

# 강의 소개 및 개론

프로그래밍2 1주차  
데이터사이언스  
곽찬희

# 강의소개 - 프로그래밍 2

- 강의 목표

- ✓ 알고리즘, 데이터 구조를 Python 으로 구현하면서, Python 의 중급/고급 문법을 배움
- ✓ 코딩 테스트에 대응할 수 있는 기본 자세를 배움

- 대상: 데이터사이언스 전공 ONLY!

- ✓ 단, 복수전공, 부전공, 연계전공은 수강 가능

- 난이도: 파이썬 기초를 배웠다고 가정함 (프로그래밍 수준)



# 질문이 있을 땐

- Ecampus 질문 게시판
  - ✓ 쪽지 확인 어려움 (알림 안 뜸)
- E-mail (chk @ kangnam . ac . Kr)
  - ✓ 메일로 질문을 보낼 때엔 다음과 같은 규칙을 준수해주시기 바랍니다.  
(아래 내용 복사해서 쓰세요. 형식을 갖추지 않은 메일은 답장하지 않습니다.)

제목: [과목이름] 질문요지 간단히

내용:

안녕하세요,

저는 \*\*\*수업을 수강하는 \*\*학과 \*\*\*입니다.

이러이러한 질문이 있어서 메일 드렸습니다.

감사합니다.

\*\*\*드림



# 평가기준

- 중간고사 30%
- 기말고사 30%
- 과제 30%
  - ✓ 미제출 시 0점
  - ✓ 늦은 제출은 받지 않습니다 (시간을 충분히 드립니다).
- 출석 10%
  - ✓ 출석 미달 시 F이므로 출석에 유의!
  - ✓ 수강 후 출석체크가 되었는지 반드시 확인 (당일에는 반영이 안될 수도 있습니다.)
  - ✓ 지각(수강시간 미달) -1점, 결석(수강 안 함) -3점



# 성적

- 성적최대비율

- ✓ A 50 %

- ✓ B 50 %

- ✓ C ? %

- 이러면 성적이 당연히 안 좋겠죠?

- ✓ 시험을 보지 않거나, 시험을 보았는데 0점이거나, 과제를 안 냈거나, 출석이 매우 미달이거나...

- 상대평가



# Ecampus 에 과제 낼 때 요청사항

- 이름을 꼭 적어주세요!

홈 > 프로그래밍 > 1주차 [9월01일 - 9월07일] > 1주차-초간단과제

## 1주차-초간단과제

sample

모든 업로드된 파일은 표절검사를 받게 됩니다.

직접 작성

2018XXXXX 홍길동

과제 내용은 이리이리이러합니다!

저장 취소

[저장] 버튼을 클릭 시 최종 과제 제출일이 업데이트 되오니 주의하세요.

# Ecampus 에 과제 낼 때 요청사항

## • 사진 첨부는 이렇게

The screenshot displays the Ecampus assignment submission interface. On the left, the '직접 작성' (Direct Entry) section shows a toolbar with various editing tools. The '이미지 삽입' (Insert Image) icon is highlighted with an orange box. Below the toolbar, the text '2018XXXXX 홍길동' and '과제 내용은 이러이러이러합니다!' are visible. A modal window titled '이미지 삽입' (Insert Image) is open, showing a 'URL 입력' (URL Input) field and a '불러오기' (Load) button, which is also highlighted with an orange box. Below the URL field, there is a section for '이미지에 설명 남기기' (Leave a comment on the image) with a text area and a checkbox labeled '이미지 설명 불필요' (Image description not required). There are also fields for '크기' (Size) and '정렬' (Align), and a '자동 크기조절' (Automatic size adjustment) checkbox. At the bottom of the modal is an '이미지 저장' (Save Image) button. On the right, a '파일 선택 도구' (File Selection Tool) window is shown, listing options like '임베디드 파일' (Embedded file), '최근 파일' (Recent files), '파일 첨부' (Attach file), and '개인 파일' (Personal file). The '파일 첨부' (Attach file) option is selected, and a '찾아보기...' (Find...) button is highlighted with an orange box. Below it is a '파일 업로드' (Upload file) button.

## 파이썬으로 쉽게 풀어쓴 자료구조



★★★★★ 0.0 | 네티즌리뷰 0건

저자 최영규, 천인국 | 생능출판 | 2020.02.24

페이지 468 | ISBN 9788970503844

도서 26,190원 ~~27,000원~~ -3%

구매혜택 상세보기 >

♡ 3



### 바로구매

에스24	N Pay 1%	27,000원	구매
인터넷 교보문고	N Pay 1%	27,000원	구매
알라딘	N Pay 1%	27,000원	구매
인터파크 도서	N Pay 6%	26,190원	구매
강컴닷컴	N Pay 1%	26,190원	구매
반디앤루니스	N Pay 6%	27,000원	구매
도서11번가		27,000원	구매
키넥츠북	N Pay 6%	27,000원	구매



# Algorithm & Data Structure

- 이거 왜 배워요?
  - ✓ 알고리즘: 문제를 푸는 효율적인 논리 구조
  - ✓ 데이터 구조: 컴퓨터가 데이터를 다루는 방식에 대한 이해
  - ✓ 프로그래밍: 논리적으로 문제를 해결하는 방법
- 따라서 프로그래밍을 잘한다 -> 문제를 구조화(with DS)하여 프로그래밍 언어로 잘 해결한다 (with Algorithm)

# 예를 들어봅시다

- 최단 경로를 찾으려면 어떻게 해야할까?

- ✓ 경로들은 어떻게 정의할 수 있을까?
- ✓ 교통상황은 어떻게 반영할 수 있을까?
- ✓ 통행요금은 어떻게 고려할까?

○ 강남역 2호선  
○ 강남대학교

다시입력 + 경유지

길찾기 >

추천 이륜차우선 ①

54분 추천 | 33.5km

통행료 1,440원 · 주유비 4,348원 · 택시비 31,710원

정체 강남대로 1.2km → 서행 경부고속도로 28.3km →

서행 중부대로 2.8km

상세보기 >

1시간 4분 거리우선 | 30.4km

통행료 800원 · 주유비 3,945원 · 택시비 30,430원

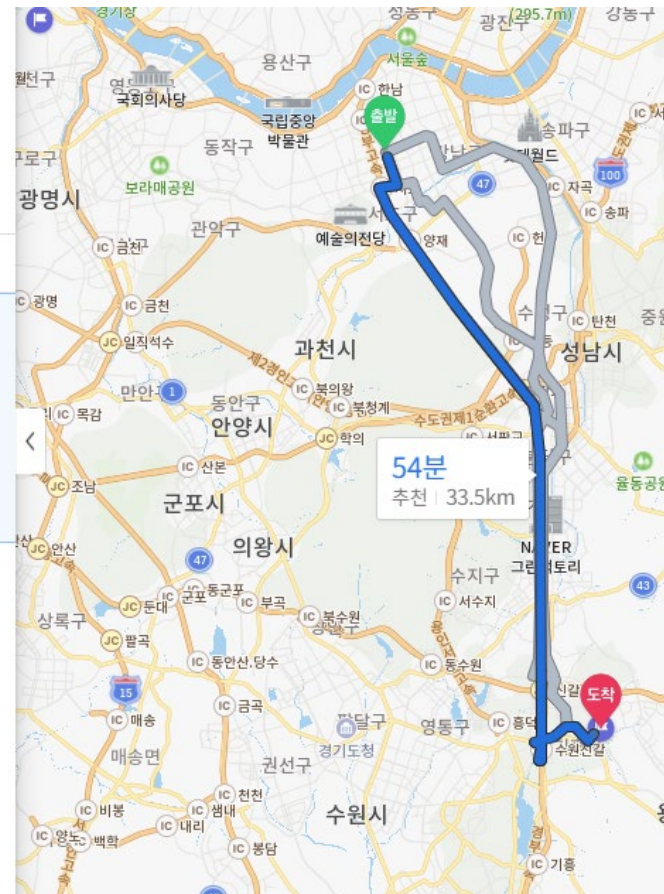
서행 경부고속도로 12.4km →

정체 분당수서간도시고속화도로 2.2km →

서행 분당수서간도시고속화도로 5.1km

상세보기 >

54분 시간우선 | 33.5km



# 코딩 테스트 알아보기

## • 코딩 테스트의 예

✓ <https://tech.kakao.com/2019/10/02/kakao-blind-recruitment-2020-round1/>

데이터 처리 전문가가 되고 싶은 "어피치"는 문자열을 압축하는 방법에 대해 공부를 하고 있습니다. 최근에 대량의 데이터 처리를 위한 간단한 비손실 압축 방법에 대해 공부를 하고 있는데, 문자열에서 같은 값이 연속해서 나타나는 것을 그 문자의 개수와 반복되는 값으로 표현하여 더 짧은 문자열로 줄여서 표현하는 알고리즘을 공부하고 있습니다. 간단한 예로 "aabbaccc"의 경우 "2a2ba3c"(문자가 반복되지 않아 한번만 나타난 경우 1은 생략함)와 같이 표현할 수 있는데, 이러한 방식은 반복되는 문자가 적은 경우 압축률이 낮다는 단점이 있습니다. 예를 들면, "abcabcdede"와 같은 문자열은 전혀 압축되지 않습니다. "어피치"는 이러한 단점을 해결하기 위해 문자열을 1개 이상의 단위로 잘라서 압축하여 더 짧은 문자열로 표현할 수 있는지 방법을 찾아보려고 합니다.

예를 들어, "abababcdabababcd"의 경우 문자를 1개 단위로 자르면 전혀 압축되지 않지만, 2개 단위로 잘라서 압축한다면 "2ab2cd2ab2cd"로 표현할 수 있습니다. 다른 방법으로 8개 단위로 잘라서 압축한다면 "2abababcd"로 표현할 수 있으며, 이때가 가장 짧게 압축하여 표현할 수 있는 방법입니다.

다른 예로, "abcabcdede"와 같은 경우, 문자를 2개 단위로 잘라서 압축하면 "abcabc2de"가 되지만, 3개 단위로 자른다면 "2abcdede"가 되어 3개 단위가 가장 짧은 압축 방법이 됩니다. 이때 3개 단위로 자르고 마지막에 남는 문자열은 그대로 붙여주면 됩니다.

압축할 문자열 s가 매개변수로 주어질 때, 위에 설명한 방법으로 1개 이상 단위로 문자열을 잘라 압축하여 표현한 문자열 중 가장 짧은 것의 길이를 return 하도록 solution 함수를 완성해주세요.



# 1. 자료구조와 알고리즘

# 이 장에서 배울 내용 (다 나가진 않을 겁니다...아마?)

- 자료구조와 알고리즘의 개념과 관계를 이해한다.
- 추상 자료형의 개념을 이해하고 Bag 자료형에 적용해 본다.
- 알고리즘의 실행 시간 측정 방법을 이해하고 활용할 수 있다.
- 알고리즘의 시간 복잡도 개념과 빅오 표기법 등을 이해한다.
- 순환의 개념과 구조를 이해하고, 다양한 순환 문제를 살펴본다.
- 순환을 통해 알고리즘의 시간 복잡도 분석 능력을 기른다.
- 2장에서 학습할 파이썬 코드의 형식에 미리 익숙해진다.



# 1.1 자료구조와 알고리즘

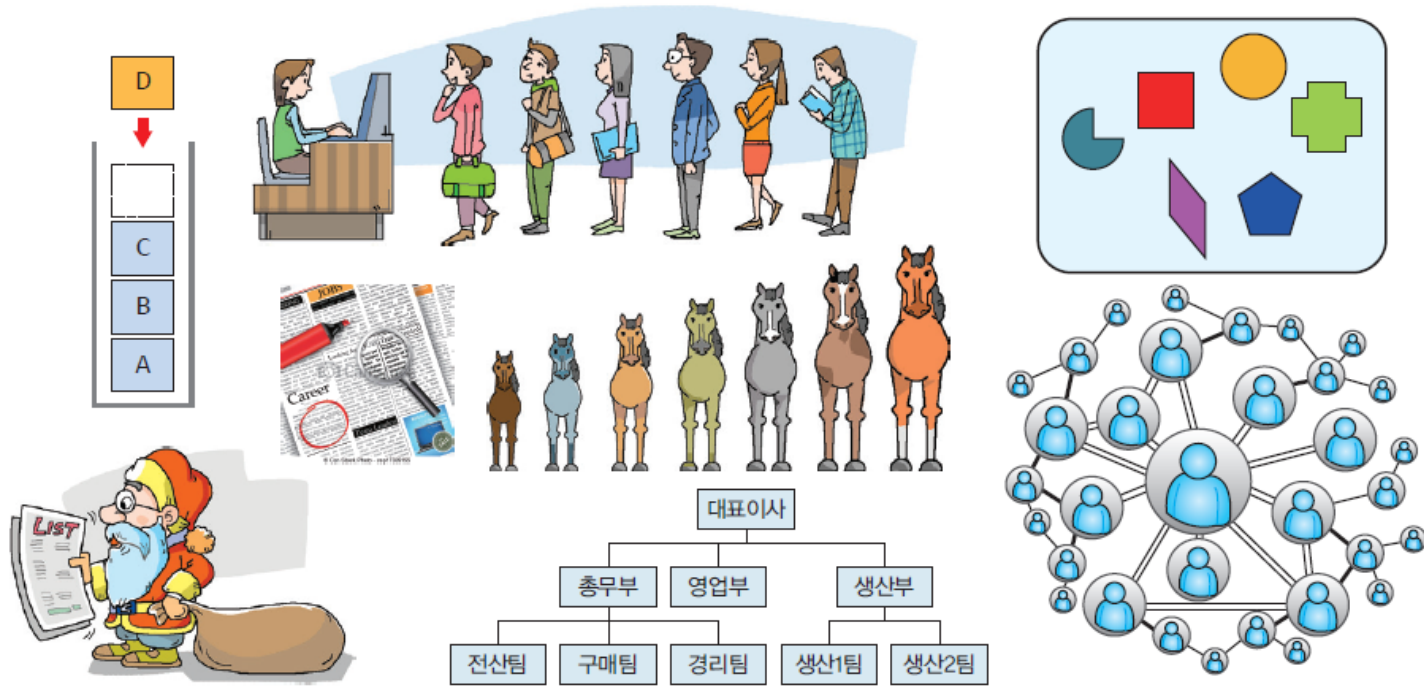
---

- 자료구조란?
- 알고리즘이란?
- 알고리즘의 조건
- 알고리즘의 기술 방법



# 자료구조란?

- 일상 생활에서 자료를 정리하고 조직화하는 이유는?
  - ✓ 사물을 편리하고 효율적으로 사용하기 위함
  - ✓ 다양한 자료를 효율적인 규칙에 따라 정리한 예



# 컴퓨터에서의 자료구조

- 자료구조(Data Structure)

- ✓ 컴퓨터에서 자료를 정리하고 조직화하는 다양한 구조

- 선형 자료구조

- ✓ 항목들을 순서적으로 나열하여 저장하는 창고
- ✓ 항목 접근 방법에 따라 다시 세분화
  - 리스트: 가장 자유로운 선형 자료구조
  - 스택, 큐, 덱: 항목의 접근이 맨 앞(전단)이 나 맨 뒤(후단)로 제한





