


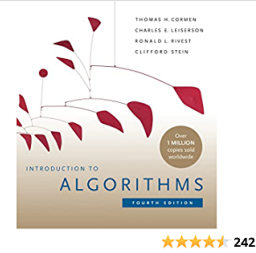
# Algorithm

## Introduction to Algorithms, fourth edition

Introduction to Algorithms, fourth edition: 9780262046305:

Computer Science Books @ Amazon.com

 [https://www.amazon.com/dp/026204630X/ref=mp\\_s\\_a\\_1\\_3?crid=WDS979R5CAKG&keywords=algorithm&qid=1674170628&sprefix=algorithm%2Caps%2C409&sr=8-3](https://www.amazon.com/dp/026204630X/ref=mp_s_a_1_3?crid=WDS979R5CAKG&keywords=algorithm&qid=1674170628&sprefix=algorithm%2Caps%2C409&sr=8-3)



## Java Online Compiler & Interpreter

Write and run Java code using our Java online compiler & interpreter. You can build, share, and host applications right from your browser!

 <https://replit.com/languages/java>



## Flowchart Maker & Online Diagram Software

diagrams.net (formerly draw.io) is free online diagram software. You can use it as a flowchart maker, network diagram software, to create UML online, as an ER diagram tool, to design database schema, to build BPMN online, as a circuit diagram maker, and more. draw.io can import .vsdx, Gliffy™ and

 <https://app.diagrams.net/>

- Algorithm

컴퓨터가 아직은 사람의 언어를 이해하지 못한다. 따라서 알고리즘을 프로그래밍 언어로 기술한 것을 프로그램이고 한다.

알고리즘은 문제나 과제를 해결하기 위한 처리 절차를 하나하나 구체적인 순서에 따라 표현한 아이디어 또는 생각을 말한다.

알고리즘은 컴퓨터에서만 사용되는 것이 아니다. 일상생활에서도 많이 사용된다. 예로는 레시피, 약보, 설명서 등이 있다.

그러나 알고리즘은 아이디어나 생각이기 때문에 표현하기 어렵다. 그래서 표현하는 도구로

1. psuedo code 의사코드
2. flow chart 순서도, 흐름도
3. program

- Algorithm & Program

알고리즘을 프로그래밍 언어 C, Java, Python 등으로 기술한 것이 프로그램이다.

알고리즘 자체는 눈으로 볼 수 없다. 그래서 이것을 전달하기 위해 문장이나 일러스트 등으로 표현한다.

하지만 컴퓨터에 전달하려면 ?

프로그래밍의 절차

기획 → 설계(알고리즘) → 프로그래밍 → 디버그 → 문서화

- Good Algorithm

좋은 프로그램은 알고리즘의 좋고 나쁨에 달렸다. 좋은 알고리즘은 어떻게 판단 ?

1. 알기 쉽다

가능한한 알기/이해하기 쉬워야 한다. 특히 여러사람이 작업할 때 다른 팀원들도 바로 이해할 수 있어야 한다. 복잡하고 난해한 알고리즘은 올바른 결과가 나오는지 검증하기도 어려워 틀린 부분을 찾기도 어렵게 된다. 따라서 알기 쉽고 이해하기 쉽게 작성하자.

2. 속도가 빠르다

속도가 빠르다는 것은 실행하고 그 결과가 나올때까지의 시간이 짧다는 것을 말한다. 결과 출력의 시간이 길다면 좋은 알고리즘이라고 할 수 없다.

3. 효율적이다 (메모리를 적게 차지한다)

프로그램을 실행할 때 사용되는 메모리의 영역이 작다는 것을 의미한다.

4. 재이용 용이

프로그래밍 시간을 단축하려면 코딩하는 속도를 높이는 것도 하나의 방법이 될 수 있지만 이것을 한계가 금방 오게 된다. 과거에 작성한 프로그램을 그대로 또는 부분적으로 이용하는 비율을 증가하게 되면 새로운 프로그램을 만드는 시간도 줄어들게 된다.

