데이터 사이언스 스쿨 4기

컴퓨터 공학 및 프로그래밍 기초 개념

2017. 01. 09 변영효

목차

- 2진수, 8진수, 16진수 이해
- bit, byte의 이해
- Hardware / Operating System / Software
- 프로그래밍 및 프로그래밍 언어의 이해
- 알고리즘 및 Big O 노테이션의 이해
- Software Engineering
- 커리어 로드맵
- 3주간 배울 내용 정리

10진법(DECIMAL)

- 각 자리수는 0 - 9로 구성
- 7, 23, 129 등의 십진수 파헤치기
- 7 = 7 * 10 ^ 0 (a^b = a의 b승 e.g 2^3 = 8)
- 23 = 2 * 10 ^ 1 + 3 * 10 ^ 0
- 129 = 1 * 10 ^ 2 + 2 * 10 ^ 1 + 9 * 10 ^ 0
- 각 자리수의 수와 10의 거듭제곱으로 표현되는 수가 곱해진 후 합 계산
- 10진법 - 각 자리수가 10의 거듭제곱으로 표현
- 연습문제 - 네자리 십진 수 중 가장 큰 수는?

2진법(BINARY)

- 각 자리수는 0, 1로 구성
- 2진수는 각 자리수가 2의 거듭 제곱으로 표현
- 즉, 해당 자리수의 값이 1을 초과하면 수의 한자리가 증가
- 1, 101, 1011
- 1 = 1 * 2 ^ 0 = 1
- 101 = 1 * 2 ^ 2 + 0 * 2 ^ 1 + 1 * 2 ^ 0 = 5
- 1011 = 1 * 2 ^ 3 + 0 * 2 ^ 2 + 1 * 2 ^ 1 + 1 * 2 ^ 0 = 11
- 연습문제) 2진수 1010101은 10진수로 얼마인가?
- 연습문제2) 7자리 2진수 중 최대로 큰 수는 10진수로 얼마인가?

8진법(OCTAL), 16진법(HEXADECIMAL)

- 8진수의 각 자리수는 0 - 7로 구성됨
- 16진수의 각 자리수는 0 - 9, A(10), B(11), C(12), D(13), E(14), F(15)로 구성
- 8진수 715 = 7 * 8 ^ 2 + 1 * 8 ^ 1 + 5 * 8 ^ 0 = 461
- 16진수 2AE = 2 * 16 ^ 2 + A * 16 ^ 1 + E * 16 ^ 0 = 686

- 연습문제) 8진수 7777은 각각 10진수, 2진수로 얼마인가?
- 연습문제2) 16진수 FFFF는 각각 10진수, 2진수로 얼마인가?

진법 변환

- 2진수 1110 10진수 변환
- 10진수 45 2진수 변환
- 2진수 10110011 16진수 변환
- 16진수 A9E1 2진수 변환
- 10진수 33 16진수 변환
- 16진수 FFA 10진수 변환

- 연습문제) 2진수 10100101은 16진수로 얼마인가?
- 연습문제2) 10진수 345는 2진수로 얼마인가?

2의 거듭제곱 근사치 계산 (POWER OF 2 APPROXIMATION)

