1, 2번 모두 잘 푸신 것 같네요. 자신이 생각하는 것을 코드로 구현했다는 것이 대단합니다.

우선 기본적으로 모든 코드에는 시간 복잡도(Big-O) 라는 것이 존재합니다. 내가 만든 코드가 얼마나 효율적인지 계산하는 것입니다. 이부분에 대해서는 블로그나 유투브에 정리된 것이 많으니 한번 찾아보시면 좋을 것 같습니다.

def test1():  
 for i in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]:  
 print(i)

다음과 같은 test1함수가 있고, for 문 하나를 부를 때마다 1초가 걸린다고 가정을 해보겠습니다.

그러면 test1의 경우는 10초가 걸릴 것입니다. 마찬가지로 변수가 100개가 있다면 100초가 소요될 것입니다. 그래서 for문은 변수 길이(=n)에 시간이 소요되기 때문에, 시간 복잡도를 O(n) 이라고 표현합니다.

def test2():  
 for i in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]:  
 for j in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]:  
 print(i, j)

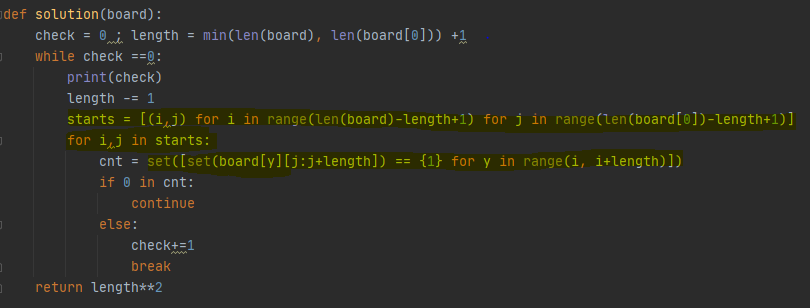
만약 다음과 같이 2중 반복문이 있다면, 시간이 얼마나 소요되는지 감이 오실 겁니다. 이 또한, 하나를 호출할 때 1초가 소요된다고 가정을 한다면, 약 100초가 소요될 것입니다. i, j가 100개씩 있다면 10000초가 소요되겠죠. 즉 시간 복잡도는 o(n^2)이 됩니다.

마찬가지로 3중 반복문이 있다면, 시간 복잡도는 o(n^3)이 소요될 것입니다.

def test3():  
 for i in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]:  
 print(i)  
 for i in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]:  
 for j in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]:  
 print(i, j)  
 for i in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]:  
 for j in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]:  
 for k in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]:  
 print(i, j, k)

만약 이러한 함수가 있다면 시간 복잡도는 o(n) + o(n^2) + o(n^3) 이 될 것입니다.

본인의 1번 문제를 살펴보면,

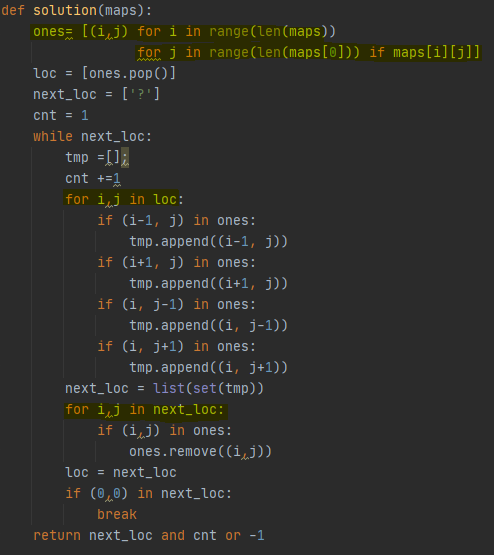


starts 변수에서 이미 2중 반복문이 실행되고, 그 밑에 또다시 2중 반복문이 실행됩니다.

그럼 while문이 반복할 때마다 2중 반복문이 두 번 반복될 것입니다.

즉, 시간계산을 해본다면, { o(n^2) + o(n^2) ) \* k (while문 반복 횟수) 이 될 것입니다.

마찬가지로 2번 문제를 살펴보면,



ones 함수에서 이미 2중 반복문이 실행되고 while문 반복할 때마다, for문이 2번 실행되는 구조입니다.

이것도 시간계산을 해본다면 o(n^2) + { o(n) + o(n) ) \* k 가 됩니다.

이처럼 내가 만든 코드에 대해서 시간 소요를 계산하실 수 있다면, 다른 답안의 시간 소요도를 계산하고 비교하여 속도가 얼마나 차이가 나는지 계산해보고 비교해보세요!

보통 시간 효율성 검사의 테스트 케이스로 들어오는 리스트 길이가 1, 10개 단위가 아닌 100, 1000, 10000… 경우에 따라 그 이상으로 들어옵니다.

즉 1000개를 돌리는데 for문 하나는 1초 이상, 2중 for문은 약 10초 이상, 3중 for 문은 100초 이상 소요될 것입니다.

따라서 ~님이 구현하였던 코드는 테스트 케이스가 1000개 들어왔다는 가정하에 1분 이상은 소요될 것으로 예상됩니다.