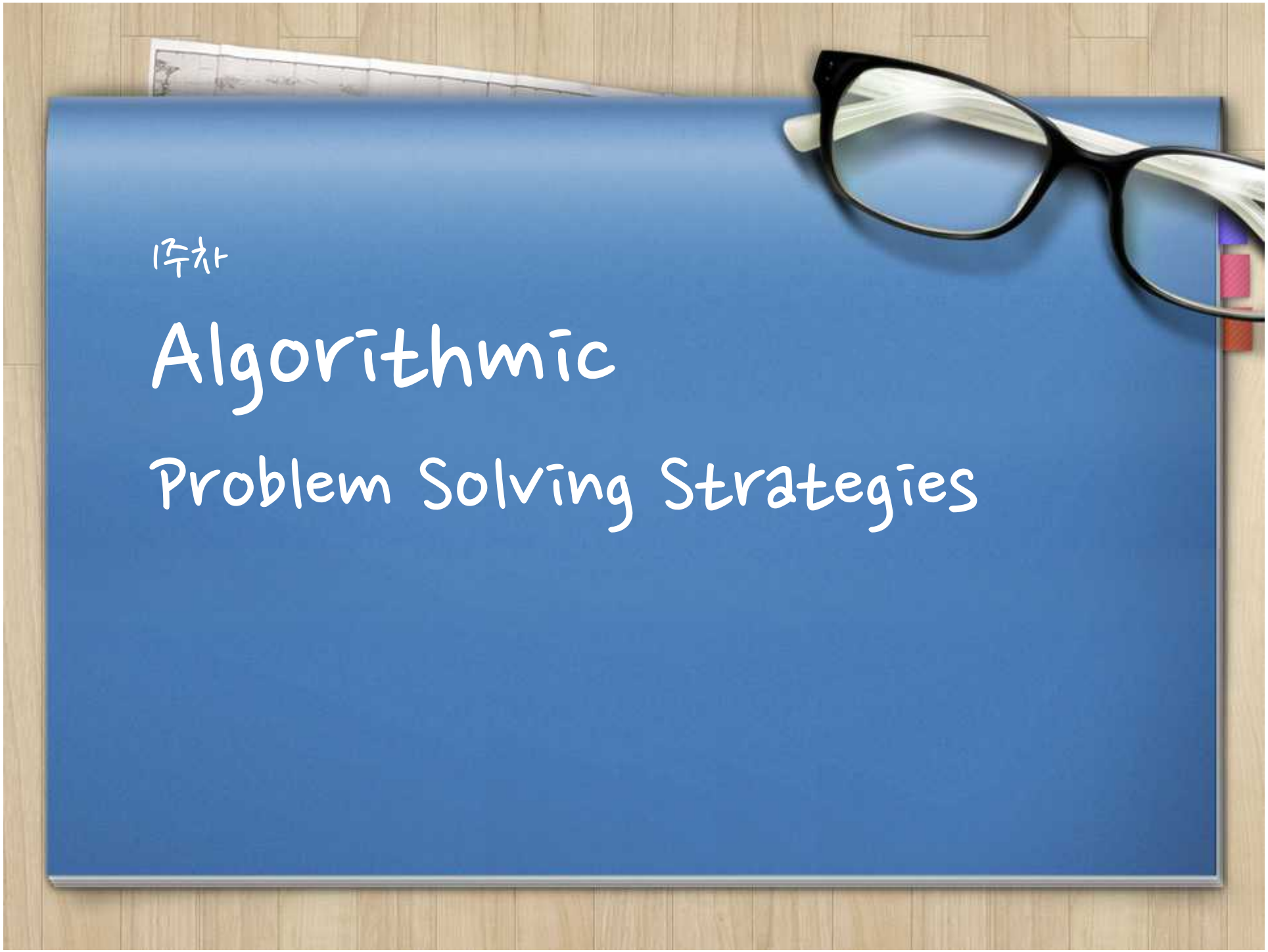

1주차
Algorithmic
Problem Solving Strategies

목차

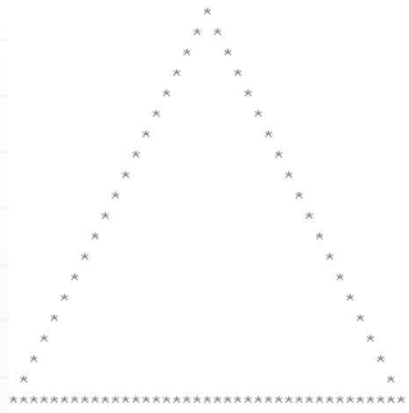
1. Introduction to Algorithm
2. Problem Solving Strategies
3. Practice

0. warming up!!