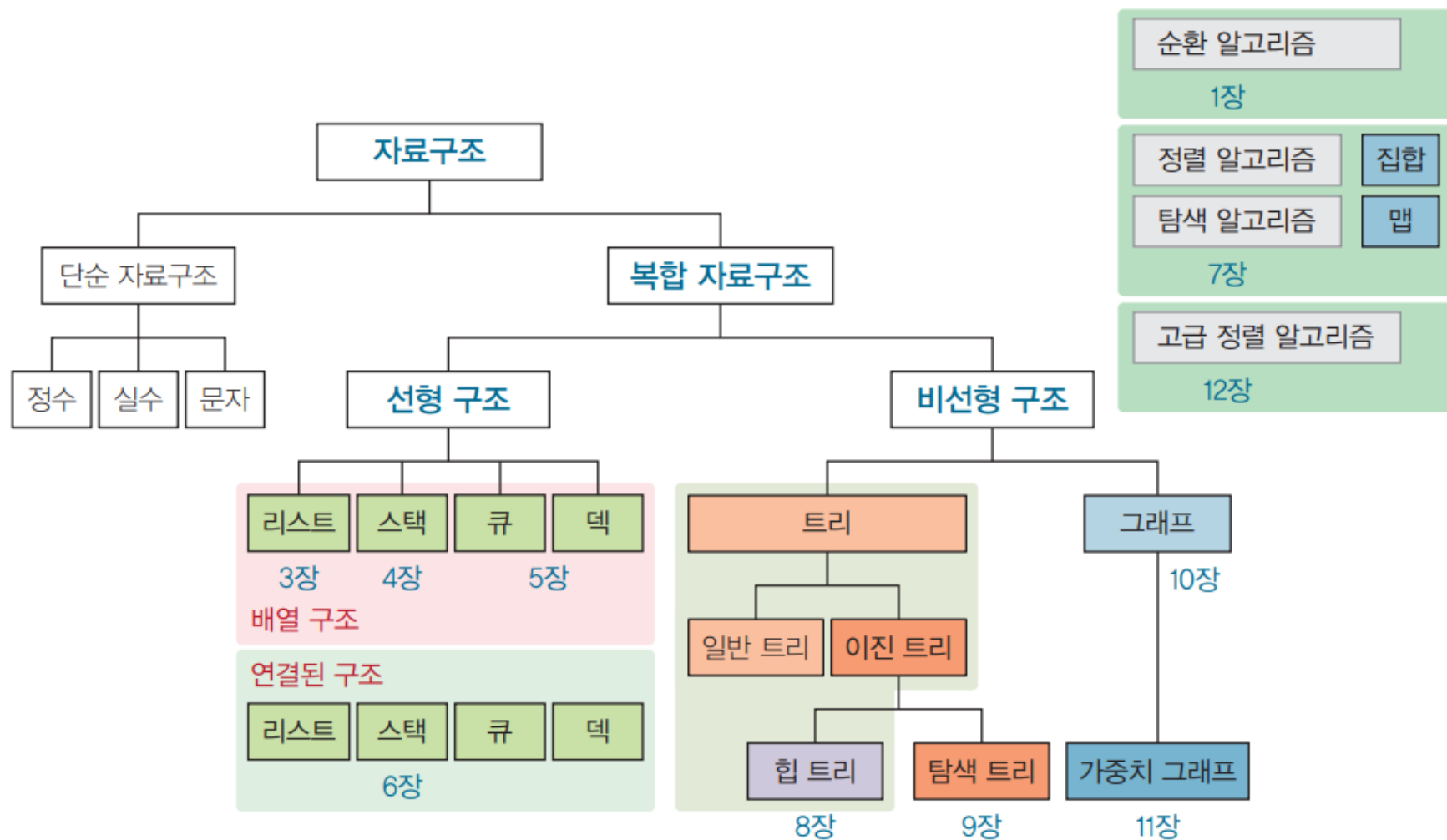
# 컴퓨터에서의 자료구조

- 비선형 자료구조

- ✓ 항목들이 보다 복잡한 연결 관계
- ✓ 트리: 회사의 조직도나 컴퓨터의 폴더와 같은 계층 구조
  - 힙 트리는 우선순위 큐
  - 이진 탐색트리나 AVL트리: 탐색을 위한 트리 구조
- ✓ 그래프: 가장 복잡한 연결 관계를 표현
  - 다양한 문제를 해결하기 위한 기본 구조로 사용된다.



# 배울 내용들을 정리해봅시다



# 알고리즘이란?

- 컴퓨터로 문제를 풀기 위한 단계적인 절차
  - ✓ 예) 사전에서 단어 찾기

사전에서 단어 하나  
찾는 것은 아주 쉽지.  
단어들이 알파벳 순으로  
정렬되어 있으니까



뭐야? 단어들이  
정렬되지 않고 섞여 있잖아?  
그럼 단어를 어떻게 찾지?



- 프로그램 = 자료구조 + 알고리즘

# 알고리즘의 조건

- 입 력 : 0개 이상의 입력이 존재하여야 한다.
- 출 력 : 1개 이상의 출력이 존재하여야 한다.
- 명백성 : 각 명령어의 의미는 모호하지 않고 명확해야 한다.
- 유한성 : 한정된 수의 단계 후에는 반드시 종료되어야 한다.
- 유효성 : 각 명령어들은 실행 가능한 연산이어야 한다.



# 알고리즘의 기술 방법

## 1. 자연어로 표현

*find\_max(A)*

1. 배열 A의 첫 번째 요소를 변수 tmp에 복사한다.
2. 배열 A의 다음 요소들을 차례대로 tmp와 비교하여, 더 크면 그 값을 tmp로 복사한다.
3. 배열 A의 모든 요소를 비교했으면 tmp를 반환한다.

쉬워 보이긴 한데...  
뭔가 좀 정확하지  
않아 보이네...

이렇게 까지는  
피곤하게 살고  
싶지 않아...

## 3. 유사 코드로 표현

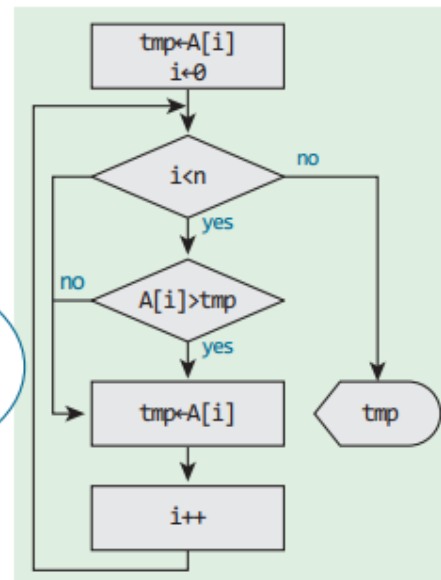
*find\_max(A)*

```
tmp ← A[0];  
for i ← 1 to size(A) do  
    if tmp < A[i] then  
        tmp ← A[i]  
return tmp
```

논문에서 많이  
사용하는 방법.  
음... 괜찮네.

다른 언어와는 다르게  
파이썬으로 표현하는 것도  
아주 간단하군...

## 2. 흐름도로 표현



## 4. 파이썬으로 표현

```
def find_max( A ):  
    tmp = A[0]  
    for item in A :  
        if item > tmp :  
            tmp = item  
    return tmp
```

# 알고리즘의 기술 방법

- (1) 자연어
  - ✓ 읽기 쉬움. 단어들을 정확하게 정의하지 않으면 의미 모호.
- (2) 흐름도
  - ✓ 직관적. 이해하기 쉬움. 복잡한 알고리즘→상당히 복잡!
- (3) 유사코드
  - ✓ 프로그램을 구현할 때의 여러 가지 문제들을 감출 수 있음
  - ✓ 알고리즘의 핵심적인 내용에만 집중 가능
- (4) 특정 언어
  - ✓ 알고리즘의 가장 정확한 기술 가능
  - ✓ 구현시의 사항들이 알고리즘의 핵심적인 내용들의 이해를 방해
  - ✓ 파이썬: C나 자바보다 훨씬 간결한 표현 가능



