


# 240719(금) 삼성 SW 역량테스트 B형 돌파 전략

## 삼성 SW 역량테스트 B형의 출제 의도

B형에서 묻고자 하는 것은 다음과 같습니다. **상당한 수준의 최적화**가 가능한가? 그리고 그 **최적화를 위한 적절한 자료 구조**를 알고 있는가? 여기서 말하는 상당한 수준의 최적화라는 것은 Optimal 정도의 최선의 답은 아니지만 최소한 Sub-Optimal 정도의 최적화를 말합니다. 문제에 의도한 정확한 자료구조를 선택해야 풀리게 되고, 어설픈 greedy한 해법이 먹히지 않는 것이 B형입니다.

문제를 받아 들게 되면 키보드부터 두드리기보다 주어진 문제를 읽고 구현해야 하는 메소드들을 분석해서 어떤 자료 구조를 사용해야 하는가부터 도출해야 합니다. 여러 개의 메소드들이 있기 때문에 그 메소드들을 전부 커버할 수 있는 자료 구조가 무엇일까를 떠올려보세요.

 B형은 생각보다 쉽습니다. 사람에 따라서는 복잡한 시뮬레이션을 구현해야 하는 A형 문제들보다 쉽다고 느낄 수도 있죠. 그리고 개별 메소드들은 실제로 구현하는데 채 10분도 걸리지 않을 수 있습니다.

## B형의 문제 구성

B형 문제는 Solution이라는 이름을 가진 파일 내에서 4시간 동안 4~6개의 메소드를 구현하는 것으로 구성되어 있습니다. 메소드의 명칭은 이미 정해져 있으며, 채점기에서 그 메소드를 직접 호출해줍니다. 따라서 다른 코딩 테스트와는 다르게 입출력의 형태로 채점이 이루어지지 않는다는 것을 기억해야 합니다. 이 점에 대해서 인지하고 있지 않은 상태에서 처음으로 문제를 풀게 되면 크게 당황할 수 있죠. (이 부분에 대해서는 실제 시험의 형태는 변경될 수도 있으니 참고만 해주세요)

개별 메소드들은 input과 return의 형태가 정해져 있습니다. 각 메소드들은 테스트 케이스 내에서 채점기가 무작위로 호출하기 때문에 요구사항 대로 정확하게 구현하지 않으면 오답을 내뱉을 수 있습니다. 또한 개별 메소드들의 기본 형태는 아래와 같이 주어집니다.

Python

복사

```
def swap_top(mSrc, mDst):  
    return
```

```
def move(mSrc, mDst):  
    return 0
```

...

I

각 메소드들이 input과 return의 형태가 정해져 있다는 것에 유의하세요. return의 경우에는 값을 리턴하거나 상태에 대한 코드를 리턴하는 경우도 있습니다. '목표한 것을 찾지 못하면 0을 리턴하라' 이런 것처럼요.

많은 경우에 몸풀기로 init 함수가 출제되기 때문에 메소드가 너무 많다고 해서 당황하지 않아도 됩니다. 사실 3~5개 정도의 메소드를 4시간 안에 풀어내면 되는 것이니까요.

## 상태 관리의 중요성

채점기는 각 메소드를 계속해서 호출합니다. 각 호출로 인해서 변경된 상태는 적절한 자료구조로 되어 있는 **전역 객체**에 보관해야 합니다.

I

```
1. swap_top(1,4)  
2. move(2,3)  
3. move(3,4)  
...
```

예를 들어서 위와 같이 메소드 호출이 연속적으로 이뤄진다고 생각해봅시다. 각 호출의 결과로 데이터의 상태는 계속 변화하게 될 것입니다.

여러분들은 각 데이터의 상태 변화를 지속적으로 관리하고 반영해야 합니다.

문제를 풀기 전에 각 데이터의 상태 변화를 효율적으로 반영할 수 있는 자료 구조가 무엇일까를 생각해 볼 필요가 있습니다.

특정 메소드에는 사용하기 적합한 자료구조라도 모든 메소드에 사용하기 적합하지는 않을 수 있기 때문입니다.

## 어떤 자료 구조들을 알고 있어야 하는가?

사용하는 언어가 Java라면 기본적으로 **collection**에 속하는 자료 구조들은 모두 알고 있어야 합니다. 또한 단순히 알고 있다가 아니라 세세하게 다뤄봤어야만 하죠. 단순히 알고 있다가의 상태로 는 절대 B형을 풀어낼 수 없습니다.


예를 들어서 우선순위 큐(Priority Queue)를 구현한다고 했을 때 Comparable을 implements 해야 하고 compareTo 라는 메소드를 Override 해야 한다는 사실을 모른다면 절대 B형 문제를 풀 어낼 수 없습니다. 또한 구현을 실제로 해본 경험이 있어야 합니다.

```
static PriorityQueue<Point> pq;  
pq = new PriorityQueue<>();
```

Java

복사

```
class Point implements Comparable<Point> {  
    int i, j;  
    int dist = 9999;  
  
    Point(int i, int j){  
        this.i=i;  
        this.j=j;  
    }  
  
    @Override  
    public int compareTo(Point targetP) {  
        return this.dist <= targetP.dist ? -1: 1;  
    }  
}
```

 위의 PriorityQueue처럼 실제로 선언하고 구현하고 Override 해보면서 자료구조를 A부터 Z까지 이해하고 있어야 합니다. PriorityQueue 같은 경우에도 위와 같은 단순한 형태의 비교 조건도 있지만 복잡한 경우의 비교 조건도 필요할 수 있으니까요.

자주 출제되는 자료 구조들은 LinkedList, PriorityQueue, HashMap, Set, TreeSet 등이 있습니다. 기본적으로 Java의 collection에 해당하는 자료 구조들은 모두 숙지가 필요하고 그 외에도 트라이와 같은 자료 구조도 익혀 둡시다.

## 공채 코딩 테스트 vs B형

공채 코딩 테스트와 B형을 비교하자면 공채 코딩 테스트에 등장하는 B형과 같은 유형의 문제가 약간 더 쉽습니다.

너무 당연한 이야기지만 같은 4시간이 주어지기 때문에 2문제를 풀어야 하는 공채 코딩 테스트와 B형 사이에는 난이도 차이가 있을 수밖에 없습니다.

또한 공채 코딩 테스트에서는 B형과 같은 유형의 문제를 해당 코딩 테스트에서 처음 접하는 경우가 많기 때문에 무작정 어렵게 내기 어렵습니다.

따라서 B형을 광탈하셨더라도 용기를 잃지 말고 공채 코테에는 도전해보시길 권해드리고요.

저는 개인적으로 B형 문제가 자료 구조에 대한 이해만 선행된다면 A형 문제보다도 쉽다고 생각합니다.

가끔 다익스트라 같은 초심자에게 조금 어려운 알고리즘을 알고 있어야 풀 수 있는 문제도 출제되긴 하지만, 대부분의 경우에는 특정 알고리즘을 알아야 한다기보다 자료 구조에 대해서 잘 이해하고 있는 지만 물어보거든요.

생각보다 쉬우니까 꼭 도전하고 격파하시길...

코드트리 가서 삼성 기출 문제 풀어보자!

ex) 산타의 선물 공장 2