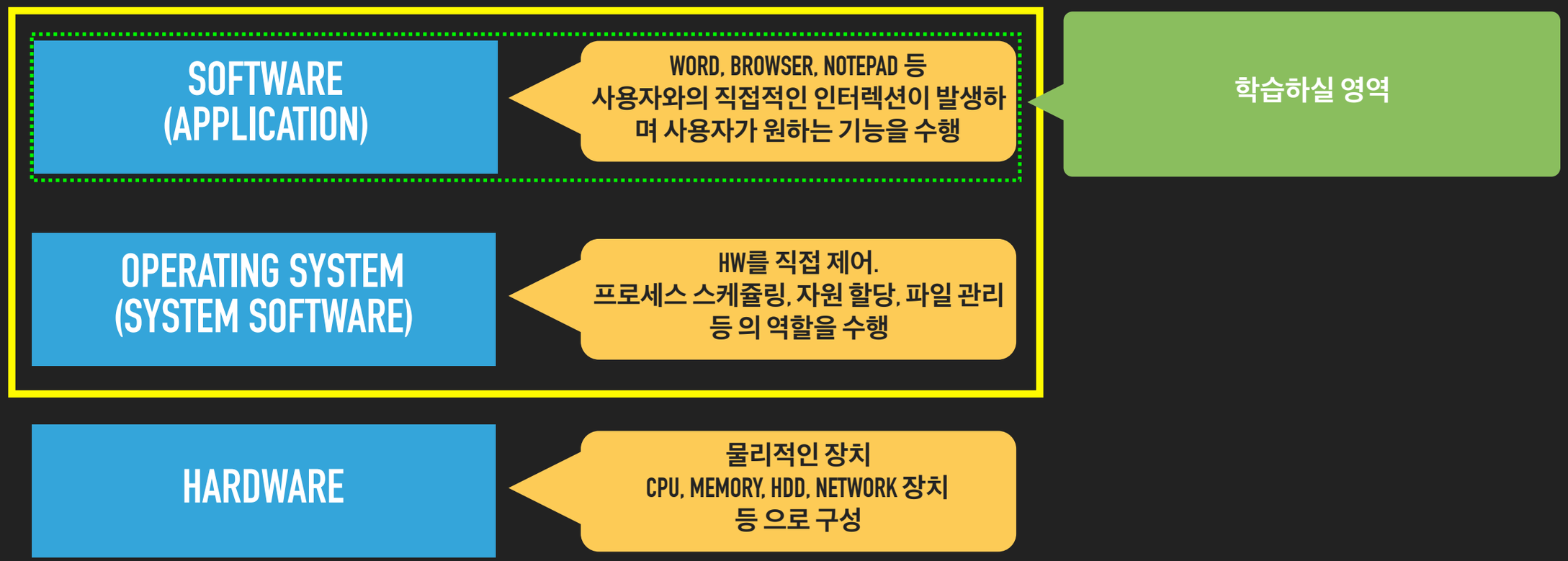
- 일반적으로 2 ^ 10 = 10 ^ 3으로 계산
- 2 ^ 25의 근사값
- 2 ^ 25 = 2 ^ 20 * 2 ^ 5 = (10 ^ 3) ^ 2 * 32 = 3천2백만\
- 연습문제 2 ^ 32의 근사값을 구하시오
- 연습문제 2 ^ 47의 근사값을 구하시오

BIT, BYTE

- bit = 정보 저장의 기본 단위로 0 또는 1의 값을 가짐 (2진법)
- byte = 8bits 즉, 256가지의 정보 저장 가능
- kilo bytes = 10 ^ 3 bytes = 2 ^ 10 bytes
- mega bytes = 10 ^ 6 bytes = 2 ^ 20 bytes
- giga bytes = 10 ^ 9 bytes = 2 ^ 30 bytes
- tera bytes = 10 ^ 12 bytes = 2 ^ 40 bytes
- peta bytes = 10 ^ 15 bytes = 2 ^ 50 bytes
- exa bytes = 10 ^ 18 bytes = 2 ^ 60 bytes
- zeta bytes = 10 ^ 21 bytes = 2 ^ 70 bytes
- yotta bytes =. 10 ^ 24 bytes = 2 ^ 80 bytes
- 2013년, 전 세계의 모든 데이터가 4.4 zeta bytes로 평가됨

http://www.northeastern.edu/levelblog/2016/05/13/how-much-data-produced-every-day/

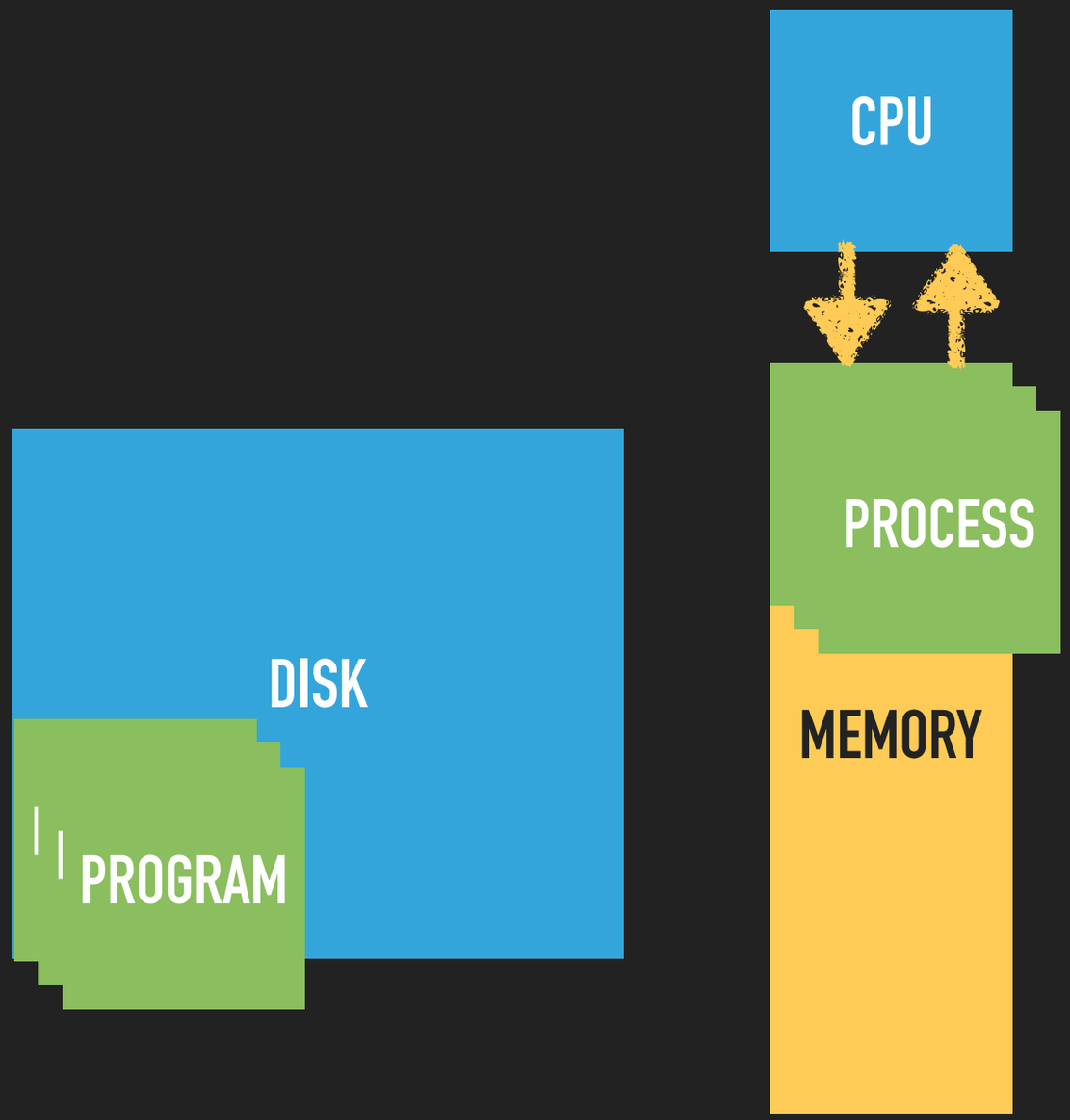
HW / OS / SW


SOFTWARE
(APPLICATION)
WORD, BROWSER, NOTEPAD 등
사용자와의 직접적인 인터렉션이 발생하며 사용자가 원하는 기능을 수행
OPERATING SYSTEM
(SYSTEM SOFTWARE)
HW를 직접 제어.
프로세스 스케쥴링, 자원 할당, 파일 관리 등 의 역할을 수행
HARDWARE
물리적인 장치
CPU, MEMORY, HDD, NETWORK 장치 등 으로 구성
학습하실 영역

PROGRAM, PROGRAMING, PROGRAMER, PROCESS

- programer - 프로그램을 만드는 사람
- programing - 프로그램을 만드는 일
- program 이란?
- 컴퓨터가 실행할 수 있도록 설계된 작업들로 구성된 명령어 집합을 의미
- 일반적으로 소프트웨어를 의미함
- process 란?
- 프로그램이 컴퓨터의 메모리에 로딩되어 실행 중인 상태를 의미

