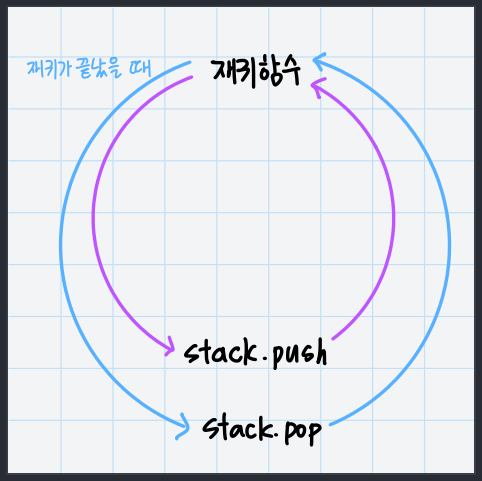
Algorithm 09 멱집합

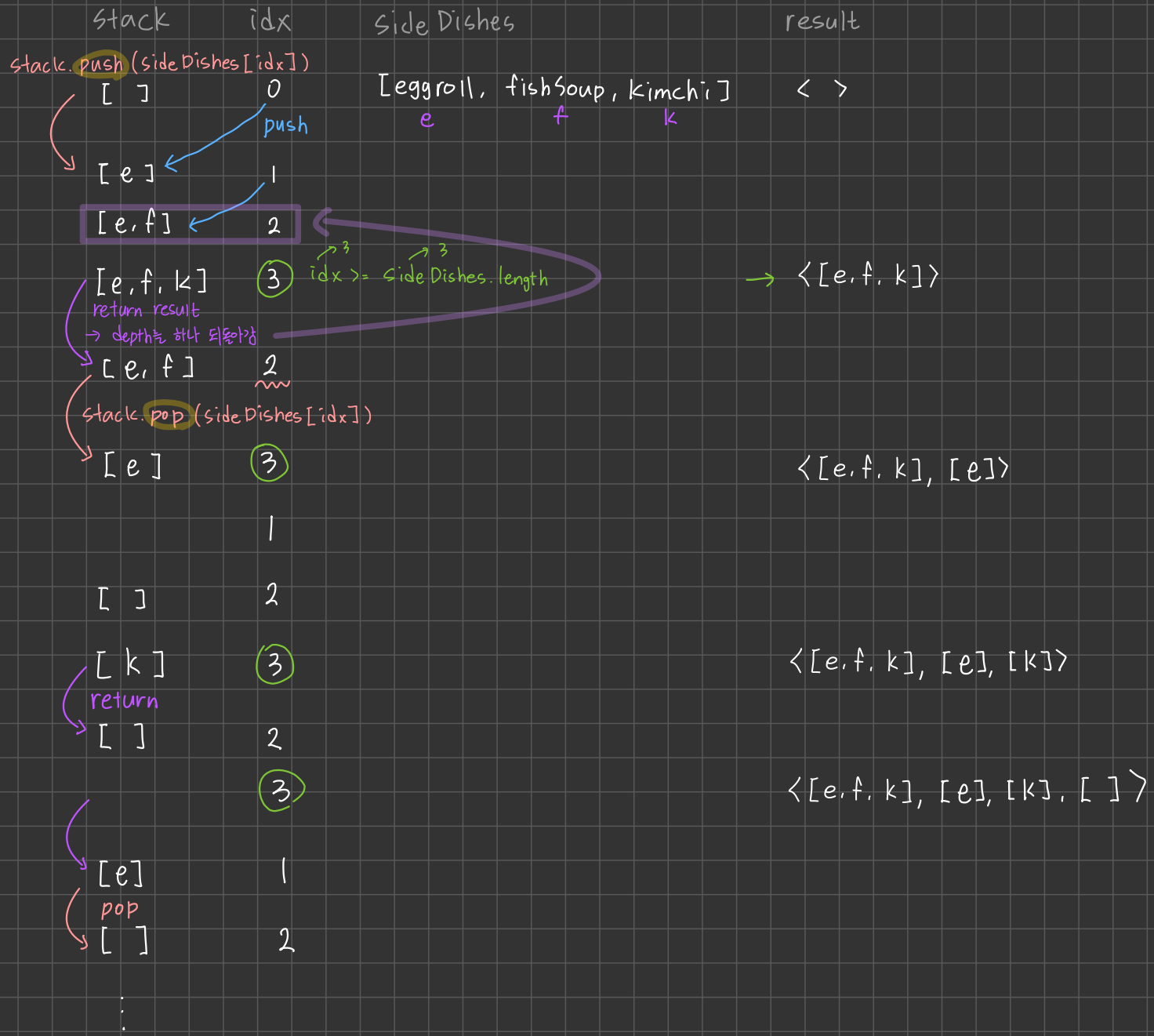
(1, 2)의 멱집합 : 4개
( ), (1), (2), (1,2)

```java
public static ArrayList<String[]> missHouseMeal(String[] sideDishes) {
    // TODO:
    // search 함수에서 사용할 빈 스택을 선언한다.
    Stack<String> stack = new Stack<>();
    // 결과를 담을 ArrayList를 선언한다.
    ArrayList<String[]> result = new ArrayList<>();
    //재료들을 오름차순으로 정렬한다.
    Arrays.sort(sideDishes);
    // 빈 스택과 0 번째 인덱스, 정렬된 재료로 구성된 배열과 결과를 담을 List를 인자로 받는 재귀 함수를 실행한다.
    result = search(stack, idx: 0, sideDishes, result);
    // 결과를 오름차순 순서로 정렬한다.
    result.sort((o1, o2) -> Arrays.toString(o1).compareTo(Arrays.toString(o2)));
    //결과를 반환한다.
    return result;
}
// 모든 조합을 검사하는 재귀함수를 작성한다.
3 usages  Kim-Jihyun1
public static ArrayList<String[]> search(Stack<String> stack, int idx, String[] sideDishes, ArrayList<String[]> result) {
    // 탈출조건
    if (idx >= sideDishes.length) {
        // idx와 sideDishes의 길이가 같거나 큰 경우 (마지막까지 검토한 경우), 스택을 배열로 변환하고해당 스택을 result에 넣는다.
        String[] arr = stack.toArray(new String[0]);
        result.add(arr);
        return result;
    } else {
        // idx가 부분집합에 포함된 경우
        stack.push(sideDishes[idx]);
        search(stack, idx: idx + 1, sideDishes, result);

        // idx가 부분집합에 포함되지 않은 경우
        stack.pop();
        search(stack, idx: idx + 1, sideDishes, result);
    }
    return result;
}
```

base case




stack
idx
side Dishes
result
stack.push(sideDishes[idx])
[ ]
0
[eggroll, fishSoup, kimchi]
e
f
k
< >
push
[e]
1
[e,f]
2
[e,f,k]
3
idx >= sideDishes.length
3
3
→ <[e,f,k]>
return result
→ depth도 하나 되돌아감
[e,f]
2
stack.pop(sideDishes[idx])
[e]
3
→ <[e,f,k], [e]>
1
[ ]
2
[k]
3
→ <[e,f,k], [e], [k]>
return
[ ]
2
3
→ <[e,f,k], [e], [k], [ ]>
[e]
1
pop
[ ]
2
: