

RaspberryPi

# 알고리즘으로 배우는 Python 과정

2022. 2. 15

정 준 수 Ph.D

# 과정 목표

프로그래밍 언어 Python를 활용하여 SW 알고리즘의 구현기술 습득

1. Python 언어특성과 기본 문법 익히기
2. 머신러닝에 필요한 Data Type 변환
3. 객체지향 프로그램 구현과 함수형 언어
4. Python 라이브러리를 이용한 고급 알고리즘 구현
5. Tree Algorithm와 Graph 구조 등을 이용한 다양한 머신러닝 알고리즘 응용 사례 구현

## 첫째 날

- 언어특성 및 기본 문법 익히기
- 언어사용 환경설정, 도구활용 방법소개
- print, 사칙연산
- data type
- 변수
- 알고리즘 구현 연습문제 풀이 및 실습

## 둘째 날

- 파이썬의 다양한 자료구조 익히기
- 함수 설명
- 함수 활용 알고리즘 연습문제
- List 와 Tuple
- For-loop 과 Range 함수
- Dictionary 자료 구조
- Binary Search 알고리즘 구현

## 세째 날

- 다양한 함수 소개 및 활용
- String, String methods
- 조건문, set, while문
- 객체지향 프로그램 구현
- 유클리드 호제법 알고리즘

## 네째 날

- 파이썬의 라이브러리를 이용한 고급 알고리즘 구현
- 정규식 기초
- Functional 프로그램 구현
- List Comprehension
- Big-O notation
- Memoization 알고리즘 구현
- Graph 구조 구현

# Software란?

[https://en.wikipedia.org/wiki/John\\_Tukey](https://en.wikipedia.org/wiki/John_Tukey)

# Object-oriented programming

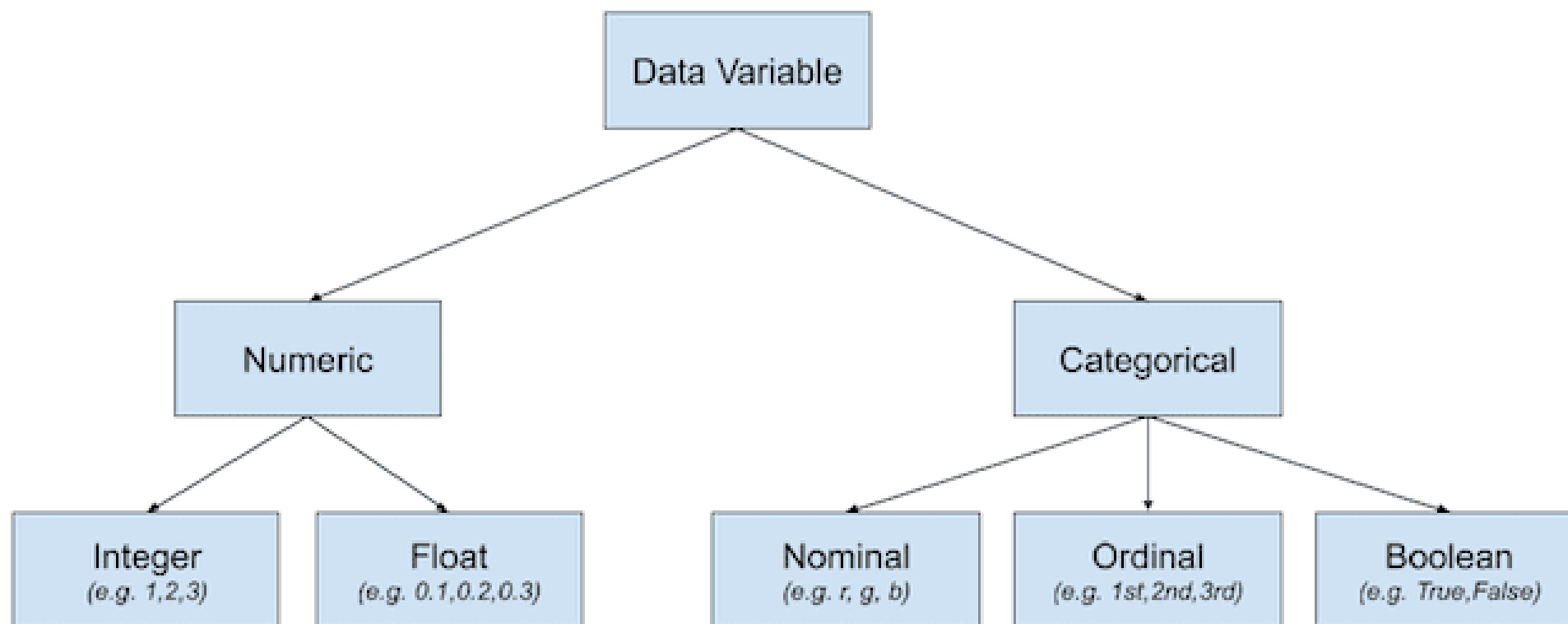
[https://en.wikipedia.org/wiki/Object-oriented\\_programming](https://en.wikipedia.org/wiki/Object-oriented_programming)



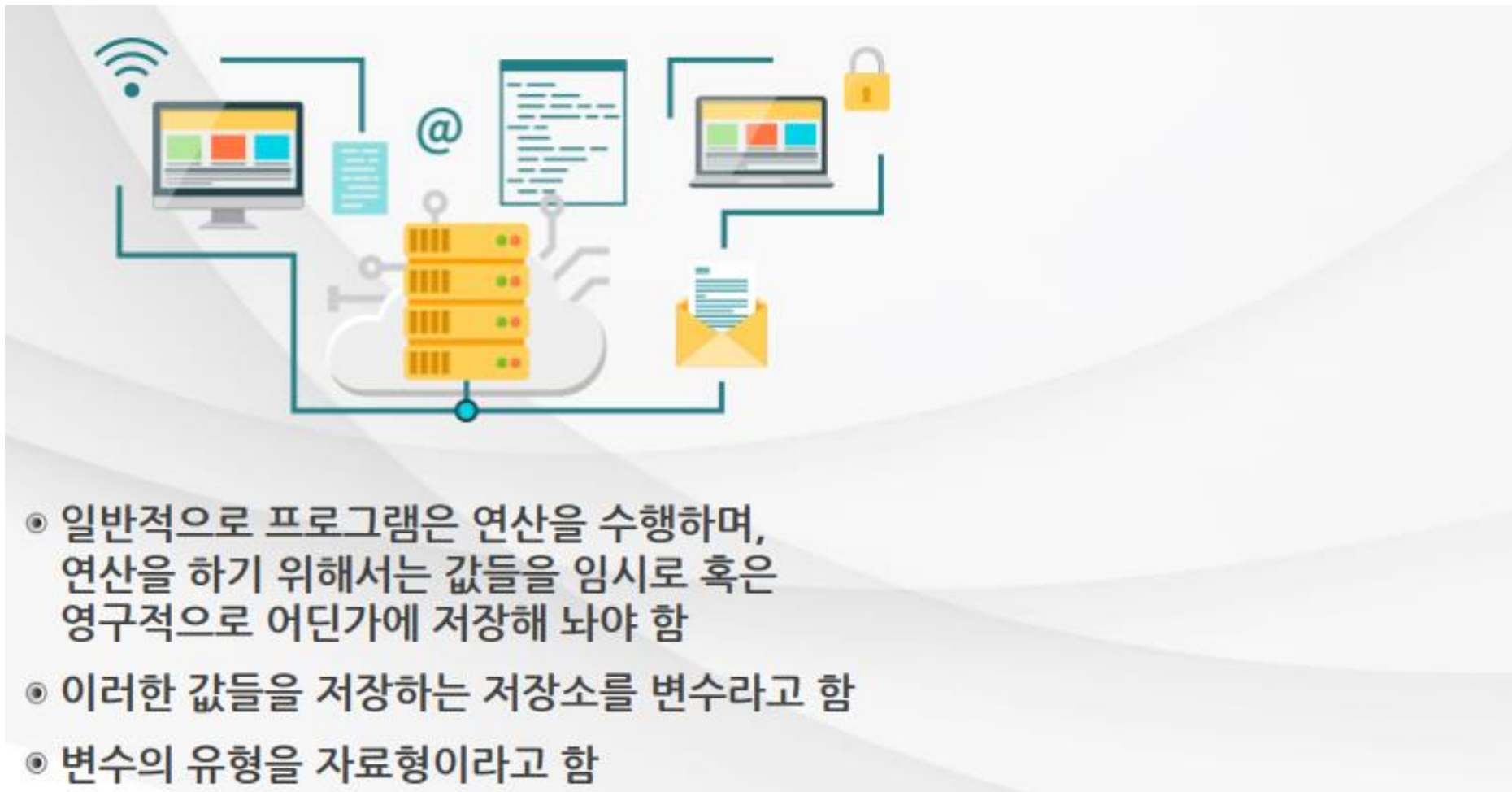
# Why Python?

