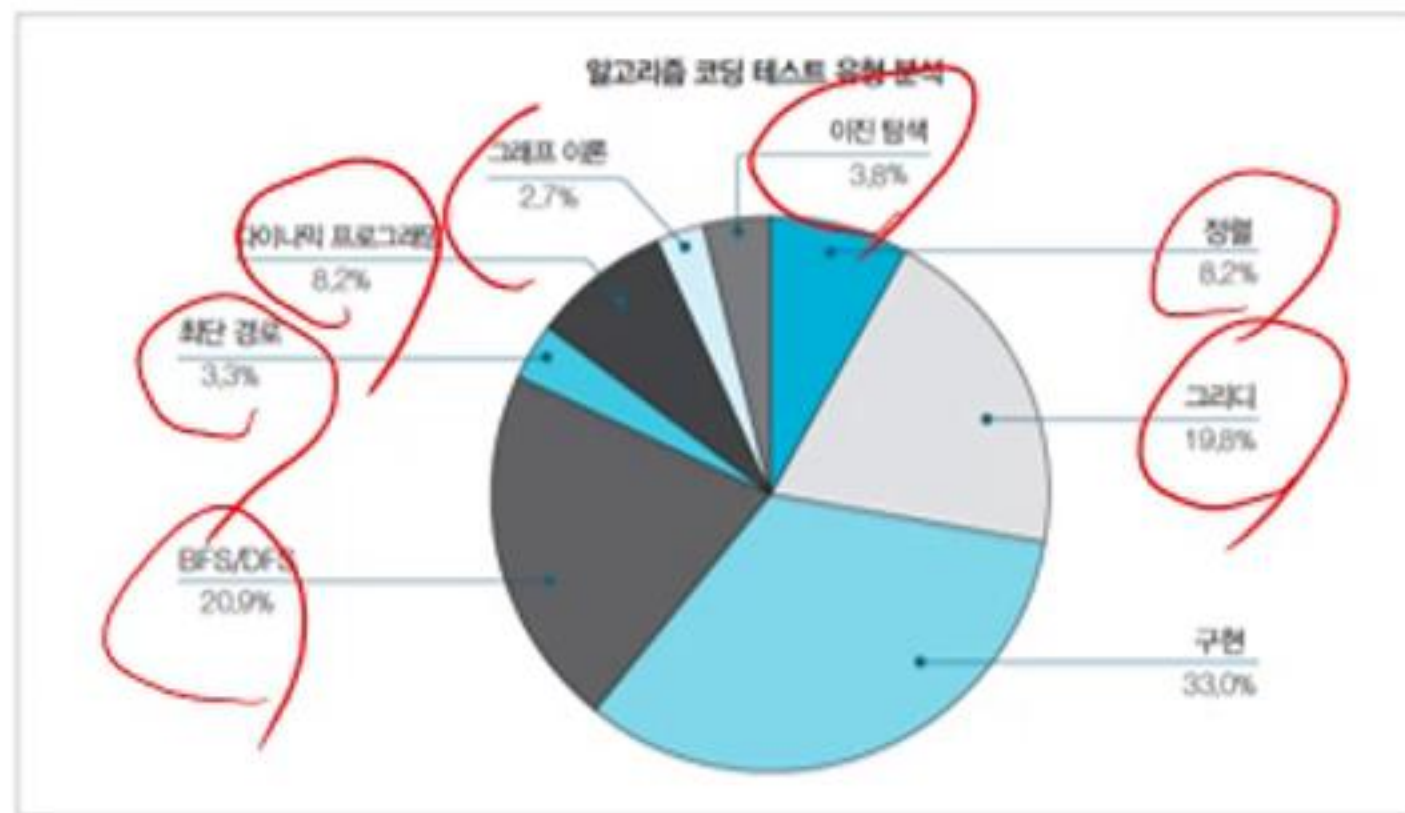


# Problem Statements

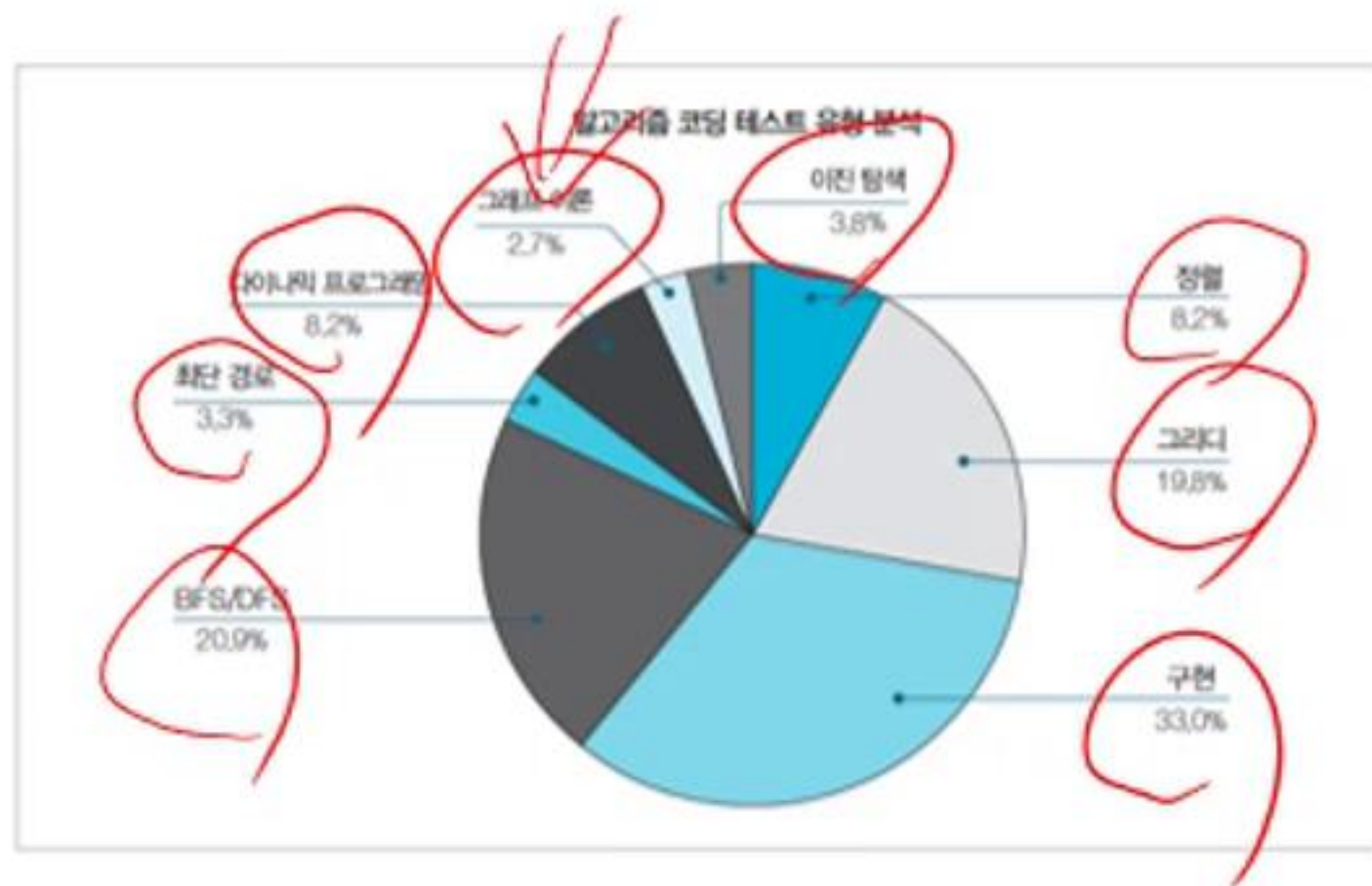
- 자료구조와 알고리즘에 대한 배경 지식
- 자료구조에 대한 라이브러리 활용 이해
- 고급 알고리즘 문제 해결 방법 이해 및 제시
- 제시된 문제에 대한 이해 및 풀이 방법 정리