PROGRAM & PROCESS



모든 명령어는 CPU상에서 실행 됨

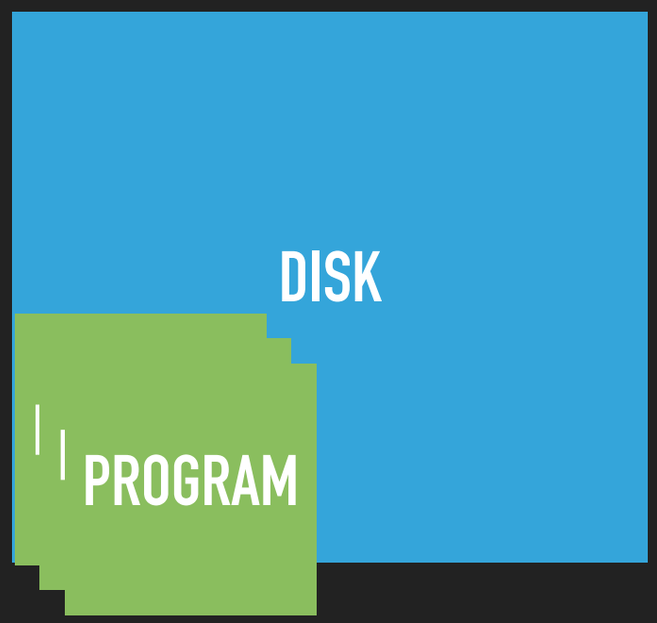
DISK에 있는 프로그램은 실행되기 전
MEMORY에 로딩되어야 함

MEMORY에 로딩되어 실행가능한 형태를
PROCESS라고 함

예시)

CHROME은 실행되기 전 파일의 형태로 DISK에 존재
사용자가 CHROME을 실행하는 순간 MEMORY에 로딩 되
며 실행중인 PROCESS가 됨

PROGRAMING



이것이 가능한 일 일까요?

프로그램은 결국 컴퓨터가 이해할 수 있어야 함.

즉, CPU가 실행 할 수 있는 명령어(2진수)로 구성

만약, 프로그램을 2진수 에디터로 열어본다면
0101001110101101010110101010000101O…
와 같이 구성되어 있음

따라서, 프로그래밍이라는 것은 01010111011등의 CPU 명령어를 나열하는 것에 불과함

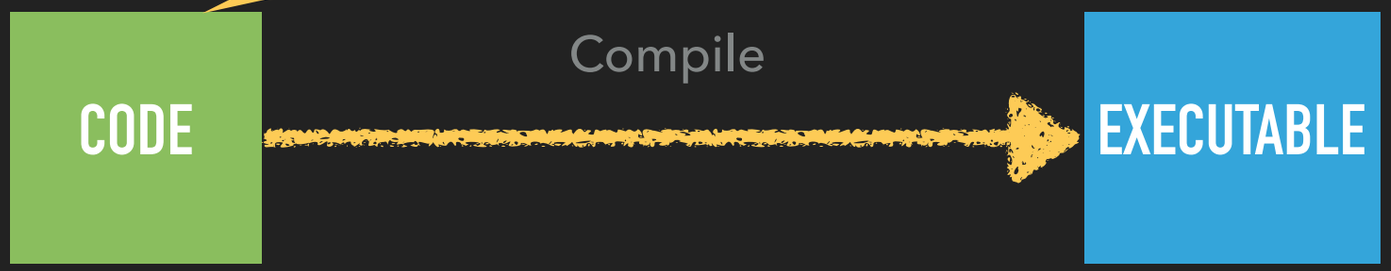
PROGRAMING 언어

결론적으로 불가능에 가까움..

1. CPU 명령어는 LOW LEVEL의 작업이기 때문에 모든 작업을 LOW LEVEL로 처리하는 것은 불가능에 가까움
2. 이진수 명령어를 이용하여 프로그래밍 하는 것은 실수를 유발하거나 엄청난 스트레스를 야기함
3. 각 CPU 아키텍쳐마다 명령어가 다르기 때문에, 각 아키텍쳐마다 독립적으로 프로그래밍 작업이 필요함

따라서, 인간이 이해할 수 있는 수준의 언어로
프로그램을 만들고, 그것을 기계가 이해할 수 있도록 변환

E.G) C/C++ JAVA PYTHON C# GO JAVASCRIPT...



