2장. 소프트웨어와 알고리즘

Software

Algorithm

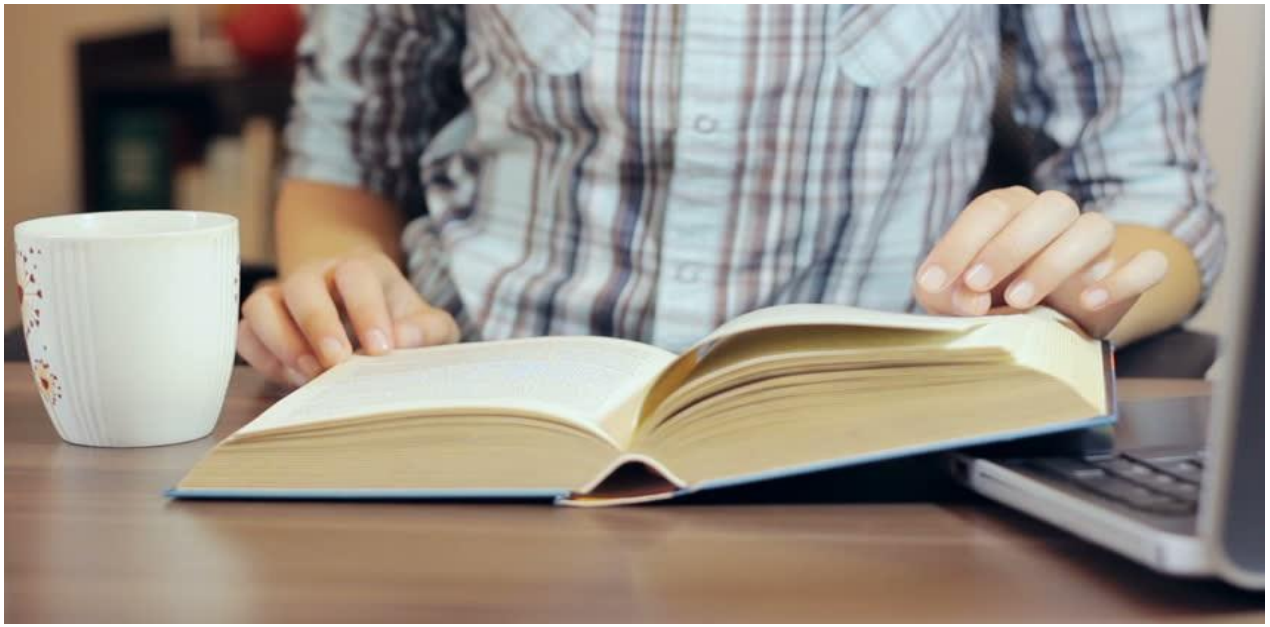
How to solve?

Language

How to represent?

알고리즘 (1/5)

책 읽기



알고리즘 (2/5)

- 자연수 n이 주어질 때,

$$1 \times 2 \times \cdots \times n$$

$$1! = 1$$
$$2! = 2(1) = 2$$
$$3! = 3(2)(1) = 6$$
$$4! = 4(3)(2)(1) = 24$$
$$5! = 5(4)(3)(2)(1) = 120$$

알고리즘 (3/5)

- 숫자가 적힌 공이 가득한 주머니에서 가장 큰 숫자 찾기

알고리즘 (4/5)

- 쇼핑목록으로 장바구니 확인하기

알고리즘 (5/5)

- 비밀번호 해킹(Hacking)

알고리즘의 복잡도 (1/5)

- 비밀번호 해킹의 사례에서, 비밀번호 자릿수가 n개라면, 가능한 조합은 10^n개

➢ 처리속도는 얼마나 될까 ?