# 알고리즘 코딩 테스트 유형



Source : 이것이 취업을 위한 코딩 테스트다 with 파이썬, 한빛미디어

# 알고리즘 코딩 테스트 유형



Source : 이것이 취업을 위한 코딩 테스트다 with 파이썬, 한빛미디어

# Category for Coding Problems

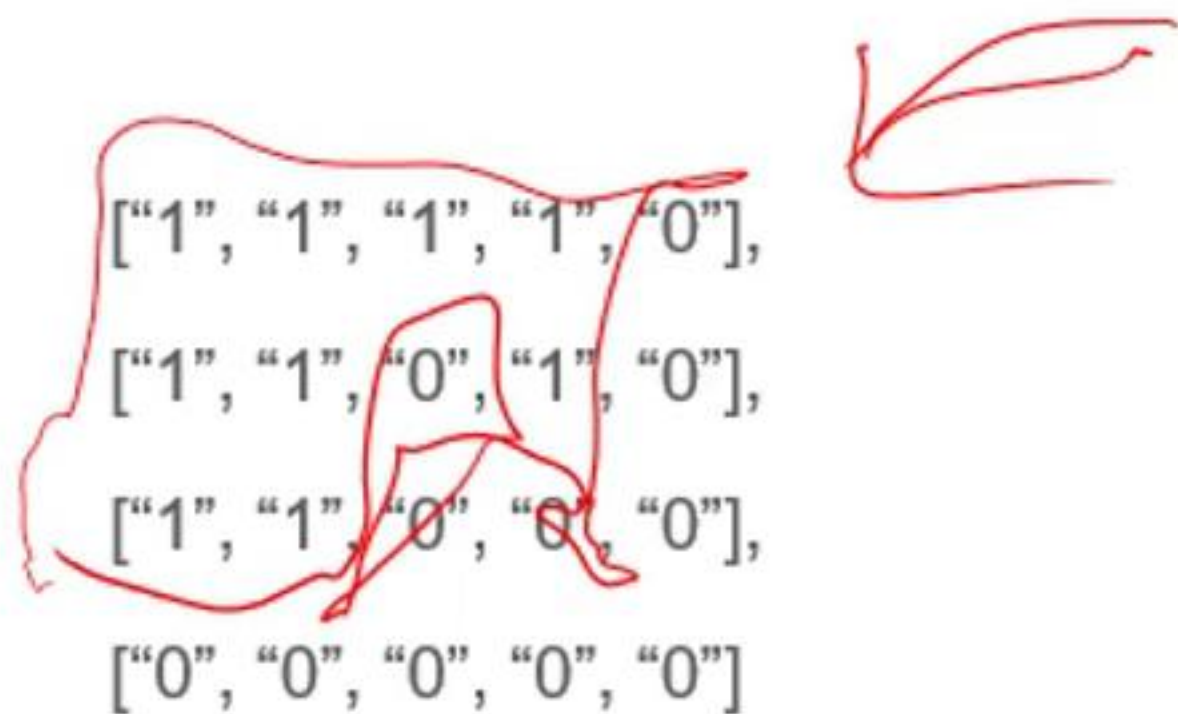
## Popular types for Problems

<u>Data Structures</u>	<u>Algorithms</u>	<u>Advanced Algorithms</u>	<u>Approaches</u>
Array	Depth-First Search	Dynamic Programming	Backtracking
String	Breadth-First Search	Greedy	Bit Manipulation
Hash Table+Hash Function	Sorting	Math	Two Pointers
Tree+Binary Tree	Graph Theory	Game Theory	Divide and Conquer
Binary Search & BST			
Matrix			
Stack+Queue			
Heap (Priority Queue)			

Based on statistics from leetcode

## Level 2 - 섬의 갯수

[DFS]



```
[ "1", "1", "1", "1", "0"],  
[ "1", "1", "0", "1", "0"],  
[ "1", "1", "0", "0", "0"],  
[ "0", "0", "0", "0", "0"]
```

이러한 배열이 있을때 1을 제외 하고는 모두 바다입니다.

이때 섬의 개수를 찾으십시오.(단 괄호는 모두 바다로 생각해도 좋습니다.)

위 예제는 섬이 하나입니다.



