**문제 해결 개관**

**1. 문제 해결 과정**

**1) 문제를 읽고 이해하기**

- 문제가 원하는 바를 이해한 후, 입출력 예제를 확인하자.

- 사소한 조건을 확실하게 확인한 후 작업을하자.

**2) 재정의와 추상화**

- 자신이 다루기 쉬운 개념을 이용해서, 문제를 자신의 언어로 풀어 써야한다.

- 직관적인 이해가 쉬워지며, 문제가 복잡할수록 유용하다.

- 현실의 문제의 본질은 남기되, 수학적/전산학적 개념으로 옮겨 적어야 한다.

- 문제의 본질을 어떠한 방식으로 남기는지에 따라 문제가 달라질 수도 있다.

**3) 계획 세우기**

- 어떠한 방식으로 해결할지 결정한 후, 어떠한 알고리즘과 자료구조를 사용할지 정한다.

- 가장 중요한 방식이지만, 문제에 따라 가장 오랜 시간을 투자하는 단계이기도 합니다.

**4) 계획 검증하기**

- 설계한 알고리즘이 모든 경우에 요구 조건을 정확히 수행하는지를 증명한다.

- 수행에 걸리는 시간과 사용하는 메모리가 문제의 제한 내에 들어가는지 확인한다.

**5) 계획 수행하기**

- 구현을 시작합니다.

**6) 회고하기**

- 문제 해결에 직접적인 영향은 없지만, 장기적으로 상당히 큰 영향을 준다.

- 처음 알고리즘보다 더 효율적인 알고리즘을 찾거나 간겨한 코드를 발견할 수 있다.

- 같은 알고리즘이라도, 더욱 직관적인 방법을 찾을 수 있습니다.

- 효과적인 회고를 수행하는 가장 좋은 방법은 문제를 풀 때마다

코드에 자신의 생각을 남기는 것이다.

- 문제의 간단한 해법, 어떤 방식으로 접근했는지, 문제의 해법을 찾은 결정적 깨달음을

기록한다.

- 반복되다보면, 문제를 푸는데 필요하는 공통된 통찰들이 패턴화 되기도 한다.

- 한번에 맞추지 못한 경우, 오답의 원인을 꼭 적는게 좋다.

- 같은 실수를 반복하지 않는데 큰 도움이 된다.

- 타인의 코드를 보며, 자신이 생각하지 못했던 방법을 보는 것도 도움이 된다.

**# 문제를 풀지 못하였을 때**

- 초보시절에는 어느정도 시간을 정해 두고 타인의 코드를 참고하는 것이 좋다.

- 단, 자신이 문제를 해결할 때 취했던 방법들을 복기를 꼭해야 한다.

**2. 문제 해결 전략**

- **직관과 체계적인 접근** 전략이다.

- **직관**은 해당 문제를 해결하는 알고리즘이 대략적으로 어떤 형태인지 짐작하게 해준다.

- 직관이 성공적으로 수행되기 위해서는, 많은 경험이 필요하지만 말이다.

- **체계적인 접근**은 문제를 해결하기 위한 기반을 쌓아올리면서 점진적으로 전진하는 것이다.

- 직관이 떠오르지 않을 떄, 문제에 질문을 적용하면서 문제 해결의 시작점을 만드는 것이다.

- 아래는 체계적인 접근하는데 유용한 질문들이다.

**1) 비슷한 문제를 풀어본적이 있는가?**

- 형태가 비슷하거나 관련된 문제를 풀어 본 적이 있다면,

이전에 적용했던 방법과 비슷한 접근 방법을 사용할 수 있다.

- 하지만, 완전히 동일한 문제는 존재하지 않기에, 기존에 풀어봤던 문제의 해결 원리를

완벽히 이해하고 변형할 수 있어야 한다.

- 하나의 알고리즘의 동작 과정과 원리에 대한 이해가 꼭 필요하다.

**# 문제**

- 합친 LIS ( 문제 ID : JLIS, 8장 연습 문제 )

- 신호 라우팅 ( 문제 ID : ROUTING, 30장 연습 문제 )

- 음주 운전 단속 ( 문제 ID : DRUNKEN, 30장 연습 문제 )

- 선거 공약 문제 ( 문제 ID : ROMISE, 30장 연습 문제 )

**2) 단순한 방법에서 시작할 수 있을까?**

- 복잡하게 풀지 않고, 간단하게 풀 수 있는지 확인한다.

- 효율적인 알고리즘이라도 단순한 알고리즘을 기반으로 구성된 경우가 많다.

- 이러한 해결이 가능한 문제인 경우, 더 효율적인 자료구조 또는

중복된 계산을 제외하는 방식으로 알고리즘을 개선할 수도 있다.

- 단순 알고리즘의 점진적인 개선은 향후 다른 알고리즘의 효율성의 척도가 되기도 한다.

- 알고리즘 책 P.30을 참조하여 예시를 보도록 하자.

**3) 내가 문제를 푸는 과정을 수식화할 수 있을까?**

- 처음에 생각한 것과는 달리, 새로운 방향에서 접근해야 풀리는 문제를 만날때가 있다.

- 이러한 문제는, 간단한 손코딩을 통해서 알고리즘을 진행하다보면 발견되는 경우가 많다.

- 손코딩을 진행하면서, 문제와 알고리즘의 어떤 점을 고려해야 할지 발견하기 때문이다.

- 나중에, 디버깅 과정에서 고려하지 못한 점을 발견 하는 것보다 훨씬 시간이 이득이다.

**# 문제**

- Quantization ( 문제 ID : QUANTIZE, 8장 연습문제 )

- 두니발 박사의 탈옥 ( 문제 ID : NUMB3RS, 8장 연습문제 )

- 실험 데이터 복구하기 ( 문제 ID : RESTORE, 9장 연습문제 )

- 출전 순서 정하기 ( 문제 ID : MATCHORDER, 10장 연습문제 )

- 마법의 약 ( 문제 ID : POTION, 14장 연습문제 )

- 함정 설치 ( 문제 ID : TRAPCARD, 32장 연습문제 )

**4) 문제를 단순화할 수 없을까?**

-

# 참고 자료

제목 : 프로그래밍 대회에서 배우는 알고리즘 문제 해결 전략,

저자 : 구종만,

발행 날짜 : 2012.11.21

출판사 : 인사이트