- PC 처리속도가 3GHz 일 때, 초당 $9*10^6$ 이상 연산
- 쿼드코어의 경우 초당 10^8 이상의 연산
- n = 20이면, 약 27시간 이상 ⇒ n이 커지게 되면 많은 시간이 걸리게 됨

- 알고리즘의 실행비용에 대해서 관심을 갖는 부분은 n이 커질 때 비용이 어떻게 증가하는지 ⇒ 복잡도

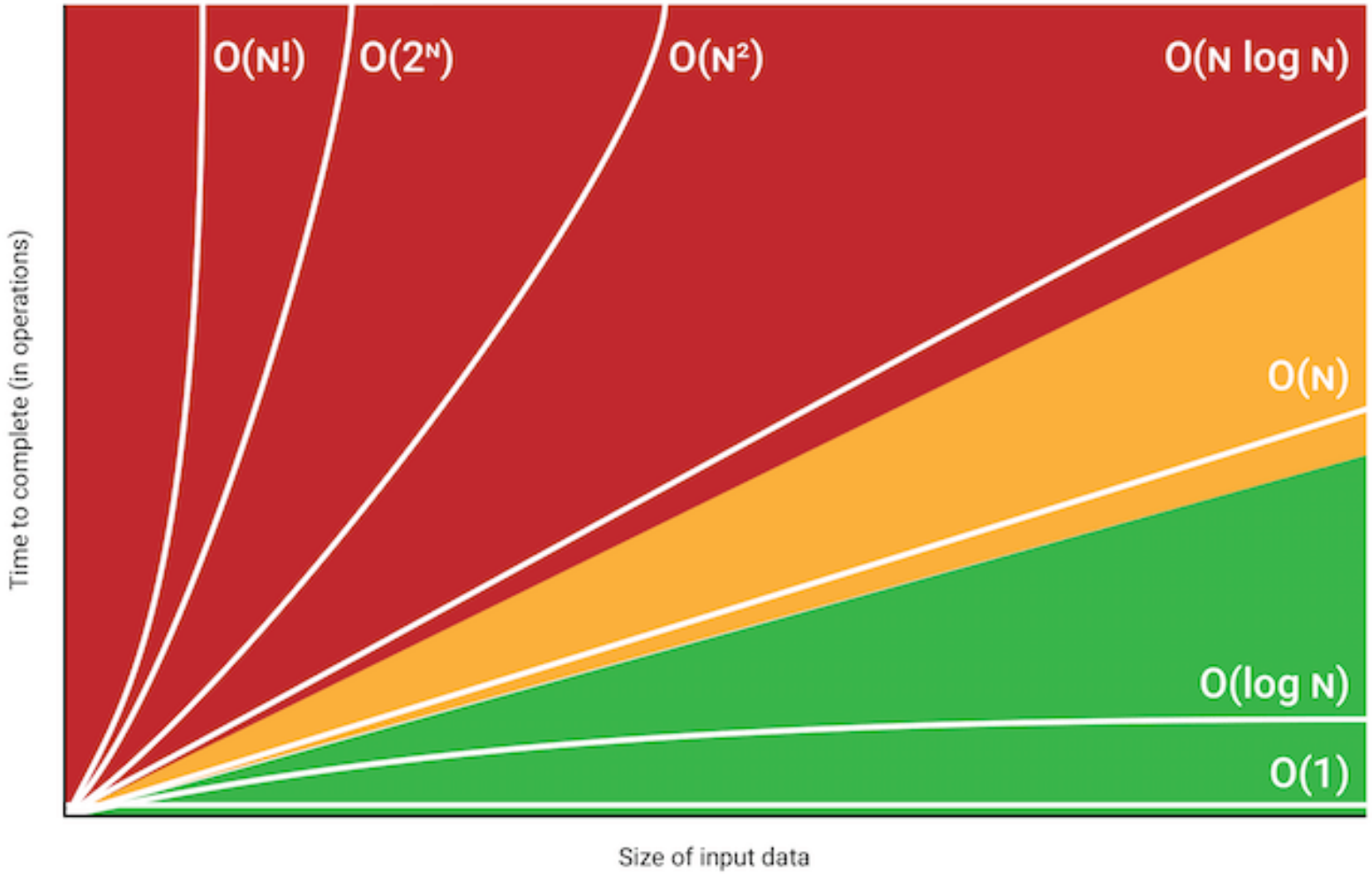
알고리즘의 복잡도 (2/5)

