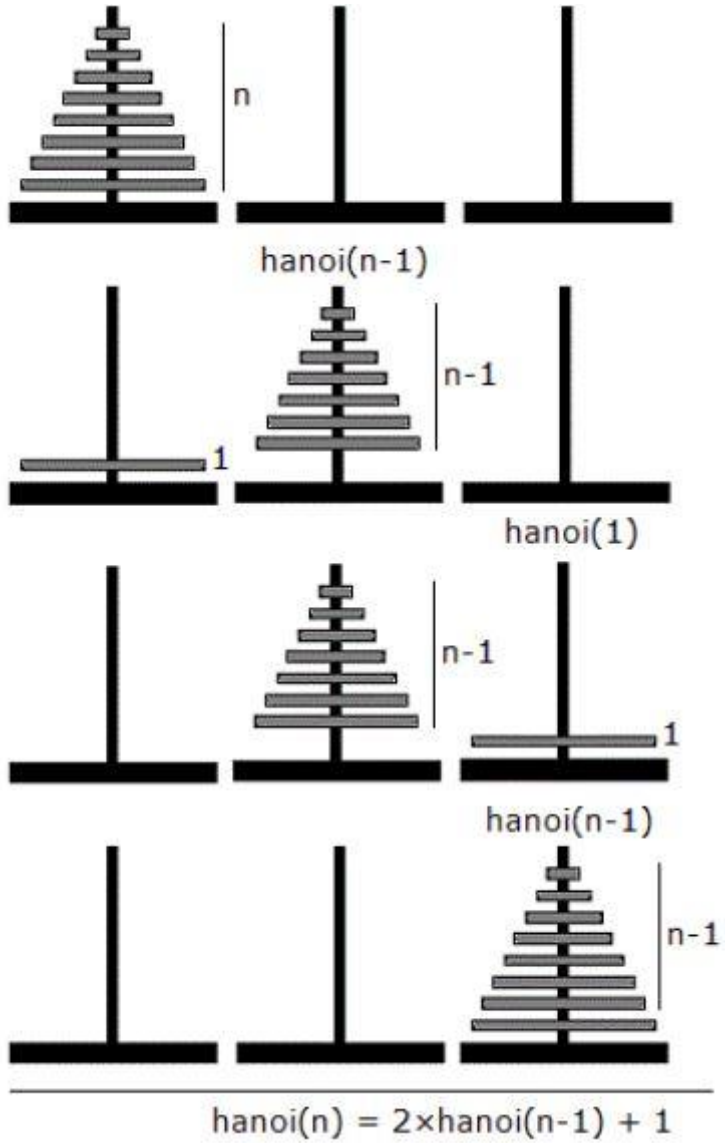
▶ 분할 정복을 이용한 풀이

▶ 크기 N인 원판을 1번에서 3번으로 옮기기 위해서는, N번보다 작은 원판들은 전부 2번에 있어야만 한다.

hanoi(from, to, N) 은

hanoi(from, other, N-1)
move(from, to, N)
Hanoi(other, to, N-1)

의 세 단계로 나누어 생각할 수 있다.



핵심 코드

```
public static void func(int from, int to, int lvl) { // from에서 to로 1~lvl 크기의 원판들을 옮겨라
    if(lvl == 1) {cnt++; sb.append(from).append(" ").append(to).append('\n'); return ;}
    int other = 6-from-to;
    func(from, other, lvl-1);
    sb.append(from).append(" ").append(to).append('\n'); cnt++;
    func(other, to, lvl-1);
}
```

● 분할정복으로 별 찍기(5341)

재귀적인 패턴으로 별을 찍어 보자. N이 3의 거듭제곱(3, 9, 27, ...)이라고 할 때, 크기 N의 패턴은 N×N 정사각형 모양이다.

크기 3의 패턴은 가운데에 공백이 있고, 가운데를 제외한 모든 칸에 별이 하나씩 있는 패턴이다.

```
***
* *
***
```

N이 3보다 클 경우, 크기 N의 패턴은 공백으로 채워진 가운데의 $(N/3)\times(N/3)$ 정사각형을 크기 $N/3$의 패턴으로 둘러싼 형태이다. 예를 들어 크기 27의 패턴은 예제 출력 1과 같다.

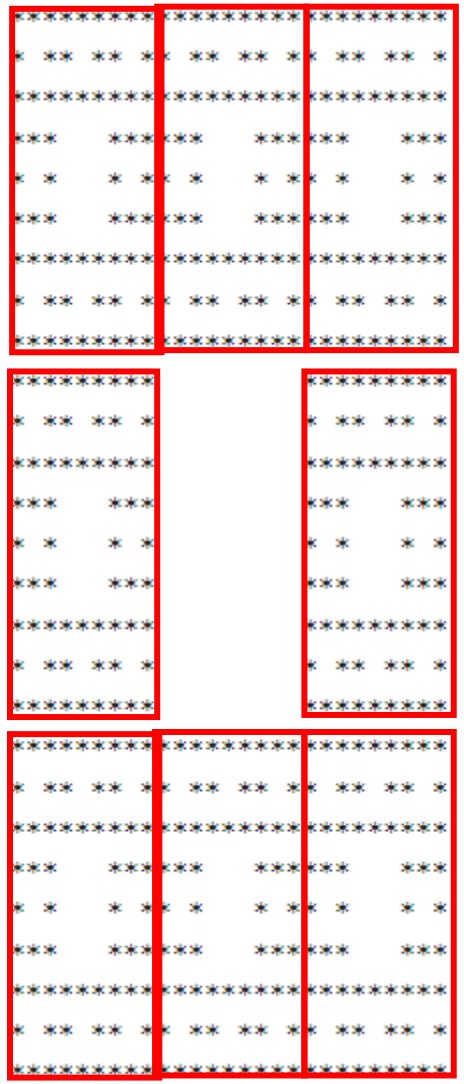
▶ 분할 정복을 이용한 풀이

▶ 크기 N의 패턴은 8개의 크기 (N-1)의 패턴들로 구성

star(x, y, len)= (x,y)부터 길이 len의 정사각형을 패턴으로 채우는 함수

star(x+(len/3)*i , y+(len/3)*j, len/3) (i!=1 || j!=1)

의 총 8개의 작은 부분으로 나누어 생각할 수 있다.



핵심 코드

```
public static void func(int x, int y, int len) {
    if(len == 1) {arr[x][y] = true; return ;}
    int slen = len/3;
    for(int i=0; i<3; i++) for(int j=0; j<3; j++)
    {
        if(i==1 && j==1) continue;
        func(x + slen*i, y + slen*j, slen);
    }
}
```