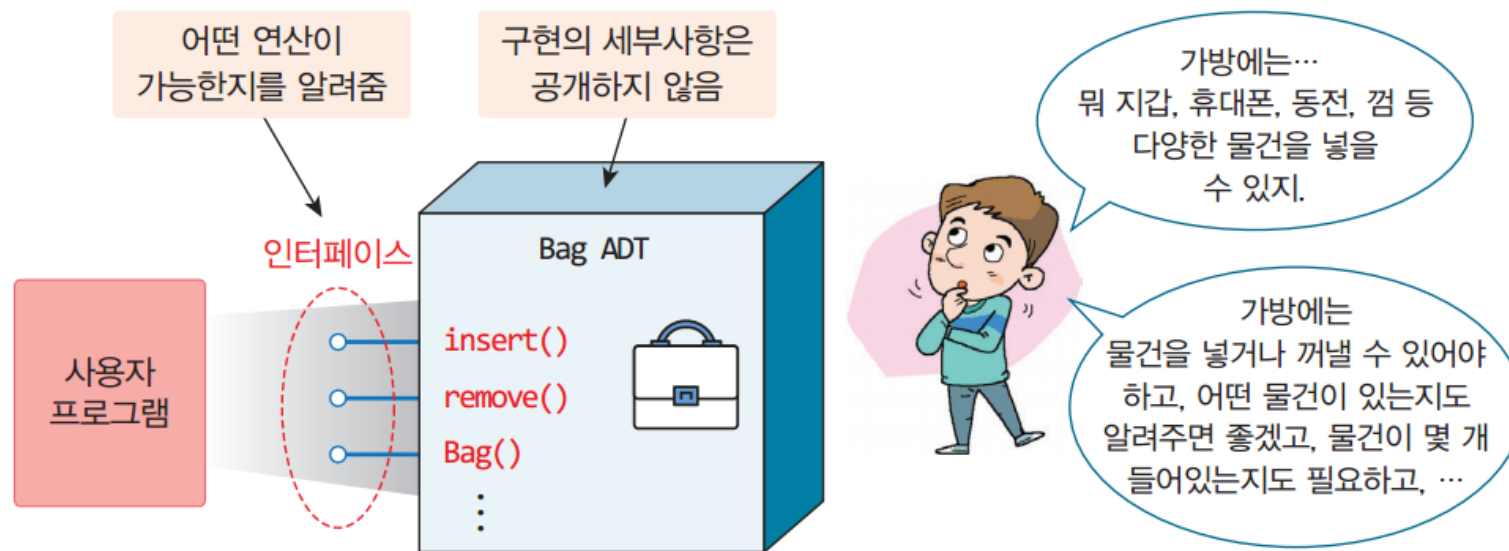
# 1.2 추상 자료형

- 추상 자료형(Abstract Data Type, ADT)이란?
- 예) Bag 추상 자료형의 정의
- <실습> Bag 추상 자료형의 구현과 활용



# 추상 자료형(ADT)이란?

- 프로그래머가 추상적으로 정의한 자료형
  - ✓ 데이터 타입을 추상적(수학적)으로 정의한 것
    - 데이터나 연산이 무엇(what)인가를 정의함
    - 데이터나 연산을 어떻게(how) 구현할 것인지는 정의하지 않음
  - ✓ 시스템의 정말 핵심적인 구조나 동작에만 집중



[그림 1.3] 가방(Bag)의 추상 자료형



# 예) Bag의 추상 자료형

데이터: 중복된 항목을 허용하는 자료들의 저장소. 항목들은 특별한 순서가 없이 개별적으로 저장되지만 항목간의 비교는 가능해야 함.

연산:

- `Bag()`: 비어있는 가방을 새로 만든다.
- `insert(e)`: 가방에 항목 `e`를 넣는다.
- `remove(e)`: 가방에 `e`가 있는지 검사하여 있으면 이 항목을 꺼낸다.
- `contains(e)`: `e`가 들어있으면 `True`를 없으면 `False`를 반환한다.
- `count()`: 가방에 들어 있는 항목들의 수를 반환한다.



# 예) Bag 추상 자료형의 구현

- 함수를 이용한 Bag 연산들의 구현 예(파이썬)

```
def contains(bag, e) :           # bag에 항목 e가 있는지 검사하는 함수
    return e in bag             # 파이썬의 in 연산자 사용

def insert(bag, e) :            # bag에 항목 e를 넣는 함수
    bag.append(e)               # 파이썬 리스트의 append메소드 사용

def remove(bag, e) :           # bag에서 항목 e를 삭제하는 함수
    bag.remove(e)               # 파이썬 리스트의 remove메소드 사용

def count(bag):                 # bag의 전체 항목 수를 계산하는 함수
    return len(bag)             # 파이썬의 len 함수 사용
```

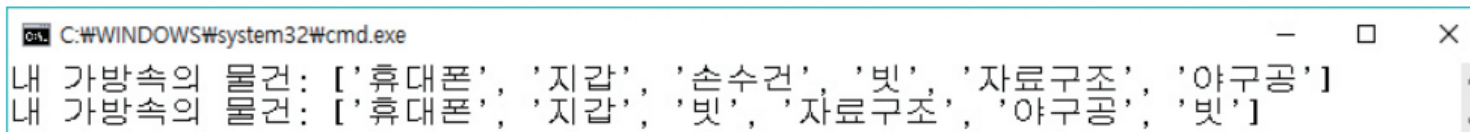


# 예) Bag의 활용

## • Bag을 이용한 자료 관리 예

```
myBag = []                                # Bag을 위한 빈 리스트를 만들
insert(myBag, '휴대폰')                   # Bag에 휴대폰 삽입
insert(myBag, '지갑')                     # Bag에 지갑 삽입
insert(myBag, '손수건')                   # Bag에 손수건 삽입
insert(myBag, '빗')                      # Bag에 빗 삽입
insert(myBag, '자료구조')                 # Bag에 자료구조 삽입
insert(myBag, '야구공')                   # Bag에 야구공 삽입
print('가방속의 물건:', myBag)            # Bag의 내용 출력

insert(myBag, '빗')                      # Bag에서 '빗'삽입(중복)
remove(myBag, '손수건')                   # Bag에서 '손수건'삭제
print('가방속의 물건:', myBag)            # Bag의 내용 출력
```