# Data 변수 타입 분류

## Overview of Data Variable Types



# Python 자료형이란?



# Python 변수명

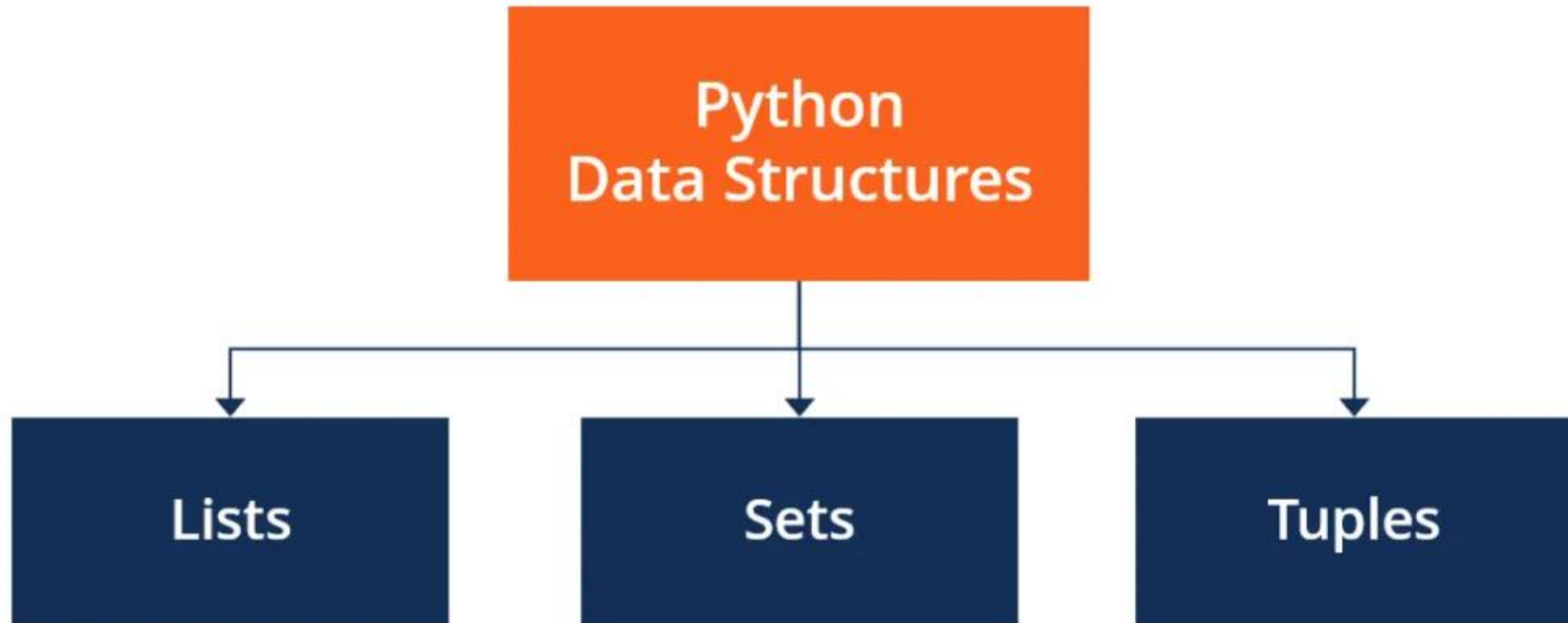
- 문자, 숫자, 밑줄(\_)로 구성  
숫자는 처음에 나올 수 없음
- 대소문자 구분
- 예약어 사용 불가

```
>>> friend = 10
>>> Friend = 1
>>> friend
10
>>> Friend
1
```

and, as, assert, break, class, continue, def, del,  
elif, else, except, is, finally, for, from, global, if,  
import, in, is, lambda, nonlocal, not, or, pass,  
raise, return, try, while, with, yield

# Python 자료의 종류

- 수치 - int float complex 등
- 문자형
- 리스트 - 리스트는 쉽게 값들의 나열
- 세트 - 집합의 개념으로 순서가 없음
- 튜플 - tuple은 리스트와 유사하나, 읽기 전용임
- 딕셔너리 - 키와 값의 쌍으로 이루어져 있음
- 부울(bool) - 참(True)과 거짓(False)을 나타내는 자료형



# 함수의 정의

- 함수의 선언은 def로 시작하고 콜론(:)으로 끝남
- 함수의 시작과 끝은 코드의 들여쓰기로 구분
- 시작과 끝을 명시해 줄 필요가 없음

## 함수 선언 문법

```
def <함수명>(인수1, 인수2, ... 인수N):  
    <구문>  
    return <반환값>
```

## 간단한 함수 선언해 보기

입력 받은 2개의 인수를 서로 곱한 값을 리턴함

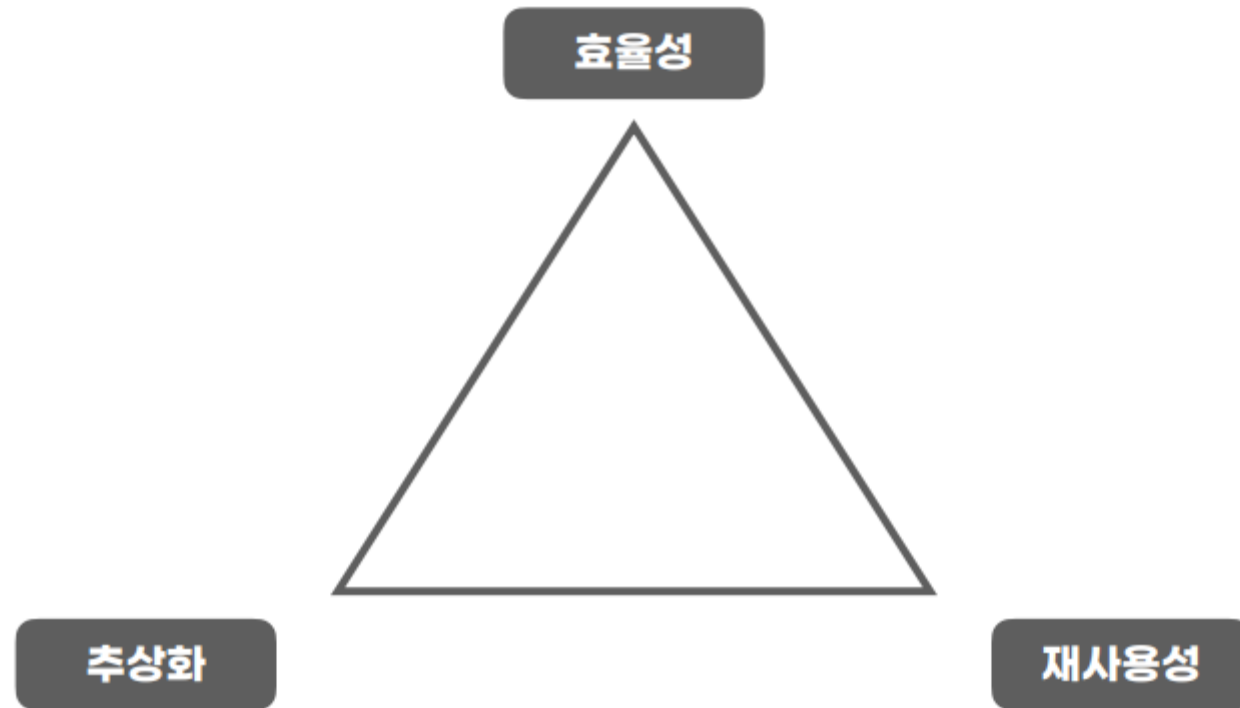
```
def Times(a, b):  
    return a*b  
  
>>> Times(10, 10)  
100
```

## <Python의 유용한 함수 예제>

[https://github.com/JSJeong-me/KOSA-Python\\_Algorithm](https://github.com/JSJeong-me/KOSA-Python_Algorithm)

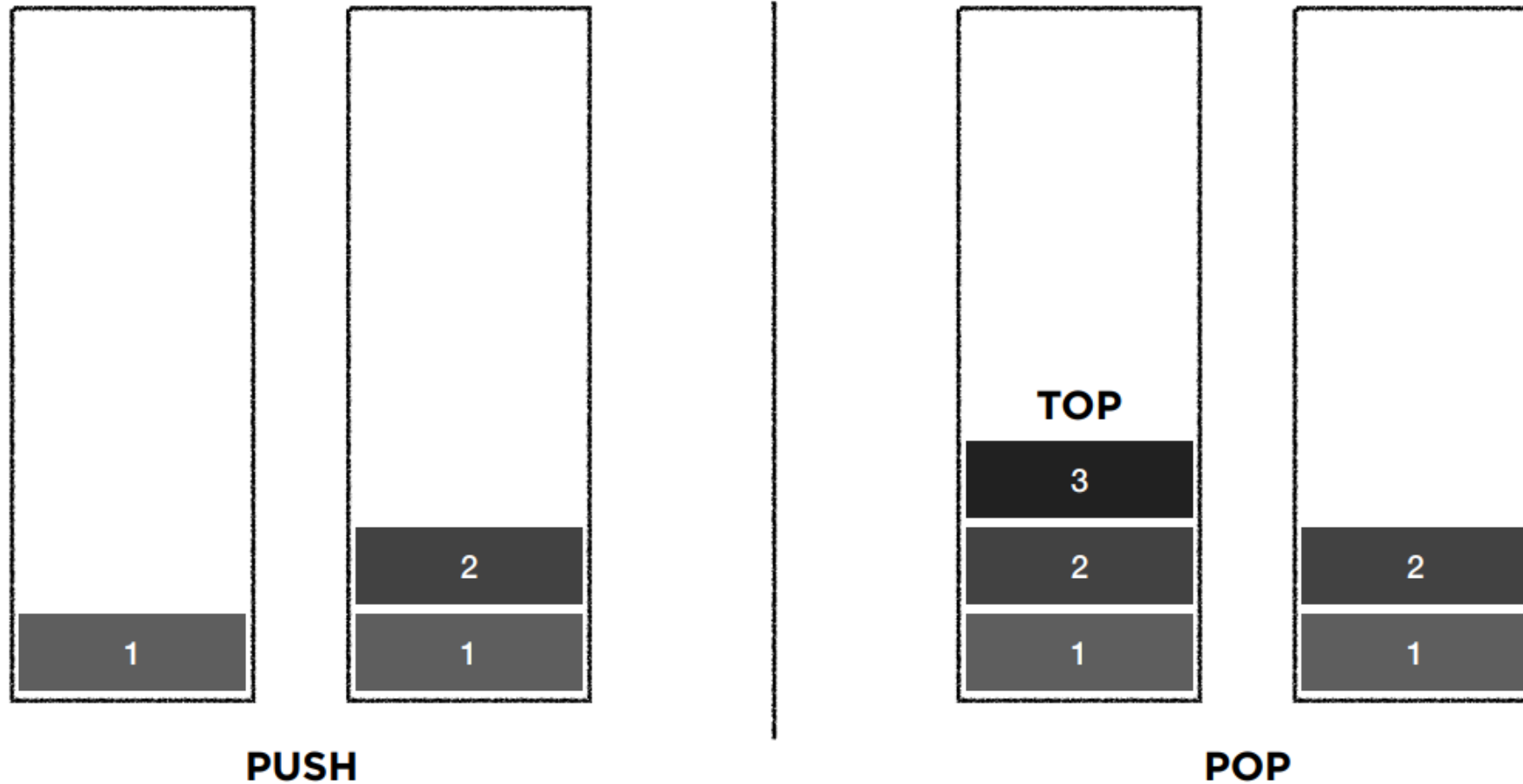


# 자료구조와 알고리즘

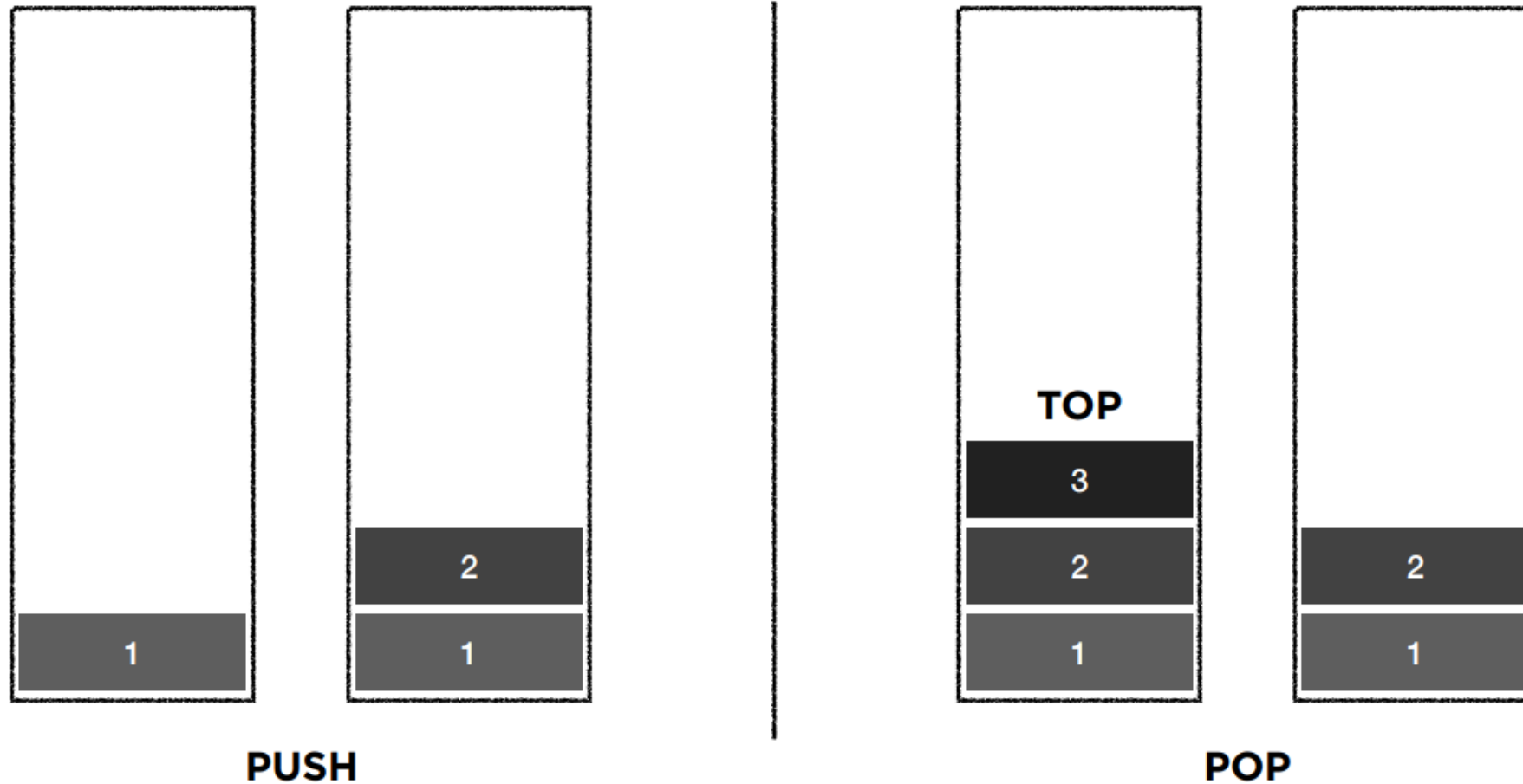


자료를 저장하는 방법론, 규칙

# Stack: FILO(First In, Last Out)



# Queue: FIFO(First In, Fast Out)





Suppose we have **13 cards** and **7 cards** have been **sorted** (with **6 card** unsorted):

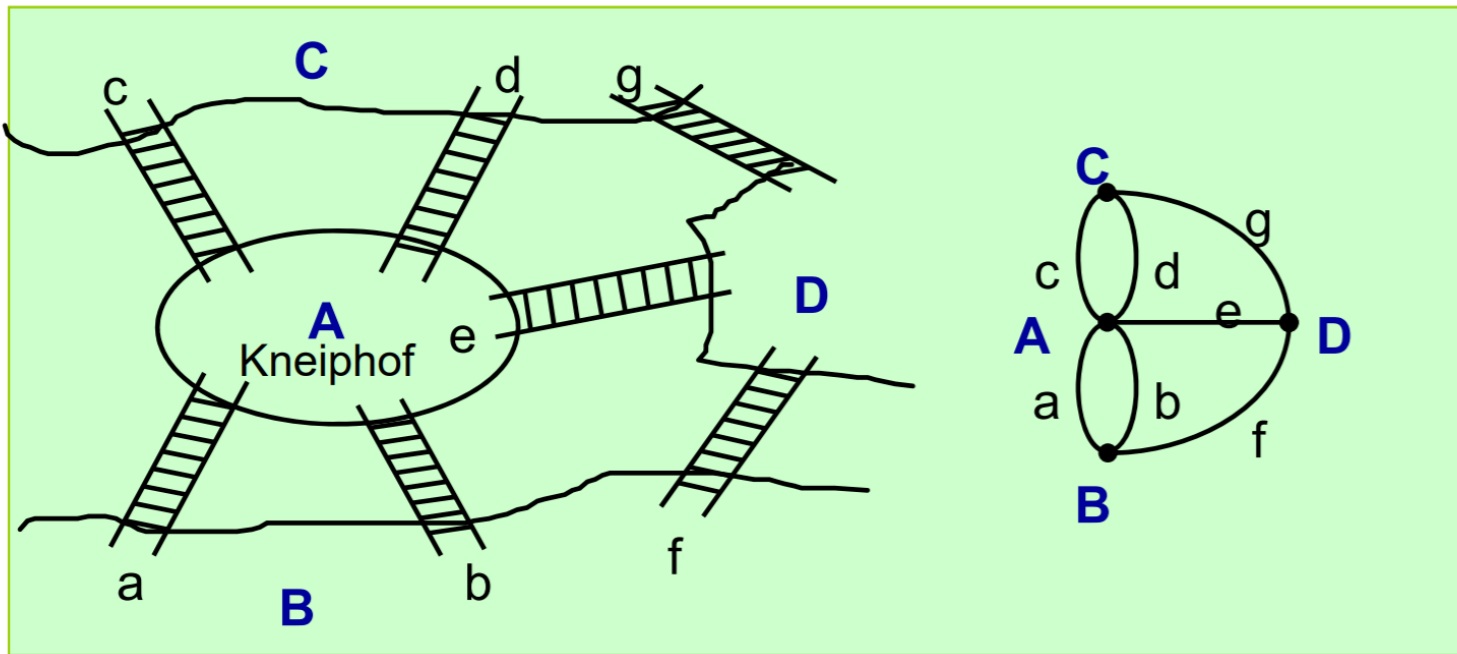
A 2 3 4 5 6 7 K 10 J 8 9 Q |  
 | +-----+ Sorted

## <실습 예제>

<http://www.mathcs.emory.edu/~cheung/Courses/171/Syllabus/7-Sort/insertion-sort.html>

# Graph: 관계를 위한 자료구조

오래된 그래프 문제로 다음과 Königsberg 다리 문제가 있다. 이 문제는 다음과 같은 지형이 있을 때 임의의 한 곳(A,B,C,D)에서 출발하여 a부터 f까지 “모든 다리를 한번씩 건널 수 있는가?” 하는 문제이다. 자료구조의 그래프는 이러한 문제를 컴퓨터에 표현하고 알고리즘을 개발하는 분야이다.



Königsberg 다리 문제

# Graph: 관계를 위한 자료구조

- 그래프의 수학적 정의

그래프 :  $G = (V, E)$  이고,  $V, E$ 는 다음과 같다.

$V(G)$  : 정점(set of vertices)

$E(G)$  : 간선(set of edges), 정점을 연결하는 선,  $V \times V$ 의 부분집합

무방향 그래프(undirected graph) – 예를 들면 쌍방향통행이 가능한 도로의 지도이다.

– 정점을 연결하는 선에 방향이 없다(undirected, unordered).

즉  $(v_i, v_j) = (v_j, v_i)$ 이다.

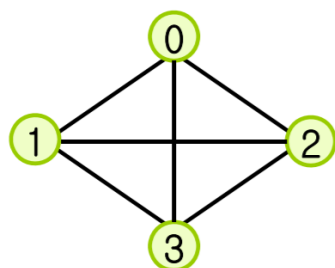
방향 그래프(directed graph) – 예를 들면 일방통행만 있는 도로의 지도이다.

– 정점을 연결하는 선에 방향이 있다(directed, ordered).

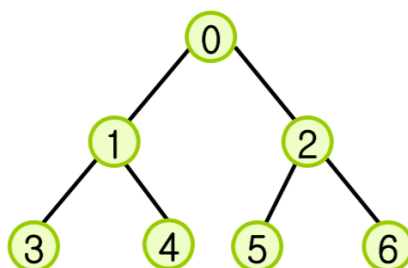
즉  $\langle v_i, v_j \rangle \neq \langle v_j, v_i \rangle$

# Graph: 수학적 표현

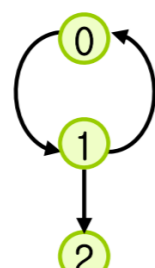
- 그래프의 예와 수학적 표현 : 그래프를 표현하는 방법은 여러 가지이다. 아래 방법은 그림으로 그리는 그래프의 모습과 수학적 기호로 표현하는 방법이다.