해결 방법 : 시작 지점을 정하고 DFS와 BFS 탐색  
알고리즘을 선택

1	1	1
0	1	0
1	0	0
1	0	1

1	1	1
0	1	0
1	0	0
1	0	1

(0,0) 부터 (높이(h), 너비(w))까지 DFS / BFS 탐색을 통해 탐색을 진행한다.

섬은 대각선으로도 연결되어 있으므로 8개 방향을 모두 체크하여 땅(1)이라면 방문한다.

## Level 2 - 일정간격의 병합

[Sorting]

- 입력 :  $[[1, 3], [2, 6], [8, 10], [15, 18]]$

- 출력 :  $[[1, 6], [8, 10], [15, 18]]$

- 위와 같이 아무 규칙도 없는 간격의 모음 들이 입력 되면 겹치는 간격을 병합합니다.

- 입력 :  $[[1, 4], [4, 5]]$       출력 :  $[[1, 5]]$

- 입력 :  $[[1, 9], [2, 6], [16, 25]]$

- 출력 :  $[[1, 9], [16, 26]]$



## 1. 모든 간격을 앞에서부터 탐색

[[1, 4], [4, 5], [10, 26]] 를 입력 받았다면 [1, 4]에 두개의 범위가 포함되는지 일일이 비교하면 답은 찾을 수 있습니다.

하지만  $O(n^2)$ 이 되어버려 시간이 짧은 문제의 경우 풀 수 없을 것입니다.

이를 해결하기 위해선 입력 받는 모듈, 출력 모듈, 비교 모듈이 필요합니다.



## 2. 정렬을 통해 해결

시작 값을 기준으로 간격을 정렬하면 병합 할 수 있는 각 간격 집합이 정렬된 목록에서 연속 실행합니다.

$$Arr[i].end < Arr[j].start \leq Arr[j].end \leq Arr[k].start$$

$$Arr[i].end \geq Arr[k].start$$

위 식은 간격안에 들지 못하는 조건을 식으로 나타낸 것입니다.

[(1, 9), (2, 5), (19, 20), (10, 11), (12, 20), (0, 3), (0, 1), (0, 2)]

[(0, 3), (0, 1), (0, 2), (1, 9), (2, 5), (10, 11), (12, 20), (19, 20)]

정렬 후 위 범위에 포함되지 않는 조건식을 적용하면 비교하지 않아도 되는 값들이 뒤로 밀리며 포함되지 않는 순간부터 해당 인덱스에 대한 연산을 멈춥니다.

## 2. 정렬을 통해 해결

시작 값을 기준으로 간격을 정렬하면 병합 할 수 있는 각 간격 집합이 정렬된 목록에서 연속 실행합니다.

$$Arr[i].end < Arr[j].start \leq Arr[j].end \leq Arr[k].start$$

$$Arr[i].end \geq Arr[k].start$$

위 식은 간격안에 들지 못하는 조건을 식으로 나타낸 것입니다.

↓  
[(1, 9), (2, 5), (19, 20), (10, 11), (12, 20), (0, 3), (0, 1), (0, 2)]

⇒ [(0, 3), (0, 1), (0, 2), (1, 9), (2, 5), (10, 11), (12, 20), (19, 20)]

정렬 후 위 범위에 포함되지 않는 조건식을 적용하면 비교하지 않아도 되는 값들이 뒤로 밀리며 포함되지 않는 순간부터 해당 인덱스에 대한 연산을 멈추면 됩니다.

시간 복잡도:  $O(n \log n)$  +  $n \Rightarrow O(n \log n)$

Arr or list(등등) = > intervals.sort **앞 자리 기준으로 정렬해줍니다.**

(링크로 구성할 때의 예시:intervals.sort(key = lambda x: x[0])

Merged = []. **새로 정렬할 리스트**

For interval in intervals:

if not Merged or Merged[-1][1] < interval[0]:

merged.append(interval)

$Arr[i].end < Arr[j].start \leq Arr[j].end \leq Arr[k].start$

$Arr[i].end \geq Arr[k].start$

이식을 나타내기위한 코드