프로그래밍 언어로 코드를 작성

PROGRAMING 언어 2가지 타입

Compile

CODE

compiler

EXECUTABLE

or

CODE

compiler

INTERMEDIATE CODE

JIT compiler

EXECUTABLE

C/C++ JAVA C# GO

Interpreting

CODE

Interpreter가 line별로 실행

PYTHON JAVASCRIPT

PROGRAMING 언어 2가지 타입

Compile

1. 빠른 실행
2. 언어 습득의 어려움
3. 낮은 생산성

Interpreting

1. 느린 실행 (상대적)
2. 언어 습득이 용이
3. 높은 생산성

일반적으로 위와 같은 특징을 갖기는 하나,
최근에는 서로의 장점을 채용하는 추세

하지만, 아직까지는 속도나 성능이 중요시 되는 임베디드, 시스템 소프트웨어는 C/C++로 작성됨

주로 스타트업이에서 혹은 개발 주기가 빨라 높은 생산성을 요구하는 작업에는 PYTHON, JAVASCRIPT, RUBY등의 언어를 사용

알고리즘(ALGORITHM)

- 특정한 문제를 계산하거나 해결하기 위해 필요한 일련의 절차들의 집합을 의미
- 기술시간에 배운 순서도와 동일한 개념
- 정립된 알고리즘(abstract)을 코드화(concrete)한 결과가 프로그램
- 결국 프로그래밍을 한다는 것은 정립된 알고리즘을 프로그래밍 언어를 이용하여 코드로 표현하는 행위를 의미
- 누가 / 언제 / 어떻게 수행해도 동일한 결과를 도출하여야 함

알고리즘(ALGORITHM)

- 덧셈
- 나눗셈
- 숫자 리스트에서 최소값 찾기
- 숫자 리스트 순서대로 정렬
- 두개의 숫자 리스트를 작은 순서대로 하나의 리스트로 병합

알고리즘(ALGORITHM)

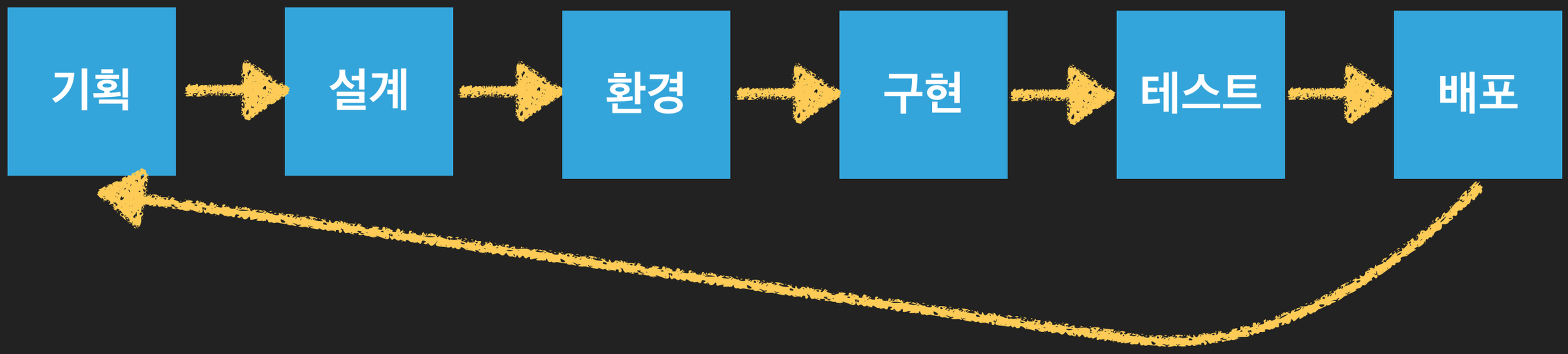
- 덧셈
- 숫자 리스트에서 최소값 찾기
- 정렬된 숫자 리스트에서 특정 값 찾기
- 숫자 리스트 순서대로 정렬
- 두개의 숫자 리스트를 작은 순서대로 하나의 리스트로 병합