$G_1$



$G_2$



$G_3$

예 1) 그래프의 수학적 표현

그래프  $G_1$

$$V(G_1) = \{0, 1, 2, 3\}$$

$$E(G_1) = \{(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)\}$$

그래프  $G_2$

$$V(G_2) = \{0, 1, 2, 3, 4, 5, 6\}$$

$$E(G_2) = \{(0, 1), (0, 2), (1, 3), (1, 4), (2, 5), (2, 6)\}$$

그래프  $G_3$

$$V(G_3) = \{0, 1, 2\}$$

$$E(G_3) = \{<0, 1>, <1, 0>, <1, 2>\}$$



# Graph: 수학적 표현 (Adjacency matrix)

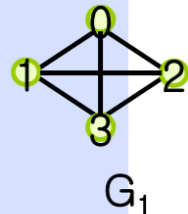
- 그래프  $G = (V, E)$ ,  $|V| = n (\geq 1)$ 일 때 그래프를 이차원 행렬에 다음과 같이 저장하는 방법이다.  
 $adj\_mat[i][j] =$

1 : if  $(v_i, v_j)$ 가 인접할 때(adjacent)

0 : 인접하지 않을 경우

-필요한 기억장소의 크기 = 공간복잡도(space complexity) :  $S(n) = n^2$

- 무방향 그래프에서는 행렬이 대각선을 중심으로 대칭(symmetric)이다.



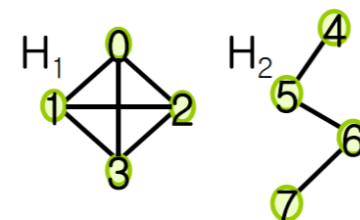
|   | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 2 | 1 | 1 | 0 | 1 |
| 3 | 1 | 1 | 1 | 0 |



|   | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 2 | 0 | 0 | 0 |

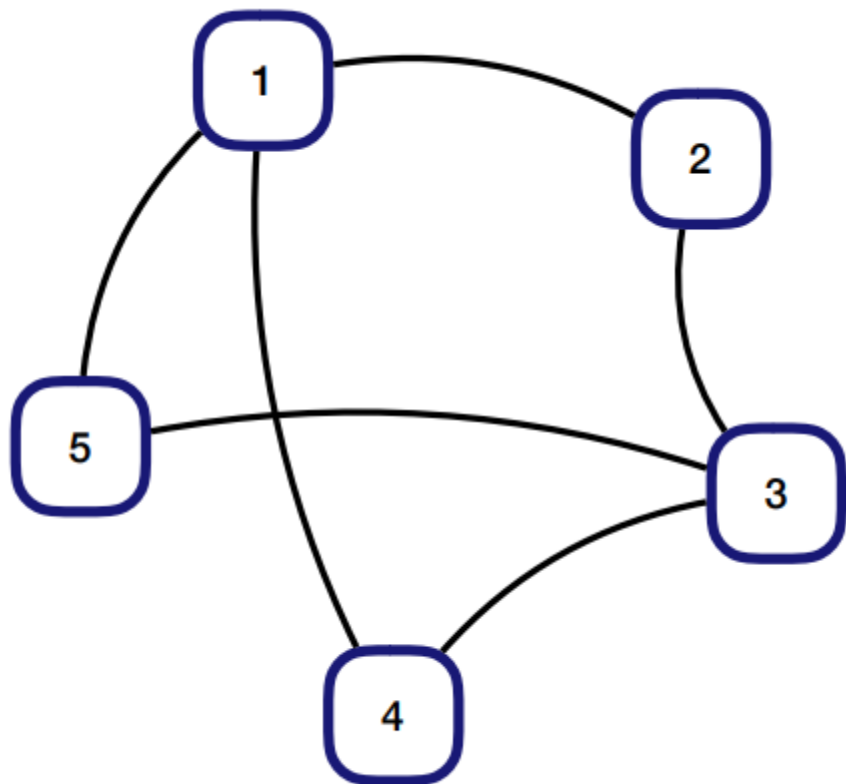
인접 행렬로 표현된 그래프  $G_1$ ,  $G_3$ , and  $G_4$

$G_4$



|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 3 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

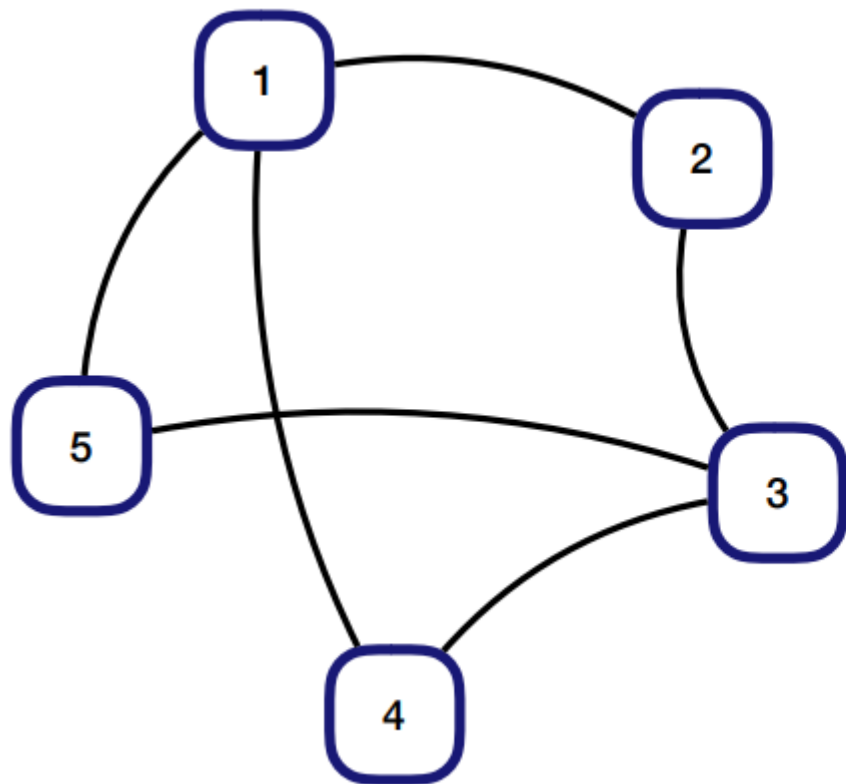
# Graph: 관계를 위한 자료구조



노드(Node)와 간선(Edge)으로 구성  
너와 나의 연결고리

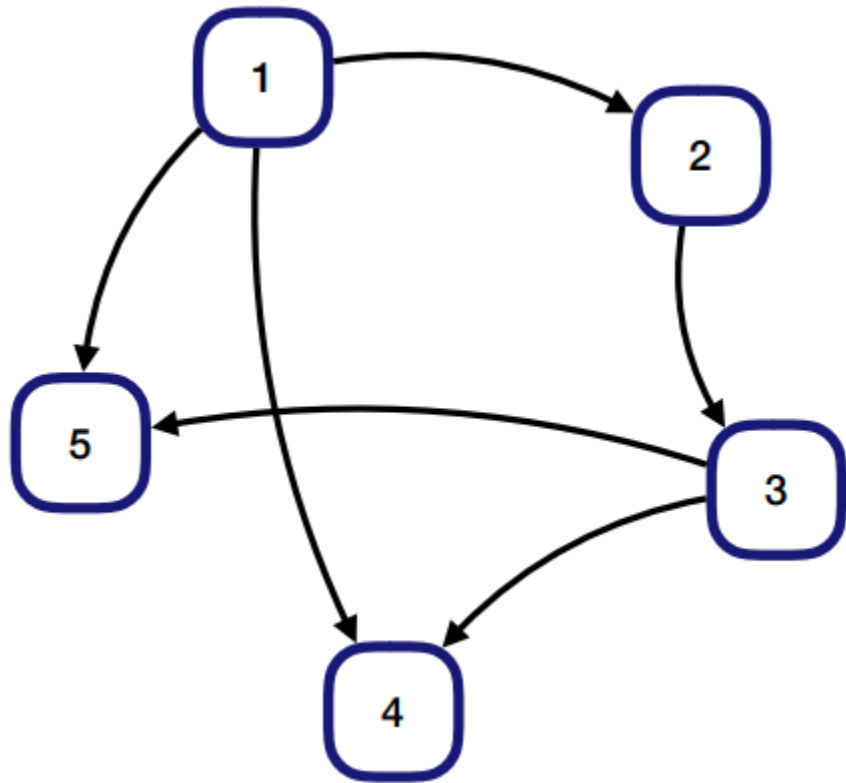
들어오는 간선 수를 **indegree**  
나가는 간선 수를 **outdegree**