C:\WINDOWS\system32\cmd.exe

내 가방속의 물건: ['휴대폰', '지갑', '손수건', '빗', '자료구조', '야구공']

내 가방속의 물건: ['휴대폰', '지갑', '빗', '자료구조', '야구공', '빗']



# 1.3 알고리즘의 성능 분석

- 알고리즘의 실행시간을 측정해 보자.
- 알고리즘의 복잡도 분석이란?
  - ✓ Bag의 삽입연산
  - ✓  $n^2$  을 구하는 세 알고리즘 비교
- 빅오, 빅오메가, 빅세타 표기법
- 입력 데이터에 따른 성능 차이



# 알고리즘의 성능분석

- 알고리즘의 성능 분석 기법

- ✓ 실행 시간을 측정하는 방법

- 두 개의 알고리즘의 실제 실행 시간을 측정하는 것
    - 실제로 구현하는 것이 필요
    - 동일한 하드웨어를 사용하여야 함

- ✓ 알고리즘의 복잡도를 분석하는 방법

- 직접 구현하지 않고서도 수행 시간을 분석하는 것
    - 알고리즘이 수행하는 연산의 횟수를 측정하여 비교
    - 일반적으로 연산의 횟수는  $n$ 의 함수
    - 시간 복잡도 분석 : 수행 시간 분석
    - 공간 복잡도 분석 : 수행시 필요로 하는 메모리 공간 분석



# (1) 실행시간 측정

- 파이썬의 실행시간 측정 코드 예

```
import time                                # time 모듈 불러오기

myBag = []                                # 비어있는 새로운 가방을 하나 만듦
start = time.time()                       # 현재 시각을 start에 저장
insert(myBag, '축구공')                   # 실행시간을 측정하려는 코드
...                                       # ...
end = time.time()                         # 현재 시각을 end에 저장
print("실행시간 = ", end-start)           # 실행시간(종료-시작)을 출력
```



## (2) 복잡도 분석

### • 시간 복잡도

- ✓ 산술, 대입, 비교, 이동의 기본적인 연산 고려
- ✓ 알고리즘 수행에 필요한 연산의 개수를 계산
- ✓ 입력의 개수  $n$ 에 대한 함수  $\rightarrow$  시간복잡도 함수,  $T(n)$



# 복잡도 분석 예: Bag의 삽입연산

- 방법 1: 리스트의 맨 뒤에 삽입

- ✓ append() 함수 사용

```
def insert(bag, e) :           # bag에 항목 e를 넣는 함수
    bag.append(e)              # 파이썬 리스트의 맨 뒤에 추가
```

- ✓ 효율적 → 바로 삽입 가능

- 방법 2: 리스트의 맨 앞에 삽입

- ✓ Insert() 함수 사용

```
def insert(bag, e) :           # bag에 항목 e를 넣는 함수
    bag.insert(0, e)           # 파이썬 리스트의 맨 앞에 추가
```