BIG O NOTATION

- 알고리즘의 performance 혹은 complexity를 표현하기 위해 사용
- 보통 worst boundary를 의미하며, 알고리즘의 실행 시간이나 메모리 사용량을 표시
- 앞에서 검토했던 알고리즘의 Big O 표현 알아보기

SOFTWARE ENGINEERING

단순히 코딩만을 의미하지 않음

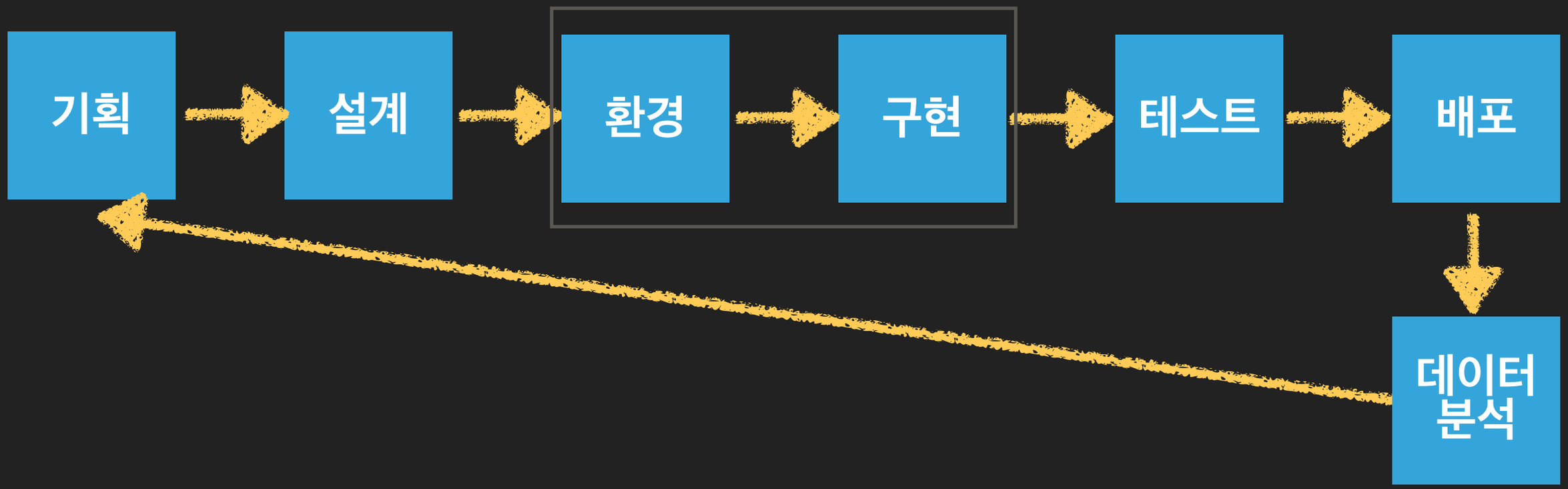


위의 모든 과정에 참여하며 코딩(구현)은 일부에 지나지 않음

QA Engineer라고 하는 테스트 전문 엔지니어도 존재

SOFTWARE ENGINEERING

최근에는 Cloud와 여러 개발툴의 발달로 환경구성과 구현에 소요되던 시간이 많이 줄어듬



줄어든 시간 만큼 개발자가 데이터 분석을 통한 마케팅(growth hacking), QA도 담당하게 됨

SOFTWARE ENGINEER

- Front-end engineer
- Back-end engineer
- Full-stack engineer
- System software engineer
- Game software engineer
- 등등등...

DATA ENGINEER & DATA SCIENTIST

- 보통 back-end engineer가 함께 하는 경우가 많음
- 이유는 database를 설계하고 data를 실제 관리하기 때문

- 무수히 많은 위와 같은 작업의 반복(iteration)
- 결과는 paper, product, dashboard 등 다양한 형태로 존재

3달간 학습 하실 내용

PYTHON
언어

RDB
NOSQL

AWS에서 진행