- 복잡도(Complexity)
  - 알고리즘의 성능을 나타내는 척도
  - 시간 복잡도(Time complexity) : 특정 크기의 입력에 대해 알고리즘의 연산이 몇 번 이루어지는지 표현
  - 공간복잡도(Space complexity) : 특정 크기의 입력에 대해 알고리즘이 얼마나 많은 메모리를 차지하는지 표현
- 빅오 표기법(Big O notation)
  - 복잡도를 표현하는 표기법
  - 가장 빠르게 증가하는 항만을 고려하여 표현
  - O(1), O(logn), O(N), $O(N^2)$, …

알고리즘의 복잡도 (3/5)

n	n^2	2^n	$n!$
1	1	2	1
2	4	4	2
3	9	8	6
4	16	16	24
5	25	32	120
6	36	64	720
7	49	128	5,040
8	64	256	40,320
9	81	512	362,880
10	100	1,024	3,628,800
20	400	1,048,576	2,432,902,008,176,640,000
30	900	1,073,741,824	너무 큼 $> 10^{39}$

알고리즘의 복잡도 (4/5)


O(N!)
O(2^N)
O(N²)
O(N log N)
O(N)
O(log N)
O(1)
Time to complete (in operations)
Size of input data

알고리즘의 복잡도 (5/5)

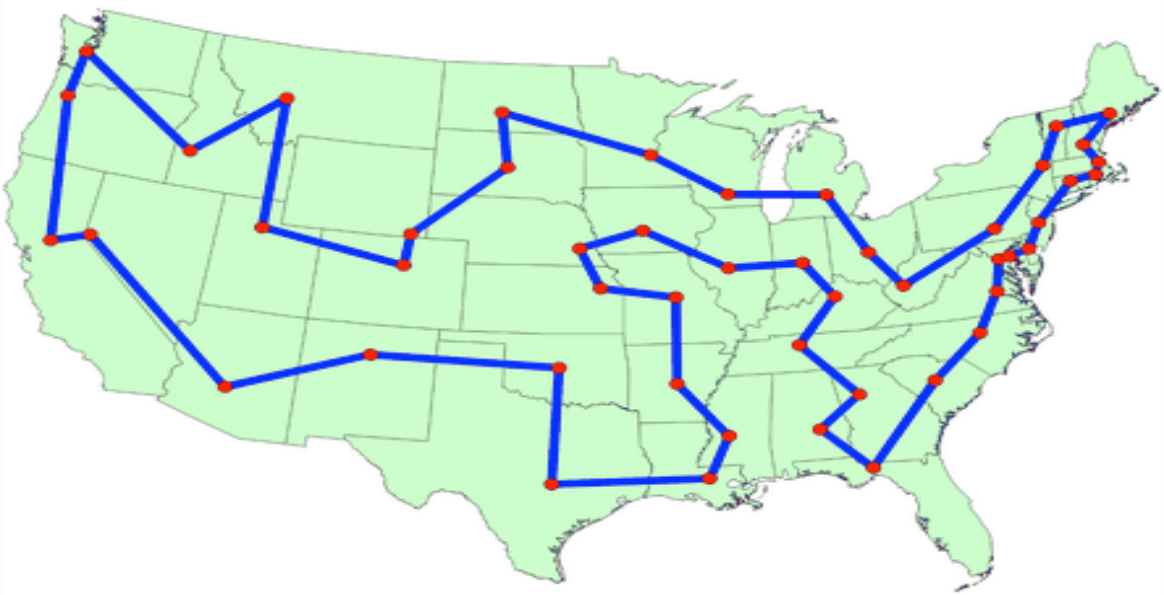
- 쓸만한 알고리즘이란 ?
  - 비용을 지불해서 사용할만한 것이 현실적인 알고리즘 ⇒ 보통 다항시간(n^k)을 기준으로 봄

Realistic	vs	Unrealistic
Polynomial **Class**		**N**on-deterministic **P**olynomial **Class**

  - 다항시간의 반대는 지수시간(k^n) : 입력 크기가 커지면 복잡도가 기하급수적으로 커짐 ⇒ NP 문제들

NP 클래스 사례 (1/2)