- ✓ 비 효율적 → 가방의 모든 물건을 먼저 이동해야 삽입 가능





# $n^2$ 을 구하는 문제

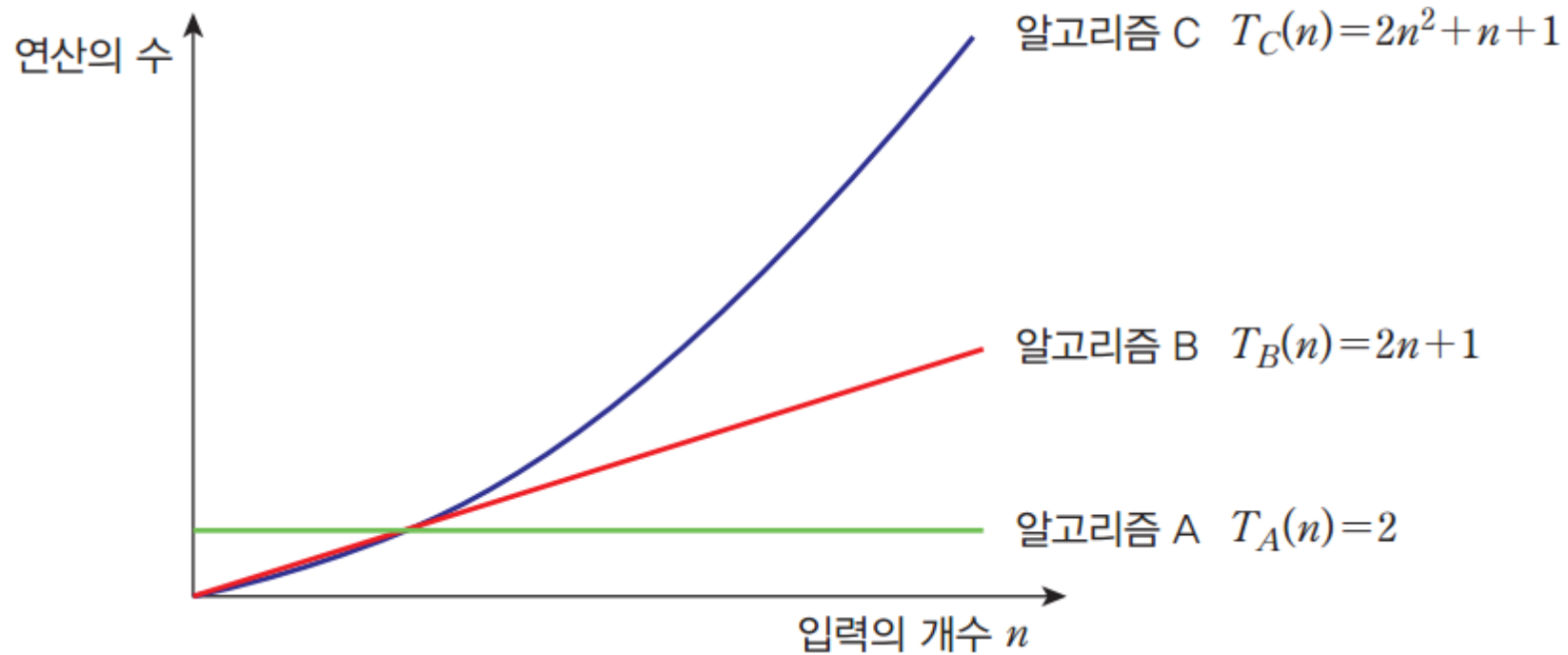
- 3 가지 알고리즘

- ✓ 각 알고리즘이 수행하는 연산의 개수 계산
- ✓ 단 for 루프 제어 연산은 고려하지 않음

알고리즘	A	B	C
유사 코드	$\text{sum} \leftarrow n * n$	<pre>for i ← 1 to n do   sum ← sum + n</pre>	<pre>for i ← 1 to n do   for j ← 1 to n do     sum ← sum + 1</pre>
연산 횟수	대입연산: 1 곱셈연산: 1	대입연산: $n+1$ 덧셈연산: $n$	대입연산: $n^2 + n + 1$ 덧셈연산: $n^2$
복잡도 함수	$T_A(n) = 2$	$T_B(n) = 2n + 1$	$T_C(n) = 2n^2 + n + 1$



# $n^2$ 을 구하는 세 알고리즘 비교



# 빅오 표기법

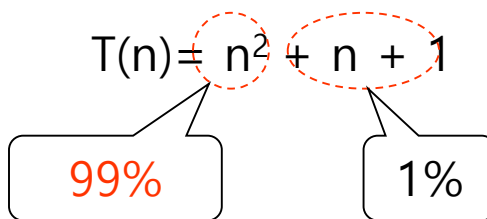
- 차수가 가장 큰 항이 절대적인 영향

- ✓ 다른 항들은 상대적으로 무시

- ✓ 예:  $T(n) = n^2 + n + 1$

- $n=1$ 일때 :  $T(n) = 1 + 1 + 1 = 3$  ( $n^2$  항이 33.3%)
- $n=10$ 일때 :  $T(n) = 100 + 10 + 1 = 111$  ( $n^2$  항이 90%)
- $n=100$ 일때 :  $T(n) = 10000 + 100 + 1 = 10101$  ( $n^2$  항이 99%)
- $n=1,000$ 일때 :  $T(n) = 1000000 + 1000 + 1 = 1001001$  ( $n^2$  항이 99.9%)

$n=100$ 인 경우

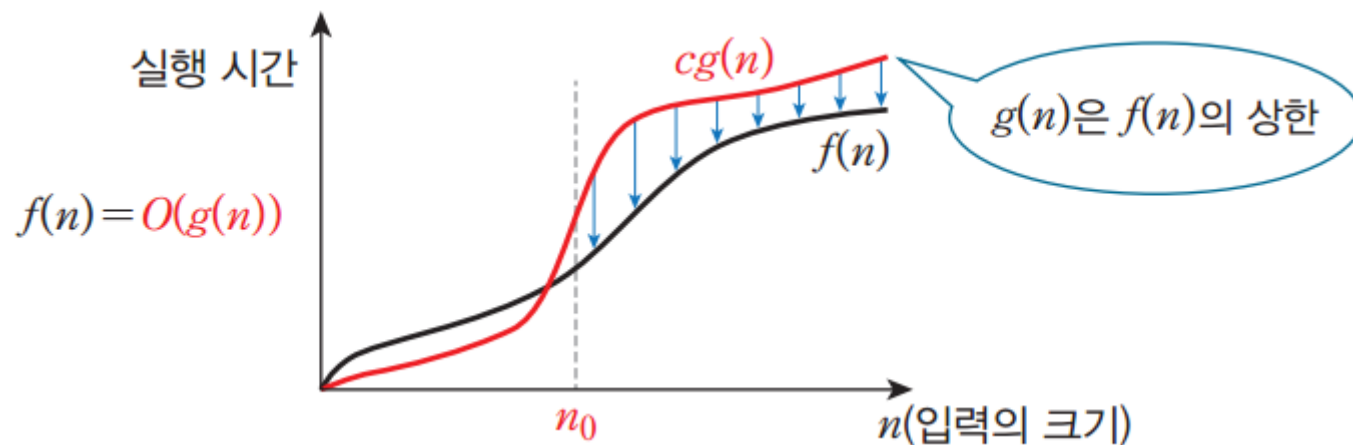


# 빅오 표기법의 정의

## 정의 1.3 빅오 표기법

두개의 함수  $f(n)$  과  $g(n)$  이 주어졌을 때 모든  $n > n_0$  에 대해  $|f(n)| \leq c |g(n)|$  을 만족하는 상수  $c$  와  $n_0$  가 존재하면  $f(n) = O(g(n))$  이다.