## Graph: 저장하는 방법 / 인접 행렬



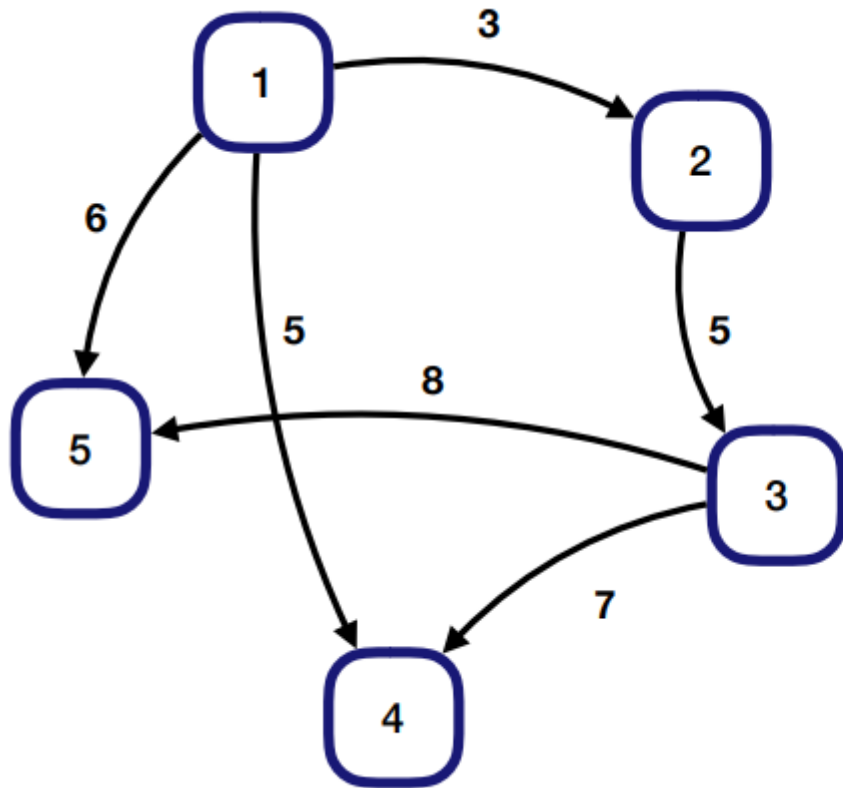
|   | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 1 |
| 2 | 1 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 | 1 | 1 |
| 4 | 1 | 0 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 | 0 | 0 |

## Graph: 방향을 추가하면?



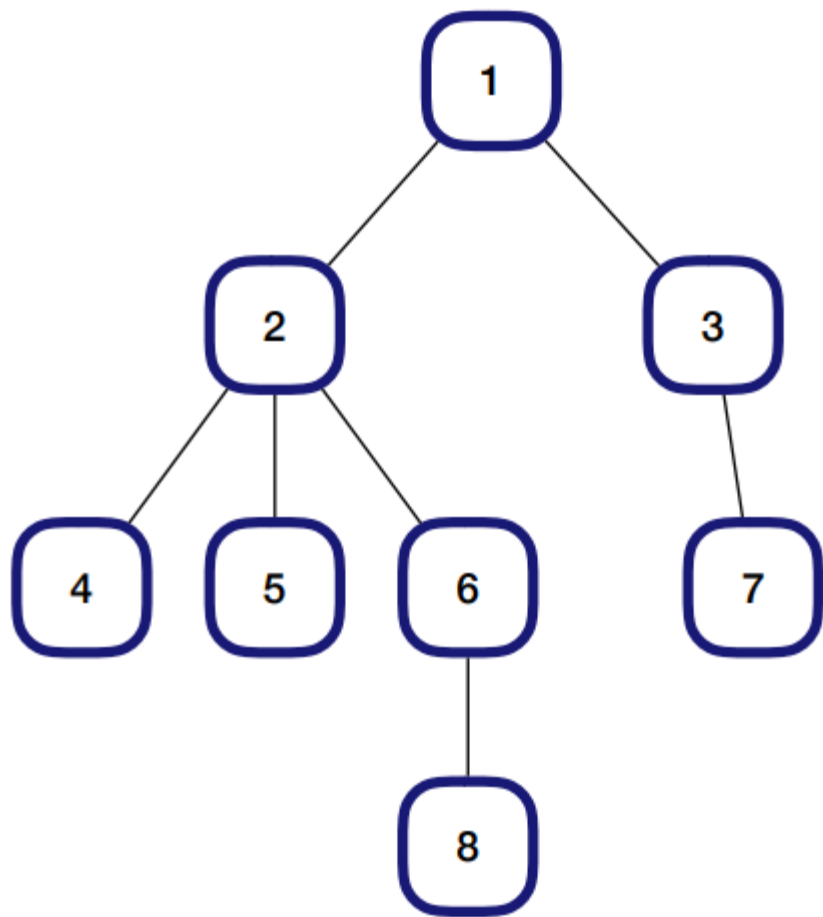
|   | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 |

## Graph: 가중치를 추가하면



|   | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 3 | 0 | 5 | 6 |
| 2 | 0 | 0 | 5 | 0 | 0 |
| 3 | 0 | 0 | 0 | 7 | 8 |
| 4 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 |

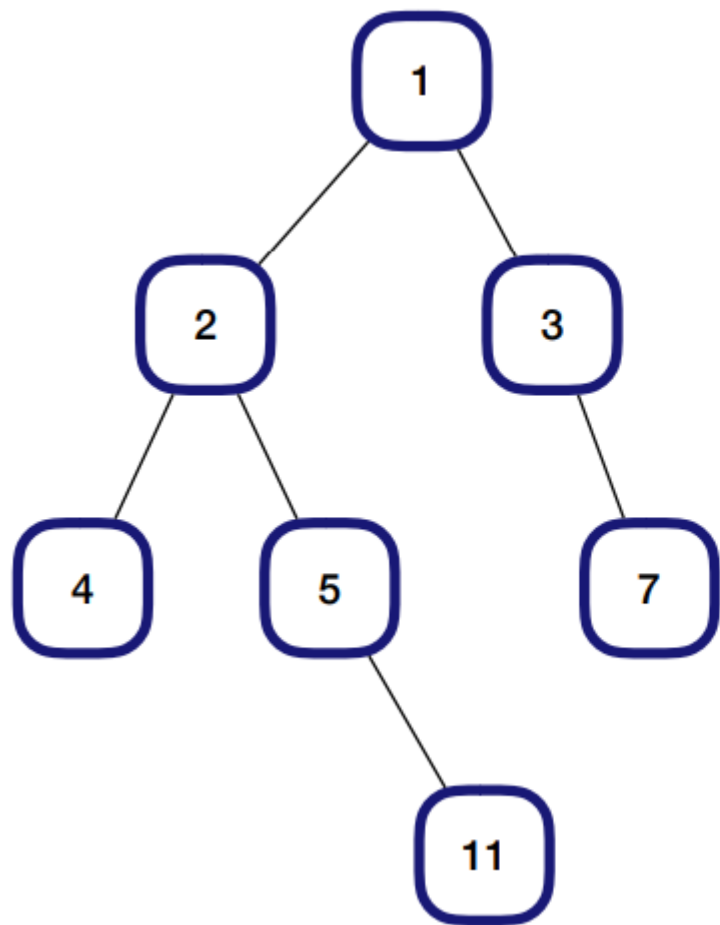
# Tree: 특수한 구조를 가진 그래프



그래프이기에 똑같이 **노드(Node)**와 **간선(Edge)**으로 구성

상위/하위 관계를 나눌 수 있음

# Binary Tree: 이진 트리



부모 노드가 X이면, 왼쪽 노드가  $2X$  오른쪽 노드가  $2X+1$   
Heap과 BST에 대해 알아봅시다.



크리스찬 베일

스티브 카렐

라이언 고슬링

브래드 피트



# 빅쇼트

월스트리트를 몰락인 4명의 괴짜 천재들이  
믿을 수 없는 실화



《머니볼》《블라인드 사이드》  
작가 원작



PARAMOUNT PICTURES AND REGENCY ENTERPRISES PRESENT  
A PLAN B ENTERTAINMENT PRODUCTION AN ADAM MCKAY FILM CHRISTIAN BALE  
STEVE CARELL RYAN REYNOLDS BRAD PITT "THE BIG SHORT" WITH NICHOLAS BRITELL  
SCREENPLAY BY SUSAN MATTHESON EDITOR HANK CORWYN EXECUTIVE PRODUCERS CLAYTON HARTLEY  
DIRECTOR OF PHOTOGRAPHY GARRY ACKROYD EXECUTIVE PRODUCERS LOUISE ROSNER-MEYER KEVIN MESSICK  
PRODUCED BY BRAD PITT, J.D. DEDE GARDNER, J.D. VECCHIAY KLEINER, J.D. ARNOLD WILCHMAN  
BASED UPON THE BOOK BY MICHAEL LEWIS SCREENPLAY BY CHARLES RANDOLPH AND ADAM MCKAY DIRECTED BY ADAM MCKAY

