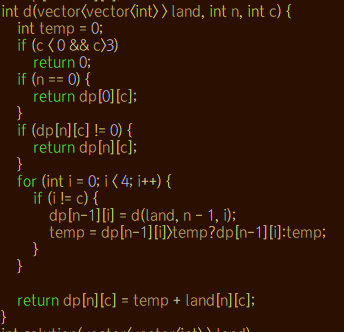
2020-11-25 기록

1. solution2의 시간 초과 해결



solution2에서 시간 초과가 발생했는데 이는 d함수가 계속 호출되면서 첫 번째 parameter로 받은 2차원 벡터 land도 따라서 복사되었기 때문이다.



이렇게 &로 참조만 하게 바꿔주면 복사하는 일이 발생하지 않기 때문에 시간 초과가 발생하지 않는다.

2. memoization과 dp의 정확한 개념

먼저 DP와 divide and conquer의 개념을 정리하자.

1) DP

DP는 입력 크기가 작은 부분 문제들을 해결하고 이 문제들의 해를 이용해서 큰 크기의 부분 문제를 해결해나가는 방식. 최종적으로 전체 문제를 해결하는 알고리즘

2) divide and conquer

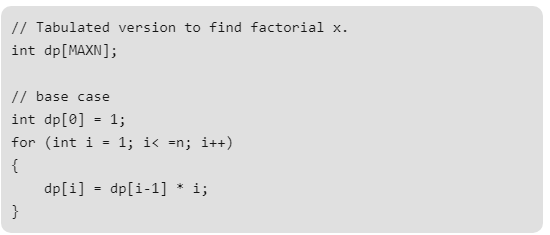
분할 정복은 문제를 나눌 수 없을 때까지 나누고 각 쪼개어진 문제를 해결하고 합병하면서 문제의 답을 얻는 알고리즘이다.

두 알고리즘의 문제를 나누어서 해결한다는 점은 동일하다. 하지만 DP는 나누어진 부분 문제가 중복될 수 있고, 분할 정복은 부분 문제가 중복되지 않는다는 차이점이 있다.

이미 가지고 있는 답을 재사용하는 방법은 2가지가 있다.

* Tabulation: Bottom Up

tabulation은 말 그대로 밑에서부터 축적하면서 최종적으로 답을 얻는 방법이다. DP문제의 답을 dp[x]라고 하자. 최종적으로 구하고자하는 답은 dp[n]인데 d[0]부터 최종 상태로 도달하는 것이 목표이다.



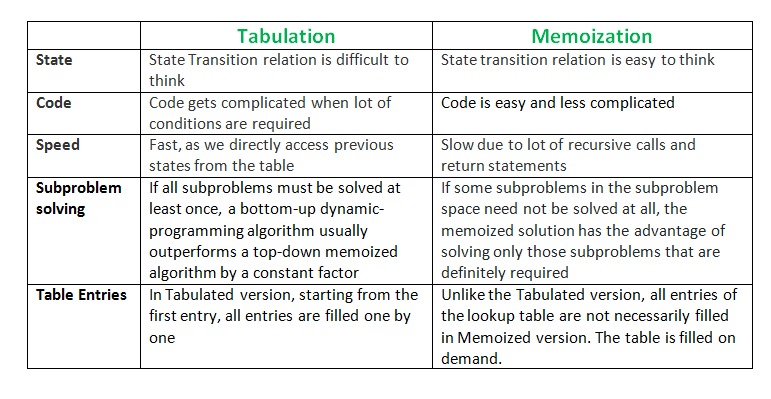
위의 코드를 보면 dp[i]의 값이 1부터 n까지 이전 dp값을 활용하면서 차근차근 계산되는 것을 알 수 있다.

* Memoization: Top Down

tabulation과 반대로 최종적으로 도달해야하는 dp[n]을 계산하기 위해 아래로 내려가면서 이전 state의 계산 결과를 활용하는 방법이다. 이전 값들은 모두 메모리에 저장해놓았기 때문에 중복되는 함수 호출을 피하고 바로 dp값을 반환할 수 있다.

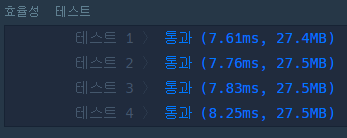
시간 복잡도의 경우 Memoization은 O(n)처럼 보이지만 Tabulation보다 느리다.

* Tabulation vs. Memoization



solution1에서는 tabulation을 이용한 DP로 문제를 풀었고, solution2에서는 memoization응 사용한 DP로 문제를 풀었다.

* Tabulation을 이용한 코드의 효율성 테스트



* Memoization을 이용한 코드의 효율성 테스트

