BruteForce - 강의 요약

왜 알고리즘(Algorithm)을 하는가?

- 실제 컴퓨터에서 프로그램을 만들 시, 메모리는 한정되어 있고 작업시간은 빠를 수록 좋다.
- 동일한 역할을 하는 프로그램에서 소요 메모리 또는 작업시간을 줄이려면 그만큼 메모리에 저장되어 있는 내용, 작업 중에 나오는 연산 횟수를 감소시켜야 한다
- 위와 같이 메모리와 작업 시간 요소를 줄이고자 나온 프로그램 최적화 식들이 알고리즘이다.

알고리즘의 시작을 완전 탐색으로 하는 이유

- 프로그램에서 소요되는 시간과 메모리를 알기 위해서 필요한 건 실제 연산 횟수와 장악하고 있는 변수의 내용물이다.
- 해당 프로그램의 연산 횟수를 파악하기 가장 좋은 건 프로그램에서 나올 수 있는 모든 경우의 수를 따져보는 것이다.
- 위처럼 경우의 수를 다 확인하는 것이 완전탐색 알고리즘이며 이후 경우의 수를 줄여나가는 알고리즘들이 나오게 된다.

연산 횟수(시간 복잡도)를 계산하는 방법

- 현실적인 방법
  - 현재 시간을 표시하는 함수를 이용하여 현재 시간과 종료 시간을 표시하는 변수 사이에 단순히 1억개 변수를 만들어서 넣는 작업을 한다
  - 이렇게 하면 현재 아키텍쳐가 1억 번 연산하는데 몇 초가 걸리는 지 알 수 있다.
- 어림짐작
  - 보통 1억번에 1초로 계산해서 푼다.

완전 탐색은 언제 하는가?

- 모든 경우의 수를 탐색 가능한가?
  - 즉, 모든 경우의 수를 탐색하는 문제이고 해당 문제에 대해 모든 경우의 수를 따질 수 있는 함수를 구성할 수 있으며 완전 탐색이 가능하다.
- 제한 시간 내에 풀 수 있는가?

    - 모든 경우의 수를 고려했을 때, 연산량에 따른 소요 시간이 제한 시간을 초과할 경우, 당연하게도 완전 탐색은 불가하다
- 내가 도저히 떠오르는 알고리즘이 없어서 부분 점수라도 얻어야 겠다
    - 보통 코테에서 연산량에 따라 부분 점수를 주는 문제가 있다. 해당 문제에서 최적화된 알고리즘이 떠오르지 않으면 연산량이 많더라도 적은 숫자에서 연산량을 만족하여 부분 점수라도 챙겨갈 수 있는 완전 탐색을 하는 게 낫다