- 별 찍기! 제한시간 10분

- 2442 별찍기 - 5

- 10992 별찍기 - 17



1. Introduction to Algorithm

- 목표

- 컴퓨터적, 수학적 사고의 향상
- 문제 해결을 위해 적절한 알고리즘을 찾고 이를 구현하여 적용
- Data structure(2학년 2학기), Algorithm(3학년 1학기) 과목에 대한 선행학습
- 프로그래밍 경진대회 참여
  (ACM-ICPC, Topcoder, Google code Jam, 등)

1. Introduction to Algorithm

- curriculum

- 문제 해결 전략
- 코딩과 디버깅
- 알고리즘의 시간 복잡도 분석
- 무식하게 풀기
- 분할 정복
- 동적 계획법
- 선형 자료 구조
- 큐, 스택, 데크
- 트리의 구현과 순회
- 이진 검색 트리
- 우선순위 큐와 힙
- 그래프의 표현과 정의
- 그래프의 깊이 우선 탐색(DFS)
- 그래프의 너비 우선 탐색(BFS)
- 최단 경로 알고리즘

구종만 저 『프로그래밍 대회에서 배우는 알고리즘 문제해결전략』 참고

1. Introduction to Algorithm

- 알고리즘 맛보기: 8393 (합) 제한시간 10분

$$\sum_{k=1}^{n} k$$

1. 컴퓨터적 사고
2. 수학적 사고
3. 문제해결 전략
4. 다양한 풀이방법

1. Introduction to Algorithm

- Algorithm

- 어떠한 문제를 해결하기 위한 여러 동작들의 모임
- 알고리즘은 정확성, 최적성, 작업량 복잡도 등으로 분석
- 복잡도는 시간 복잡도와 공간 복잡도가 있음
- 알고리즘은 유한성을 가지며 언젠가는 끝나야 함
- 알고리즘은 무작정 외우고 문제를 푼다고 해서 실력이 쌓이는게 아님
- 따라서 이런 추상적인 개념을 해결하기 위해 문제해결전략을 도입해야 함

2. Problem Solving Strategies

- 문제해결전략

- 문제를 해결하기 위해 무작정 알고리즘을 외우는 것이 아닌 문제를 푸는 기술을 연마하는 것
- 자신이 문제를 어떤 방식으로 해결하는지를 의식하고 부족한점과 개선점을 파악할 것

- 책에서는 다음과 같은 문제해결전략을 제안 함

1. 문제를 읽고 이해한다
2. 문제를 익숙한 용어로 재정의한다
3. 어떻게 해결할지 계획을 세운다
4. 계획을 검증한다
5. 프로그램으로 구현한다
6. 어떻게 풀었는지 돌아보고, 개선할 방법이 있는지 찾아본다

2. Problem Solving Strategies

- 문제를 읽고 이해하기

- 알고리즘 문제를 풀 때 그림과 입출력 예제 등은 원하는 것을 유추하기 쉬움
- 따라서 문제를 공격적으로 읽으며 문제가 원하는 바를 정확하게 이해하는 과정이 반드시 필요
- 직관적인 생각은 문제를 어떤 방식으로 풀어나가야 할지 열개가 잡힐 수 있음
- 문제를 단순화(제약조건의 완화, 계산 변수 수를 줄임, 더 낮은 차원에서 생각해보기 등)

2. Problem Solving Strategies

- 문제를 익숙한 용어로 재정의하기

- 문제가 요구하는 바를 직관적으로 이해하기 위해 꼭 필요한 작업
- 문제의 추상화(현실 세계의 개념을 수학적 개념으로 옮겨 표현)

  이는 현실 세계의 복잡한 개념들을 어느 정도 본질만을 남겨두고 축약하여 다루기 쉽게 끔 만드는 것
- 추상화 과정이 프로그래밍이 나아갈 방향을 결정한다고 볼 수도 있음
- 이를 통해 체계적인 접근을 할 수 있음
  - 비슷한 문제를 풀어본 적이 있는가?
  - 단순한 방법에서 시작할 수 있을까?
  - 무식하게 풀 수 있을까?
  - 문제를 그려볼 수 있을까?
  - 수식으로 표현할 수 있을까?
  - 문제를 분해할 수 있을까?
  - 뒤에서부터 생각해서 문제를 풀 수 있을까?
  - 순서를 강제할 수 있을까?
  - 특정 형태의 답만 고려할 수 있을까?