- Traveling Salesman

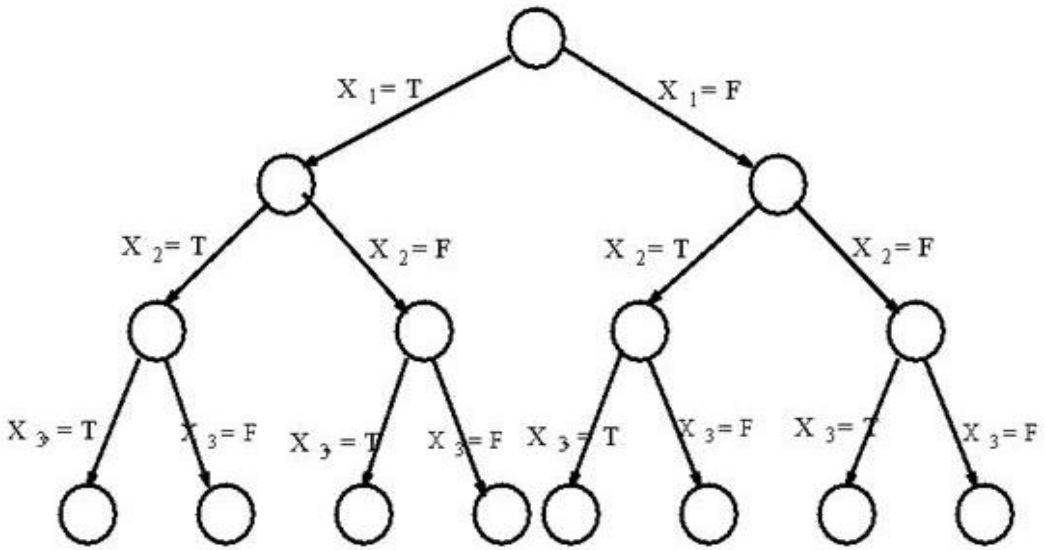


Hamitonian Cycle : O(n!)

NP 클래스 사례 (2/2)

- Boolean Satisfiability Problem

x_1	x_2	x_3
F	F	F
F	F	T
F	T	F
F	T	T
T	F	F
T	F	T
T	T	F
T	T	T



Tree representation of 8 assignments.

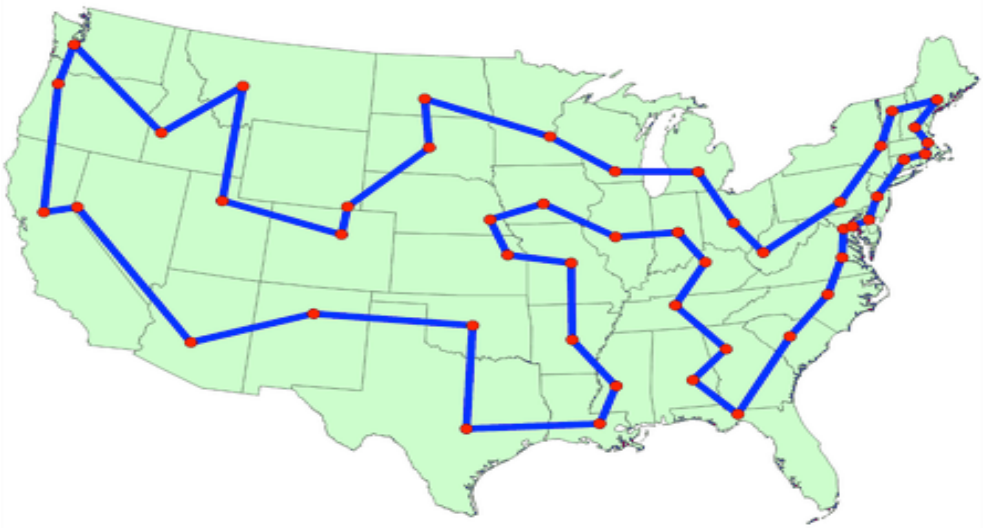
If there are n variables $x_1, x_2, \ldots, x_n$, then there are 2^n possible assignments.

How to Solve Problems in NP?

- 휴리스틱 알고리즘 (Heuristic Algorithm)
  - 휴리스틱은 비슷한 문제에 대한 과거의 경험들을 바탕으로 직관적으로 판단하여 선택하는 의사결정 방식
  - 휴리스틱은 문제 해결을 위한 규칙 또는 경험적인 지침으로, 최적해를 찾는 것보다 빠르게 문제를 해결하고자 할 때 적용
  - 정답이 보장되어 있는 것이 아니라, 적당한 답을 찾아 줌
  - 정확성에 대한 증거가 없으며, 최적의 결과를 산출하지 못할 수도 있음
  - 인공지능 분야에서 많이 사용

How to Solve Problems in NP?