REGENCY

PLAN B

LOTTE ENTERTAINMENT

THE BIG SHORT

1월 21일 대개봉

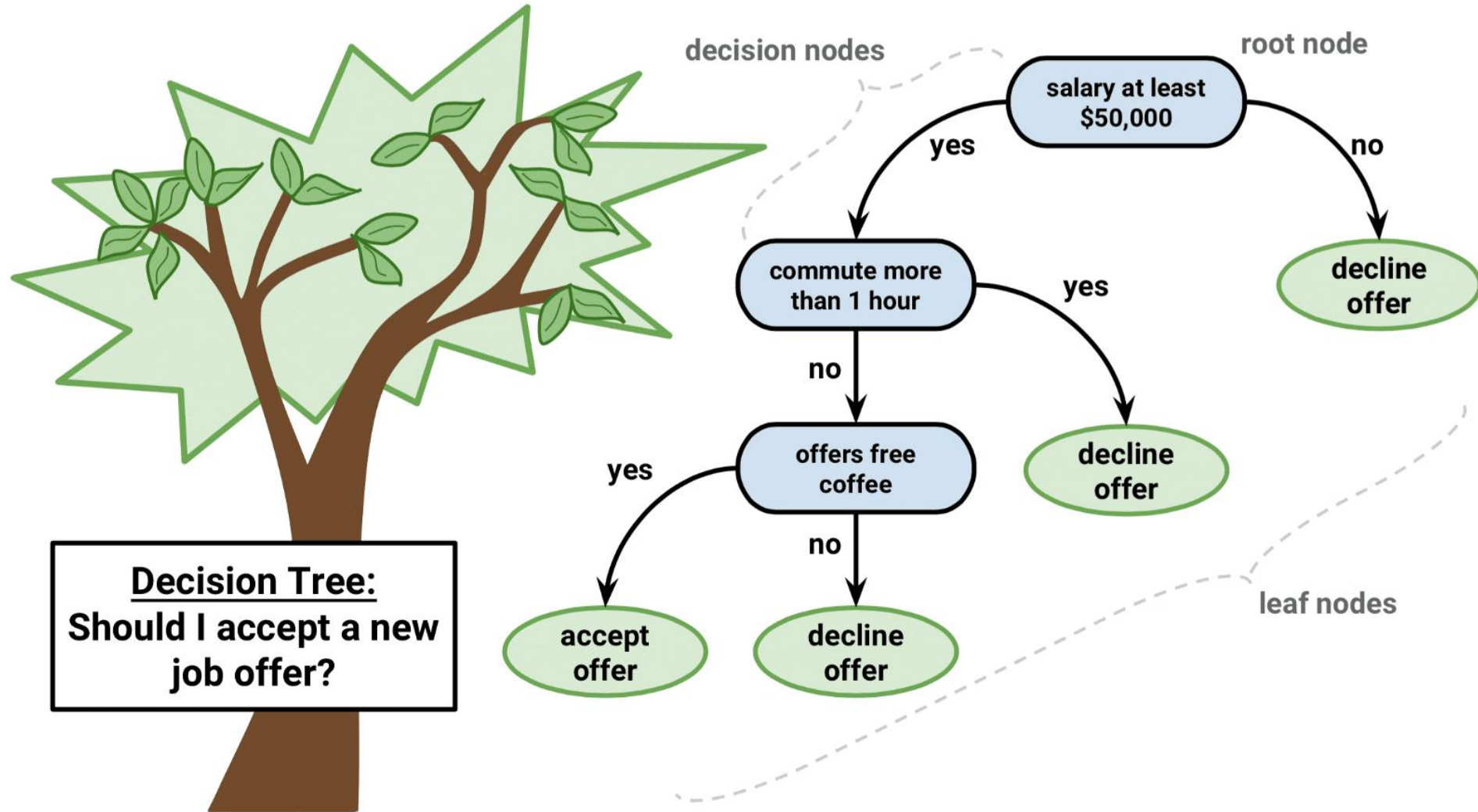
15세 관람가

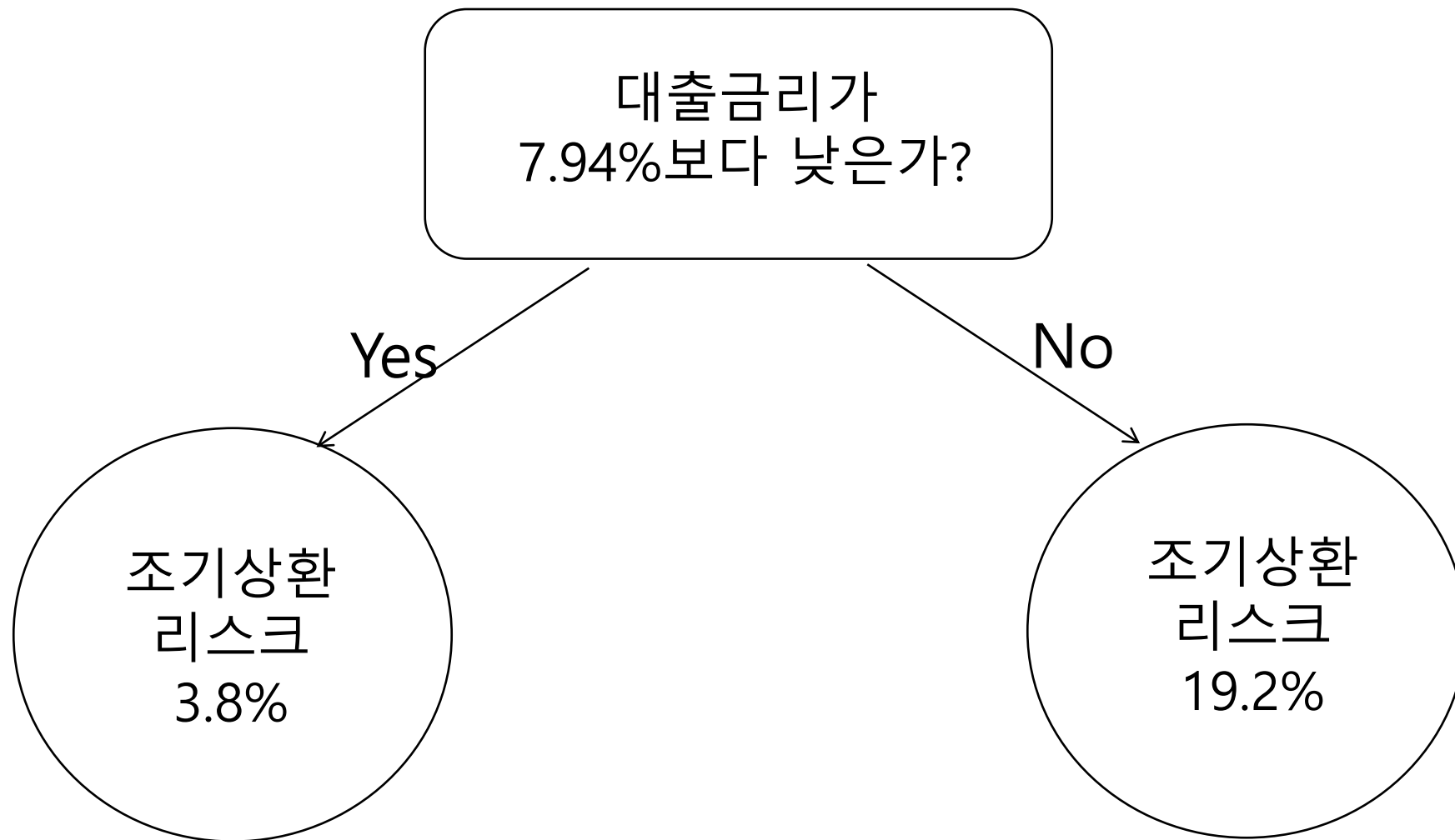


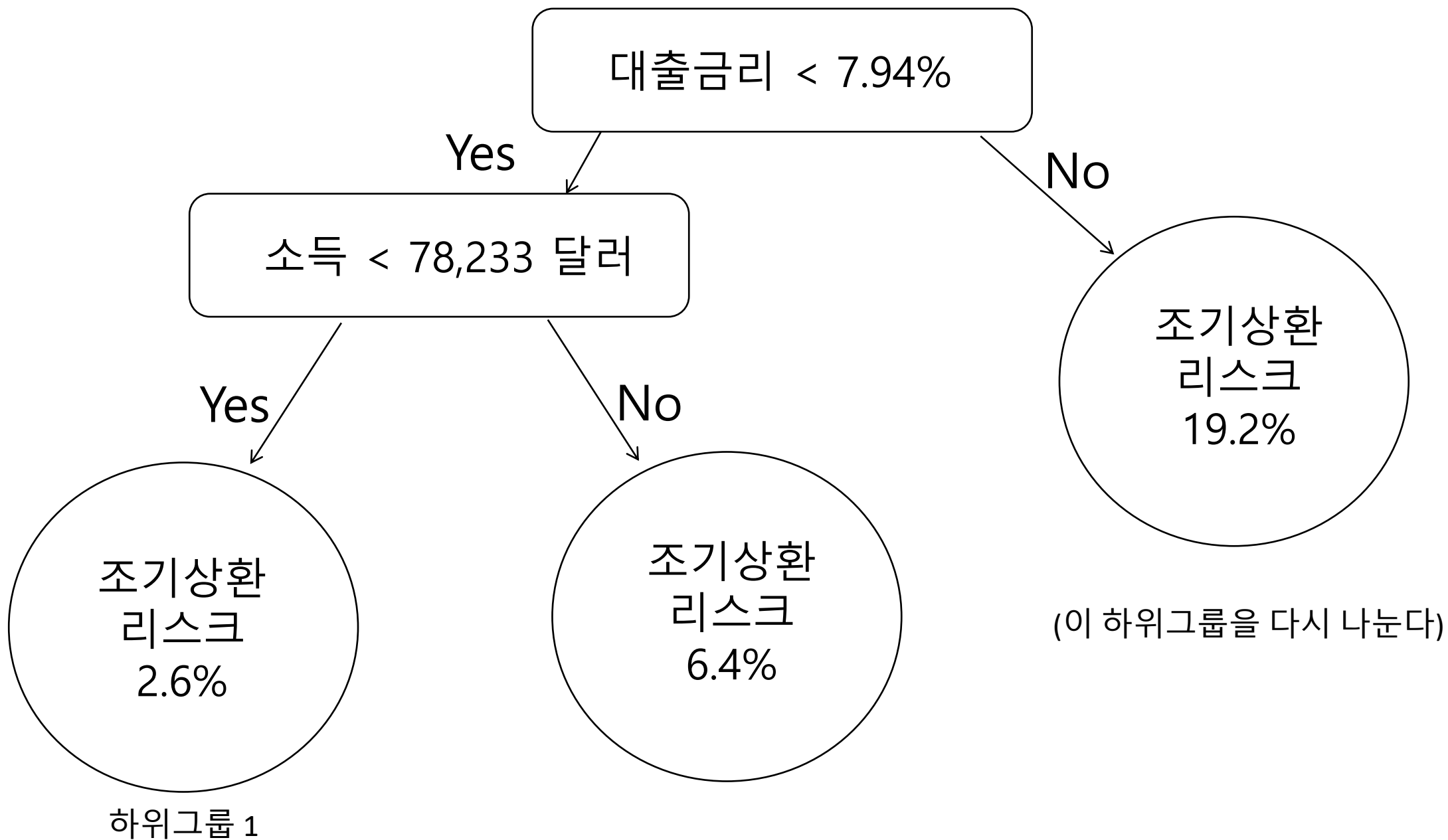
# 나무에서 돈이 자란다!