따라서 프로그램을 작성 할 때 가능하면 재사용이 쉽고 범용성이 높은 알고리즘을 고려하면 좋은 프로그램을 만들 수 있게 된다.

## 좋은 알고리즘의 조건

- 정확한 결과를 얻을 수 있어야 한다.
- 반드시 종료되어야 한다.

## 왜 알고리즘을 공부해야 하나 ?

- 좋은 프로그램을 만들 수 있다.
- 프로그램의 좋고 나쁨을 판단 할 수 있다.
- 프로그램 작성 과정 전체를 효율화 할 수 있다.
- 프로그래밍 기술을 향상 시킬 수 있다.

## • 3 Basics 알고리즘 3가지 기본형

### 1. 순차구조

처음부터 순서대로 처리하는 절차

### 2. 선택구조 if

조건식으로 판단하여 실행 할 처리를 전환하는 절차

### 3. 반복구조 while/for

조건을 만족하는 동안 같은 처리를 반복하는 절차

반복구조는 효율적인 알고리즘의 핵심 포인트다.

모든 알고리즘은 아무리 복잡해보여도 위 3가지의 조합으로 표현 가능하다. 대부분의 알고리즘을 위 3가지 구조로 작성 할 수 있다.

시작  
종료

처리

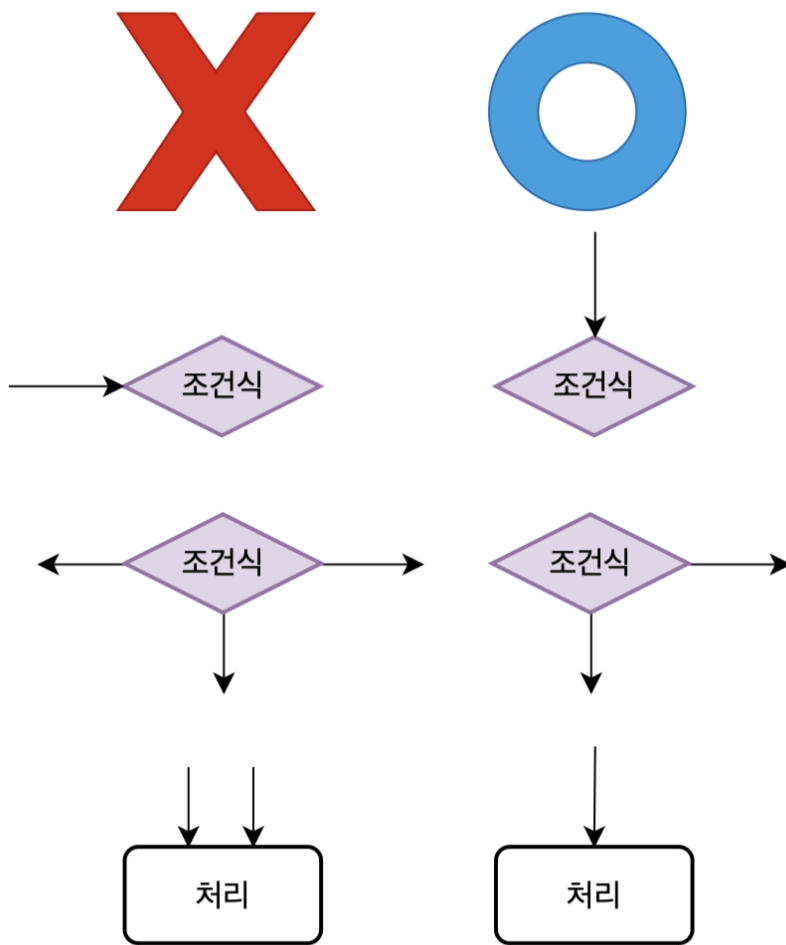
데이터

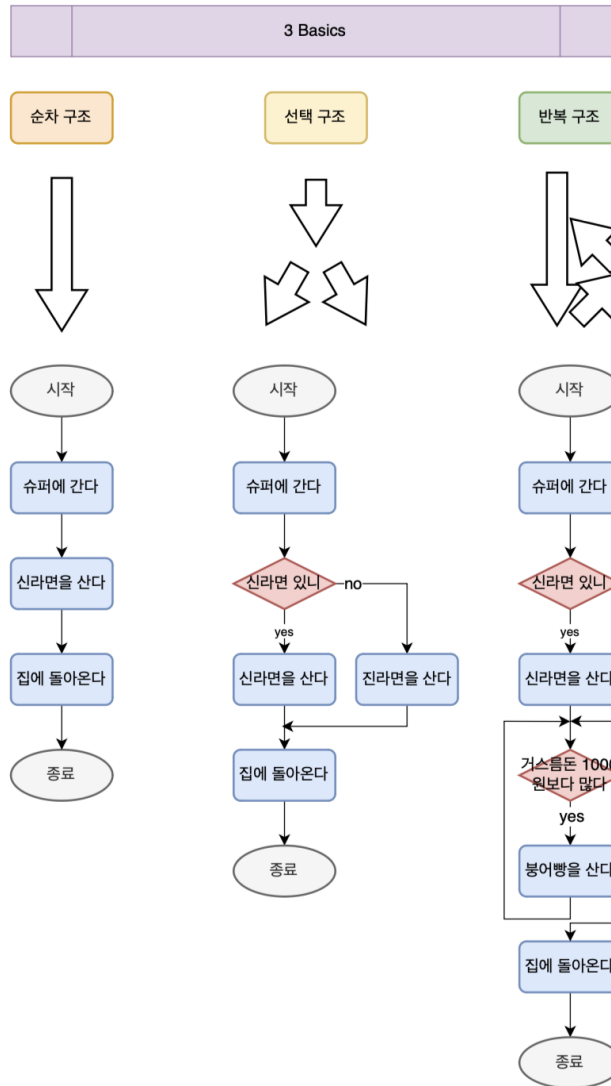
출력

조건식

반복문

처리





- Triangle

삼각형의 면적을 구하는 알고리즘

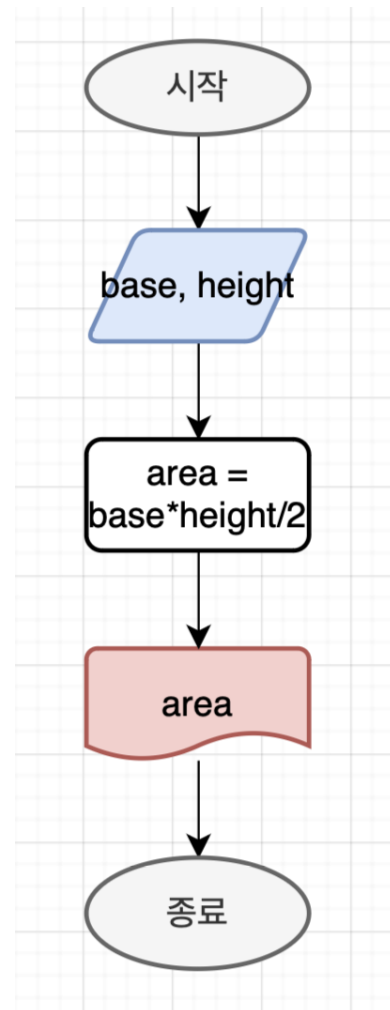
1. 순차적 분해하여 절차적으로 생각한다
2. 사칙 연산 처리는 산술 연산자를 사용한다
3. 나눗셈은 / 기호를 사용한다

우리가 이미 알고 있는 공식

$$\text{밑변} * \text{높이} / 2$$

변수 필요

면적 area  
 밑변 base  
 높이 height  
 의사코드로 생각해보기  
 base, height 입력  
 $\text{base} * \text{height} / 2 \rightarrow \text{area}$   
 area 출력