- 랜덤 알고리즘(Randomized Algorithm)
  - 선택이 필요한 기로에서 동전을 던져서 결정한다는 의미
  - 즉, 난수를 발생시켜 진행과정을 결정하는 방법
  - 난수는 확률변수이므로 확률적 알고리즘이라고도 부름



How to Design Algorithm?

- 모조리 훑기(Exhaustive search)
- 되돌아가기 (Backtracking)
- 나눠 풀고 합치기 (Divide-and-Conquer)
- 기억하며 풀기 (Dynamic Programming)
- 질러 놓고 다듬기 (Iterative Improvement)

Software

Algorithm

How to solve?

Language

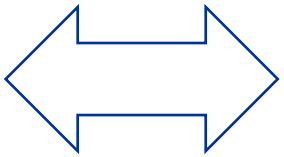
How to represent?

Machine Language

- 컴퓨터는 0과 1만을 이해할 수 있음

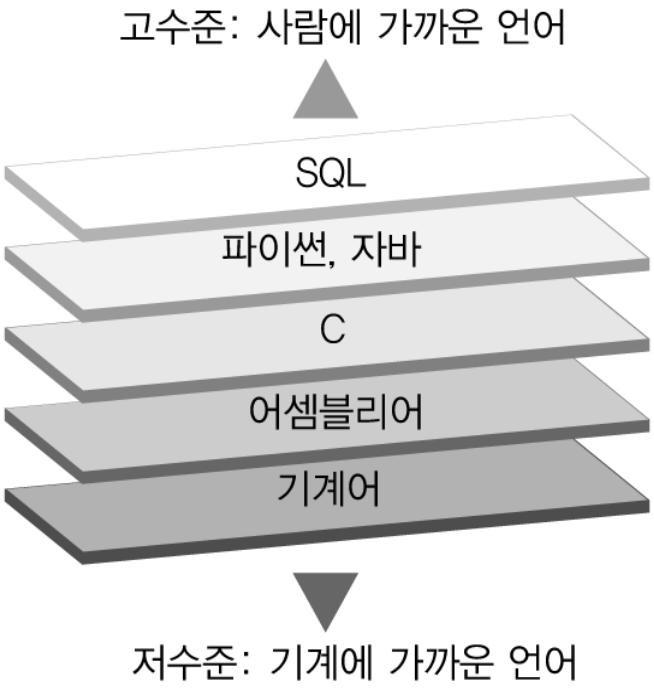
Language Translation (1/7)

- 표현은 사람들이 이해하기 쉬운 방법으로..
- 기계가 이해할 수 있도록 번역해주기



Language Translation (2/7)