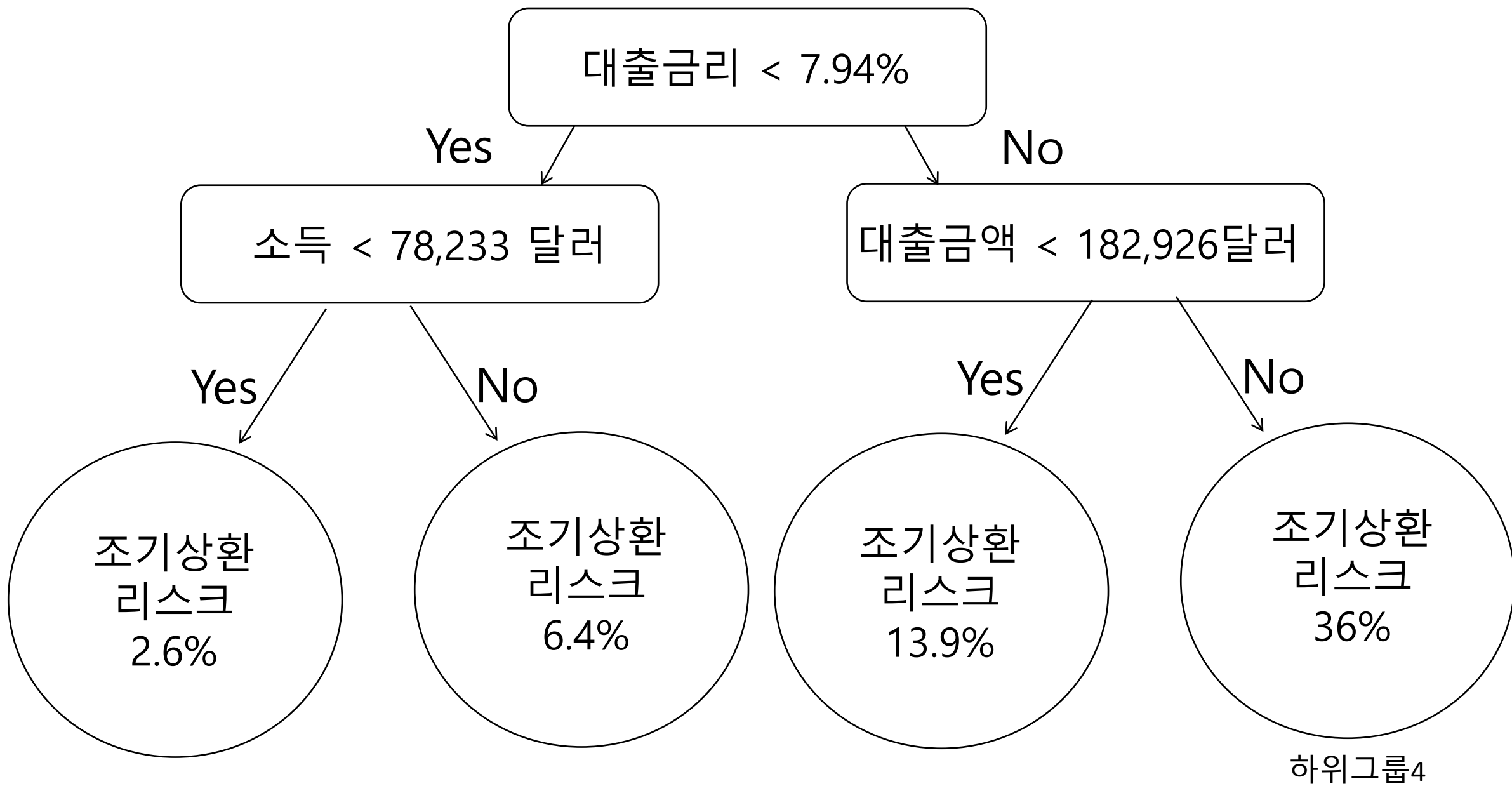
체이스 은행 예측모델은 부동산 담보대출 중에서 실제로  
조기상환된 대출 건들 중 74%를 정확하게 인식해 내어서  
부동산 담보대출 포트폴리오를 성공적으로 관리함.

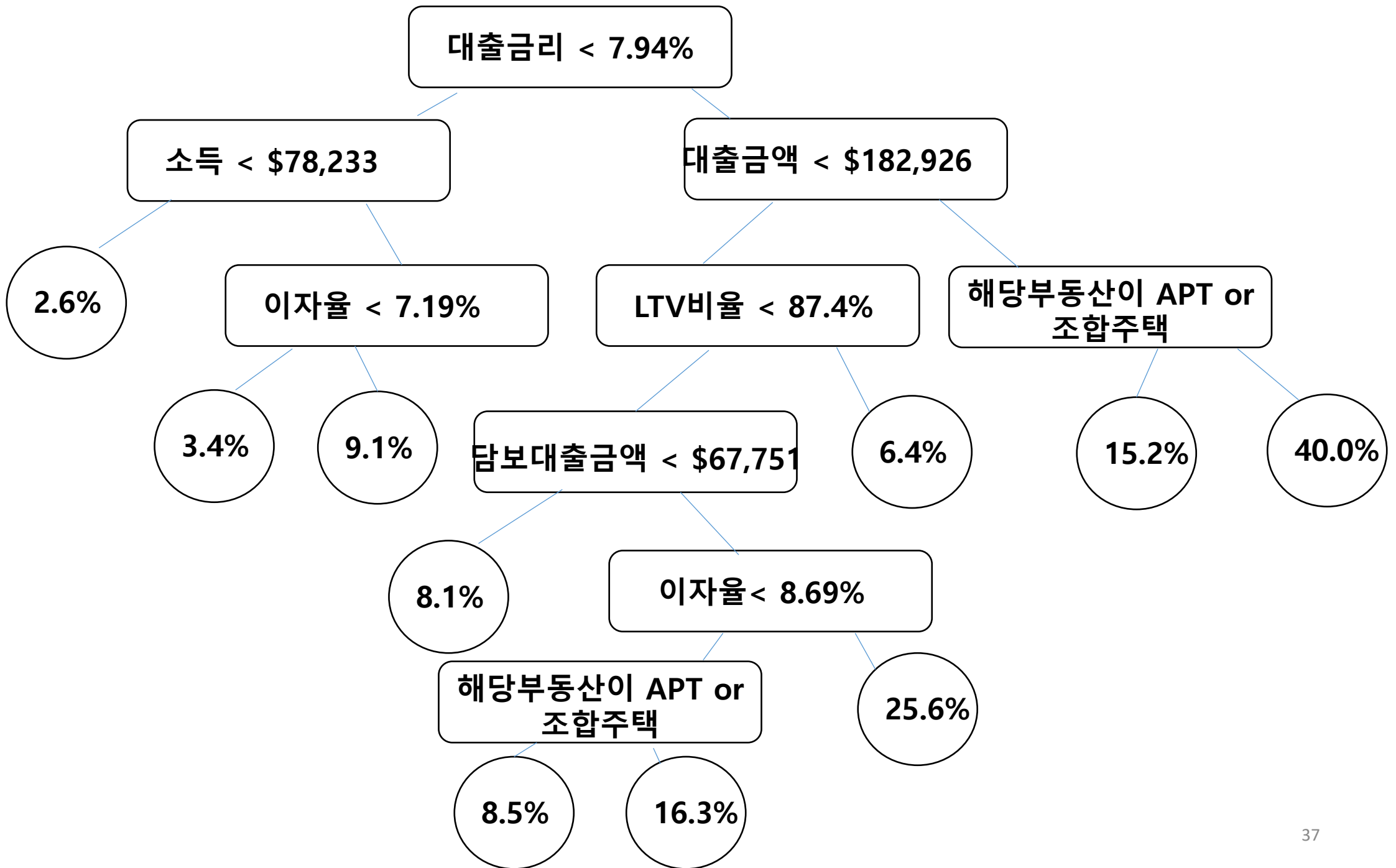
# Decision Tree











### 만약(IF):

부동산 담보대출 금액이 67,751 달러와 같거나 그보다 더 많고 182,926 달러보다 작다.

### 그리고(AND):

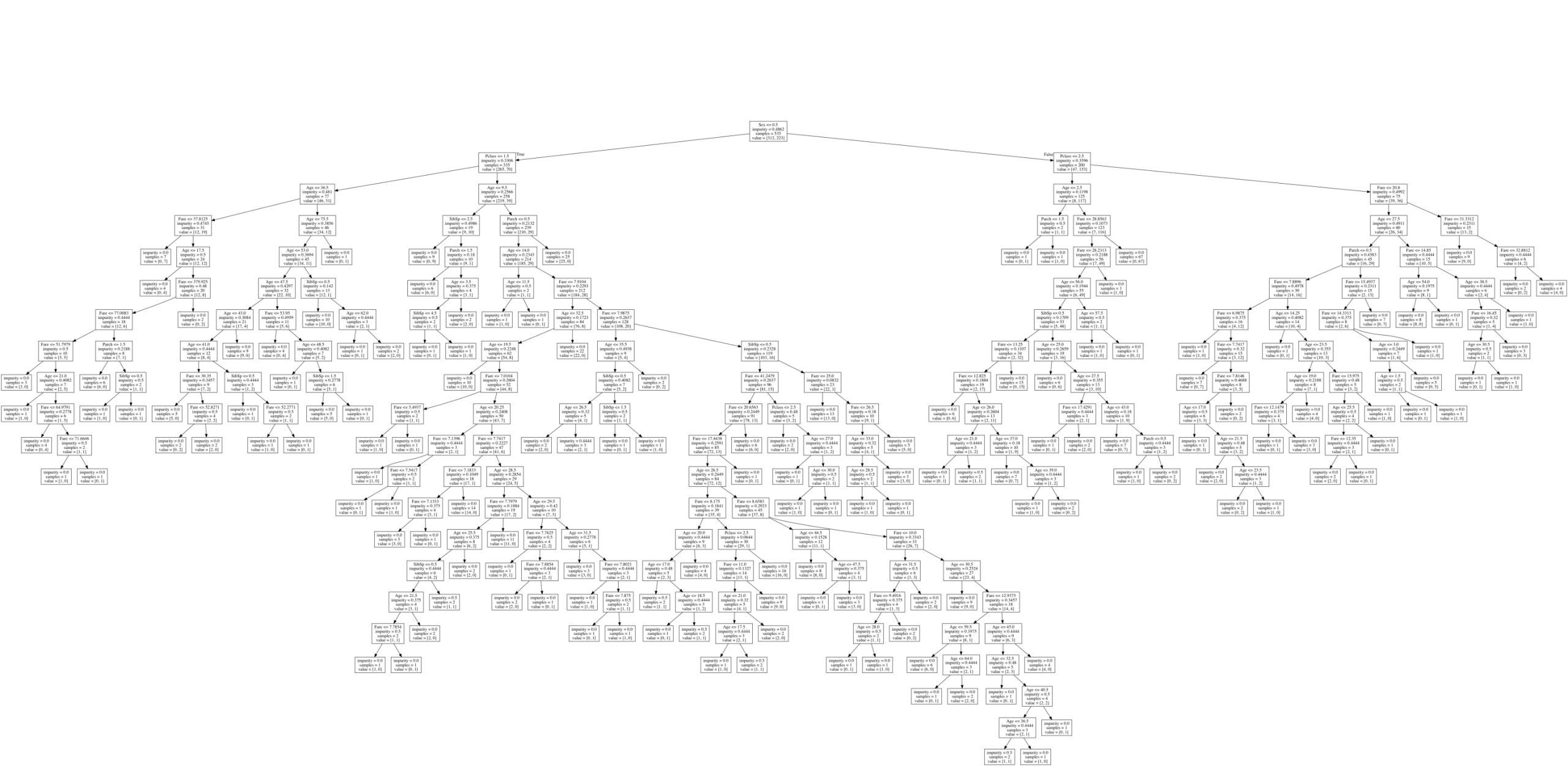
이자율이 8.69%와 같거나 그보다 더 높다.