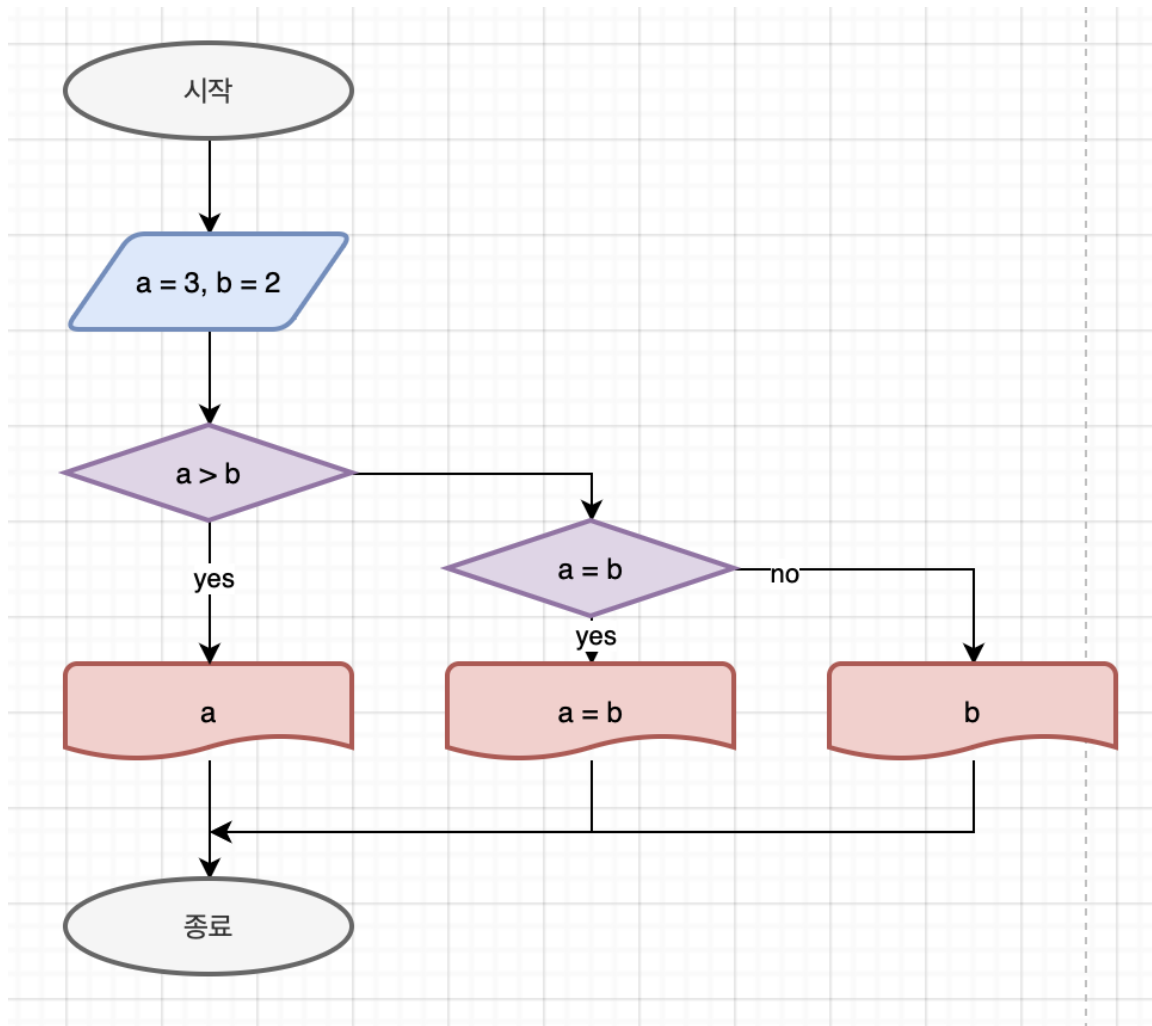
- Bigger

두 수의 대소를 판별

1. 두개의 데이터를 비교하기 위해 선택 구조 필요
2. 조건식에서는 관계 연산자를 사용

데이터를 비교하는 처리를 고려하기 위해

a와 b를 비교하여 a가 크면 a를 출력하고 그렇지 않으면 b를 출력한다



- Exchange
- Sum
- Ceiling Value
- Linear Search