- 프로그래밍 언어



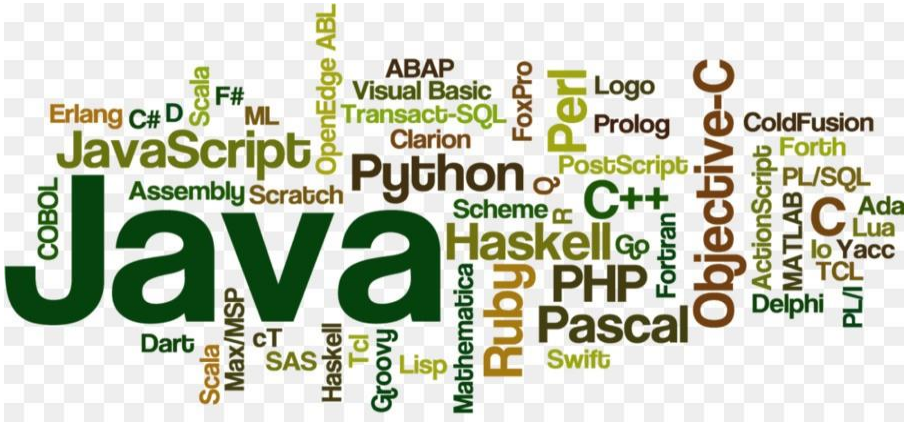
Python
```
1789 + 211
```

어셈블리어
```
mov   rax, 0x6fd
add   rax, 0xd3
```

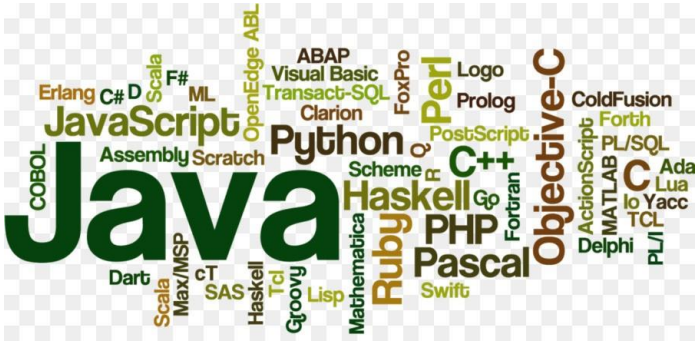
기계어
```
01001000 11000111 11000000
11111101 00000110 00000000
00000000
01001000 00000101 11010011
00000000 00000000 00000000
```

Language Translation (3/7)

- 고수준 프로그래밍언어(High-level Programming Languages)
- 튜링 완전(Turing Complete)
  - 어떤 프로그래밍 언어가 튜링머신과 동일한 계산능력을 가진다는 의미
  - 즉, 튜링머신으로 풀 수 있는 문제는 그 프로그래밍 언어로 풀 수 있다는 의미



Language Translation (4/7)



High-level Language

Compiling

Machine Language

Language Translation (5/7)

Java

Compiling

```
Field public java.awt.Image carvequest       # DATA XREF: init+182↓w ...
Field public java.awt.Image inst             # DATA XREF: init+196↓w ...

# ============================================================================

# Segment type: Pure code
Method public void init()
max_stack 6
max_locals 4
{
  aload_0 # var001_0
  aload_0 # var001_0
  invokevirtual java.awt.Dimension java.awt.Component.size()
  getfield int java.awt.Dimension.width
  putfield int width
  aload_0 # var001_0
  aload_0 # var001_0
  invokevirtual java.awt.Dimension java.awt.Component.size()
  getfield int java.awt.Dimension.height
  putfield int height
  aload_0 # var001_0
  ldc1 "GRIDWIDTH"
  invokevirtual java.lang.String java.applet.Applet.getParameter(java.lang.String)
  astore_1 # var001_1
```

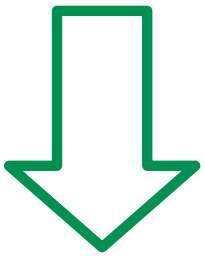
Java Bytecode

Language Translation (6/7)

source :
Arithmetic Language

$$E \longrightarrow n \mid -E \mid E_1 + E_2$$

```
-(3 + 4)
```



번역 규칙

$$trans(n) = \text{push } n$$
$$trans(E_1 + E_2) = trans(E_1).trans(E_2).\text{add}$$
$$trans(-E) = trans(E).\text{minus}$$