### 그리고(AND):

부동산 자산가치 대비 대출금액의 비율이 87.4% 보다 작다.

### 그러면(THEN):

조기상환 확률은 25.6% 이다.





# 시간 복잡도: Time Complexity

연산에 따라 속도는 모두 같을까?    + 연산과 \* 연산은 같을까?

모든 연산을 Counting 할 수 있을까?    break 등의 생략은?

최악의 경우와 최선의 경우?    수열의 정렬에서 최악과 최선?

Counting에 따라 실행시간은?    서버와 컴퓨터의 속도 차?



컴퓨터 과학(Computer Science) 에서 알고리즘은 어떠한 문제를 해결하기 위한 방법이고, 어떠한 문제를 해결 하기 위한 방법은 다양하기 때문에 방법(알고리즘) 간에 효율성을 비교하기 위해 빅오(big-O) 표기법을 보통 가장 많이 사용한다.

빅오 표기법은 보통 알고리즘의 시간 복잡도와 공간 복잡도를 나타내는데 주로 사용 된다.

(시간 복잡도는 알고리즘의 시간 효율성을 의미하고, 공간 복잡도는 알고리즘의 공간(메모리) 효율성을 의미한다.)

그런데 시간과 공간 복잡도를 나타내는 방법으로는 점근 표기법이라고 해서

빅오(Big-O), 빅오메가(big-Ω), 빅세타(big-Θ) 표기법이 있다.

빅오 표기법은 알고리즘 효율성을 상한선 기준으로 표기하기 때문이다.  
(알고리즘 효율성은 값이 클수록 즉, 그래프가 위로 향할수록 비효율적임을 의미한다.)

# 시간 복잡도: Big-O Notation

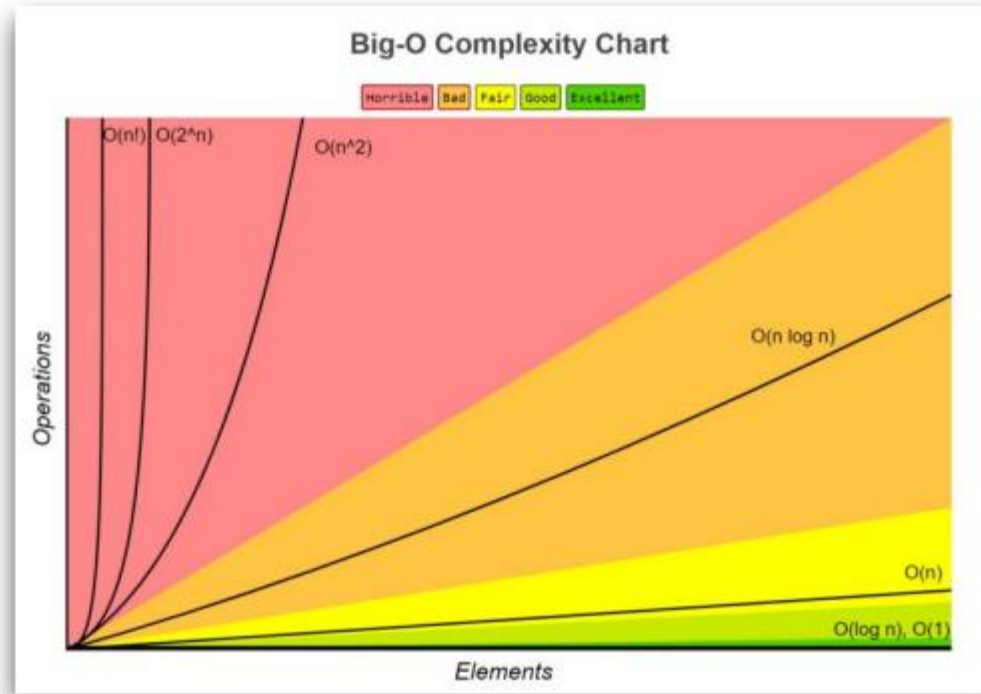
대충 계산하자 => 점근적 표기법

입력이 N일때, 연산 횟수가 최악이  $2N^2 + 4N$  이라면??

N이 무한대로 커질 때, 증가에 미치는 영향은 가장 큰 항만 필요하다!

**$O(N^2)$**

# 시간 복잡도: Big-O Notation



Slow       $O(n!) > O(2^N) > O(N^2) > O(N \log N) > O(N) > O(\sqrt{N}) > O(\log N) > O(1)$       Fast

(지수함수 > 다항함수 > 선형함수 > 로그함수 > 상수함수)

# 시간 복잡도: $O(1)$ 예시

Q. 1부터 N까지 합을 구하시오.



```
def sum_N(N):  
    return N * (N + 1) // 2
```

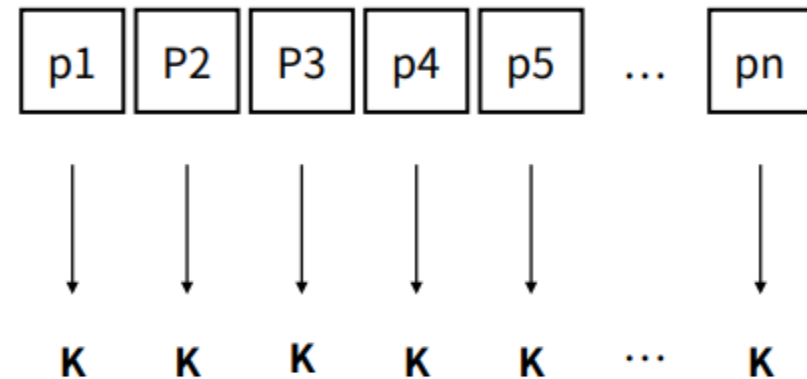
$$N * (N + 1) // 2$$

단순한 수식으로 표현 가능

# 시간 복잡도: $O(N)$ 예시

Q. 길이 N 수열에서 수 K 찾기 (sum, min, max도 유사)

```
def search(lst, N, K):  
    for i in lst:  
        if i == K : return True  
    return False
```



최악의 경우 N번을 돌려봐야함  
확률적으로는  $N/2$ 번

# 시간 복잡도: $O(\log N)$ 예시

계속 길이가 절반으로 줄어듦  $N/2/\dots/2 \simeq 1 \Rightarrow N \simeq 2^k \Rightarrow k = \log_2 N$

2라는 상수를 때면...  $O(\log_2 N) = O(\log N)$

```
def binary_search(lst, N, K):  
    lo, hi = 0, N-1  
    while lo <= hi :  
        mid = (lo + hi) // 2  
        if lst[mid] == K: return True  
        if lst[mid] > K: hi = mid-1  
        else : lo = mid+1  
    return False
```

# Big O 예제

1.  $O(1)$  : 스택에서 Push, Pop
2.  $O(\log n)$  : 이진트리
3.  $O(n)$  : for 문
4.  $O(n \log n)$  : 퀵 정렬(quick sort), 병합정렬(merge sort), 힙 정렬(heap Sort)
5.  $O(n^2)$ : 이중 for 문, 삽입정렬(insertion sort), 거품정렬(bubble sort), 선택정렬(selection sort)
6.  $O(2^n)$  : 피보나치 수열



# 유클리드호제법: 최대공약수와 최소공배수

최대공약수란 공통적인 약수 중 최댓값  
최대공약수가 1이면 서로소

최소공배수는 다음 공식으로 구할 수 있음  
공통된 배수 중 최솟값

$$LCM(A, B) = A \times B / GCD(A, B)$$

우리는 GCD만 잘 구하면 LCM은 O(1)에 구할 수 있다

# 유클리드호제법: 최대공약수와 최소공배수

혹시 유클리드 호제법 아시는 분??

$$GCD(A, B) = GCD(B, A \% B)$$

```
def gcd(a, b):  
    return b if a%b==0 else gcd(b, a%b)
```

정 준 수 / Ph.D ( jsjeong@hansung.ac.kr )

- 前) 삼성전자 연구원
- 前) 삼성의료원 (삼성생명과학연구소)
- 前) 삼성SDS (정보기술연구소)
- 現) (사)한국인공지능협회, AI, 머신러닝 강의
- 現) 한국소프트웨어산업협회, AI, 머신러닝 강의
- 現) 서울디지털재단, AI 자문위원
- 現) 한성대학교 교수(겸)
- 전문분야: Computer Vision, 머신러닝(ML), RPA
- <https://github.com/JSJeong-me/>