2. Problem Solving Strategies

- 어떻게 해결할지 계획 세우기

- 사용할 알고리즘과 자료구조의 선택
- 이를 위해 잘 알려진 알고리즘과 자료구조에 대해 공부

2. Problem Solving Strategies

- 계획 검증과 계획 수행

- 구현을 시작하기 전 설계한 알고리즘이 모든 경우에 요구 조건을 정확히 수행하는지 증명
- 수행에 걸리는 시간과 사용하는 메모리가 문제의 제한 내에 들어가는지 확인
- 이를 통해 계획의 수행 즉, 프로그램을 작성

2. Problem Solving Strategies

- 회고하기

- 자신이 문제를 해결한 과정을 돌이켜 보고 개선하는 과정
- 문제를 여러 번 풀으로써 더 효율적인 알고리즘을 찾거나 간결한 코드를 작성하는 등을 할 수 있음
- 이를 위해 코드 작성 시 자신의 경험을 기록으로 남기는 것이 큰 도움이 될 수 있음
- 특히 한번에 풀지 못한 경우 오답의 원인을 적는 것이 좋음
- 다른 사람의 코드를 참고하여 더 나은 방법을 찾을 수도 있음
- 따라서 이 스터디를 통하여 이러한 과정을 겪으며 같이 성장했으면 하는 바

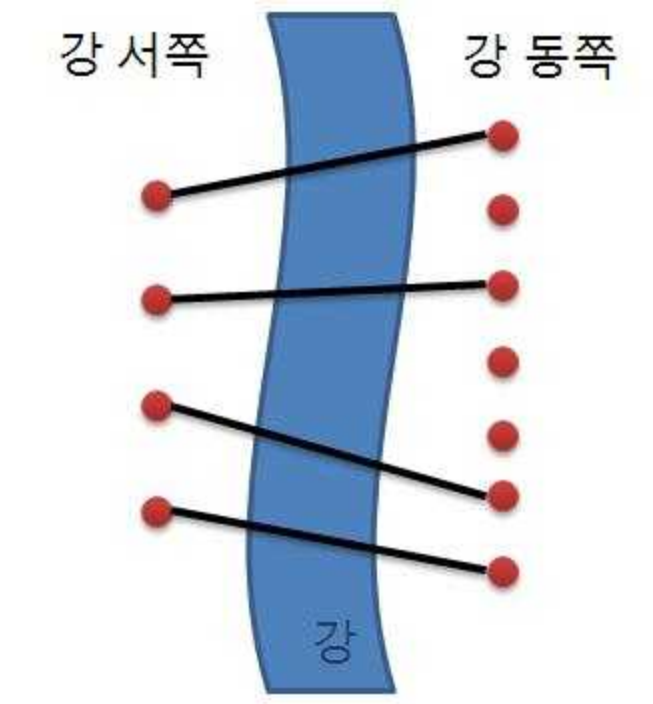
2. Problem Solving Strategies

- 주의!!

- 풀지 못하는 문제에만 계속 매달려 있는 것도 좋지 않음
- 어느정도 고민해보고 답을 찾지 못할 때는 다른 사람의 코드나 풀이방법을 참조하기
- 문제를 풀 때마다 항상 이러한 단계를 하나하나 맞출 필요는 없음
- 이는 생각을 돕기 위한 도구이므로 어느정도 숙달이 되면 이를 의식하지 않아도 자연스럽게 수행할 수 있음

2. Problem Solving Strategies

- 알고리즘 맛보기: 1010 (다리놓기) 제한시간 15분



1. 수학적 사고
2. 컴퓨터적 사고
3. 문제해결 전략
4. 다양한 풀이 방법

2. Problem Solving Strategies

- 함께 풀어보기: 2407 (조합) 제한시간 5분

문제

$_nC_m$을 출력한다.

입력

n과 m이 주어진다. ($5 \le n \le 100$, $5 \le m \le 100$, $m \le n$)

출력

$_nC_m$을 출력한다.

예제 입력 복사

```
100 6
```

예제 출력 복사

```
1192052400
```

1. 수학적 사고
2. 컴퓨터적 사고
3. 문제해결전략

Thank u!