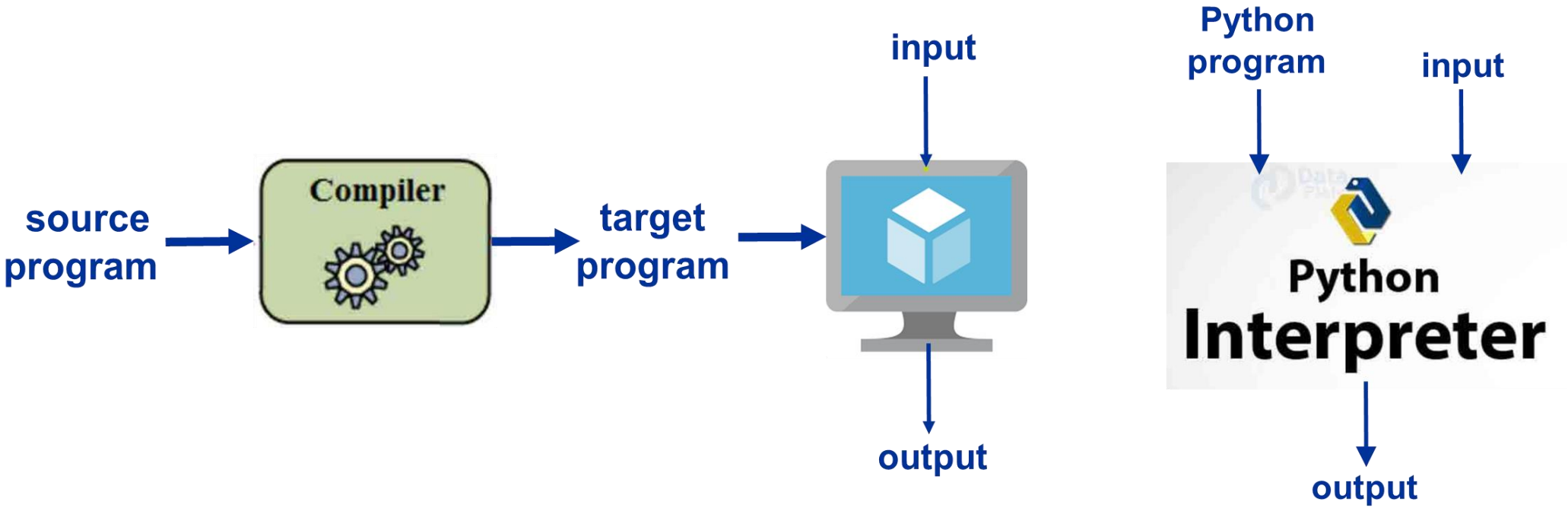
```
push 3.push 4.add.minus
```

target :
Stack Language

```
push n    add    minus
```

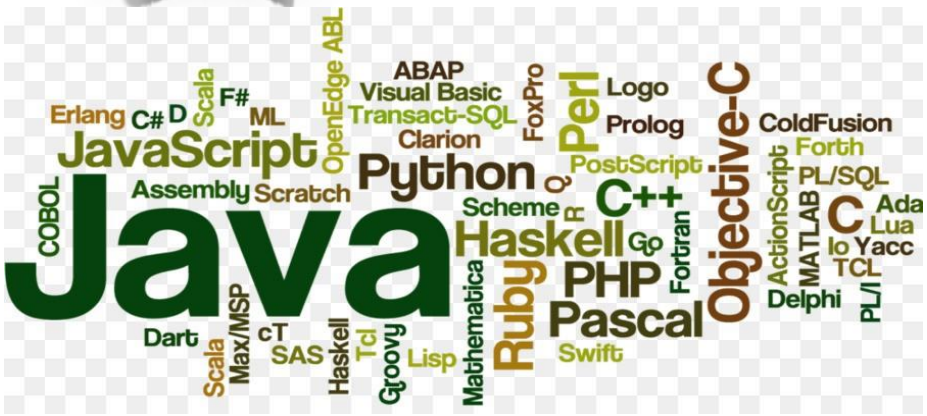
Language Translation (7/7)

- 컴파일러 vs. 인터프리터

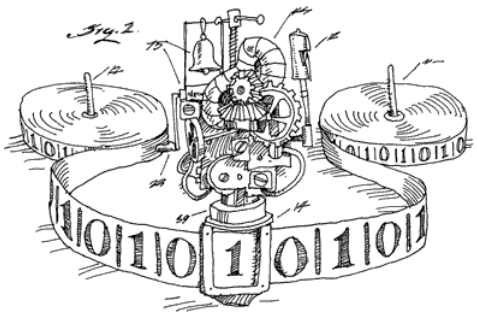


Programming Languages

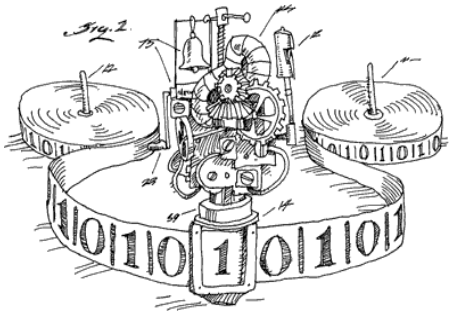
Alonzo Church
1903-1995



Alan Turing
1912-1954



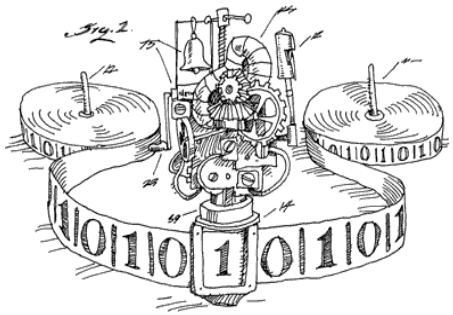
문제 해결 방법 (1/2)



- 자전거 사이의 폭의 변화
- 변화하는 폭을 왕복하는 파리의 비행거리
- 무한수열의 합

- 파리가 비행한 총 시간을 계산 = 자전거가 충돌할 때까지의 시간
- 1시간 동안 파리가 이동한 비행거리

문제 해결 방법 (2/2)



```
    a ← 0
    b ← read
L : a ← a + b
    b ← b + 1
    write a
    goto L
```

$$\sum_{i=b}^{n} i$$

```
sum(range(b,n))
```


LAMBDA
CALCULUS
EST. 1936

≡

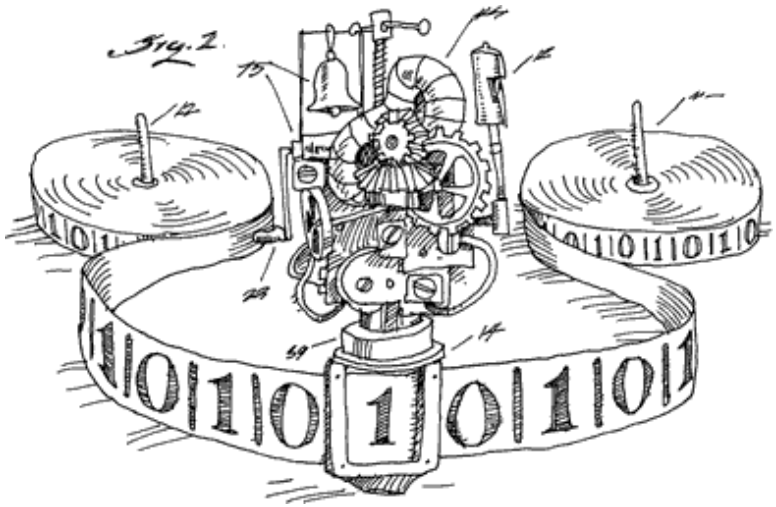


Fig. 1
1 0 1 1 0 1 0 1 1 0 1

Programming Languages for AI

- LISP, Prolog
- Python – TensorFlow, Pytorch
- R or C ..?

LISP

- 1960년 존 매카시(MIT) 개발
- 기본자료형을 기호형태로 나타낸 리스트(list)와 트리(tree)를 사용 : 복잡한 자료구조를 쉽게 처리가 가능
- 람다 계산식 계열(함수형) 프로그래밍 언어
- 대화식으로 구성된 인터프리터 방식 언어

```
(+ 2 3) → 5
(* 4 5 2) → 40
(* 2 (+ 2 4)) → 12
(< 3 5) → T
(< 1 2.5 3) → T
```

```
(DEFUN square (x) (* x x))
(square 5) → 25

(DEFUN sum (x y) (+ x y))
(sum 12 4) → 16
```

텐서플로우(Tensorflow)

- 2015년 구글에서 머신러닝과 딥러닝을 위해 개발해서 공개
- 데이터를 의미하는 텐서(tensor)와 데이터플로우 그래프를 따라 연산이 수행되는 형태
- 머신러닝과 딥러닝을 지원하는 여러 가지 SW가 있지만, 텐서플로가 가장 널리 이용되는 라이브러리
- 필기체 글씨나 숫자의 인식, 음성인식, 영상인식, 자연어 처리 등 다양한 분야에서 텐서플로우가 사용되고 있음

파이토치(PyTorch)

- 2016년 페이스북에서 머신러닝을 위해 개발해서 공개
- Python 기반의 오픈소스 머신러닝 라이브러리
- Python 라이브러리(Numpy, Scipy, Cython)와 높은 호환성
- 학습 및 추론 속도가 빠르고 다루기 쉬움
- Torch를 기반으로 하며, 자연어 처리 등에 주로 사용됨