✓ 연산의 횟수를 대략적(점근적)으로 표기한 것



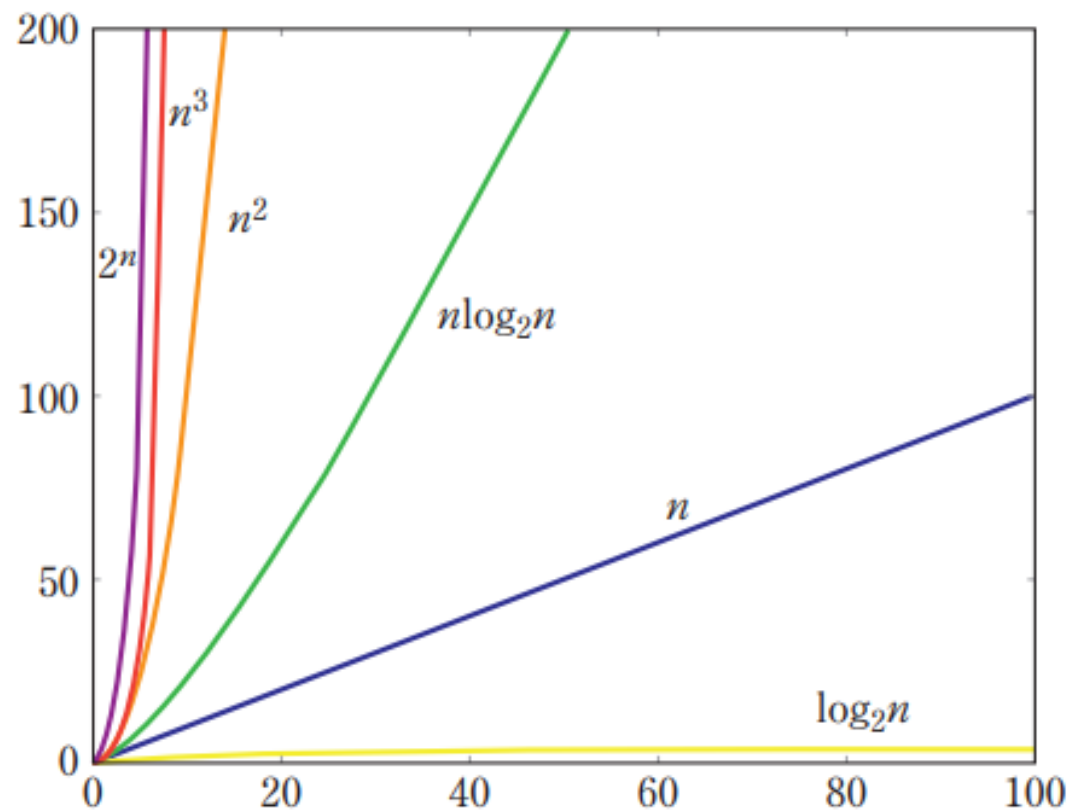
# 빅오 표기법의 예

- ✓  $f(n)=5$ 이면  $O(1)$ 이다. 왜냐하면  $n_0=1, c=10$ 일 때,  $n \geq 1$ 에 대하여  $5 \leq 10 \cdot 1$ 이 되기 때문이다.
- ✓  $f(n)=2n+1$ 이면  $O(n)$ 이다. 왜냐하면  $n_0=2, c=3$ 일 때,  $n \geq 2$ 에 대하여  $2n+1 \leq 3n$ 이 되기 때문이다.
- ✓  $f(n)=3n^2 + 100$ 이면  $O(n^2)$ 이다. 왜냐하면  $n_0=100, c=5$ 일 때,  $n \geq 100$ 에 대하여  $3n^2 + 100 \leq 5n^2$ 이 되기 때문이다.
- ✓  $f(n)=5 \cdot 2^n + 10n^2 + 100$ 이면  $O(2^n)$ 이다.  
왜냐하면  $n_0=1000, c=10$ 일 때,  $n \geq 1000$ 에 대하여  $5 \cdot 2^n + 10n^2 + 100 < 10 \cdot 2^n$ 이 되기 때문이다.



# 빅오 표기법의 종류

$O(1)$ : 상수형  
 $O(\log n)$ : 로그형  
 $O(n)$ : 선형  
 $O(n \log n)$ : 선형로그형  
 $O(n^2)$ : 2차형  
 $O(n^3)$ : 3차형  
 $O(2^n)$ : 지수형  
 $O(n!)$ : 팩토리얼형



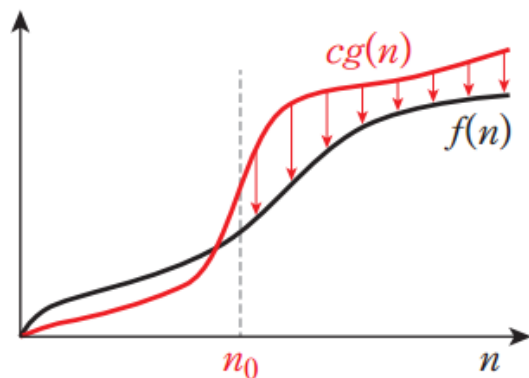
# 빅오메가와 빅세타 표기법

## 정의 1.4 빅오메가 표기법

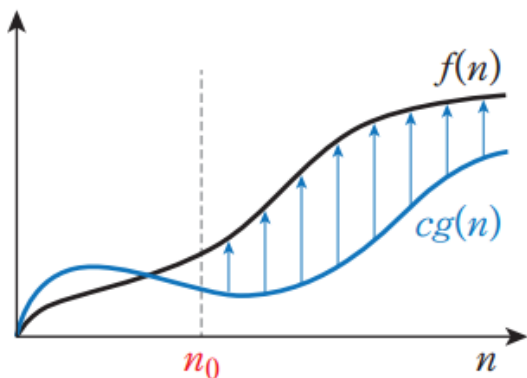
두개의 함수  $f(n)$  과  $g(n)$  이 주어졌을 때 모든  $n > n_0$  에 대해  $|f(n)| \geq c |g(n)|$  을 만족하는 상수  $c$  와  $n_0$  가 존재하면  $f(n) = \Omega(g(n))$  이다.

## 정의 1.5 빅세타 표기법

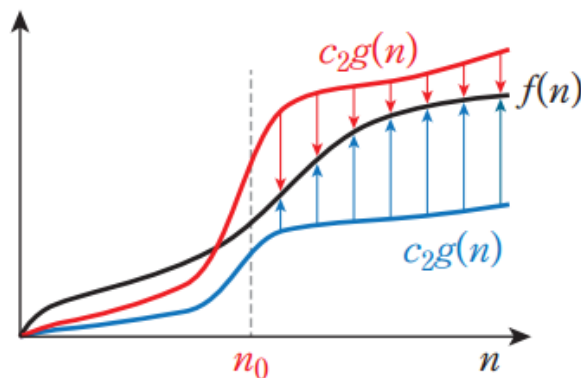
두개의 함수  $f(n)$  과  $g(n)$  이 주어졌을 때 모든  $n > n_0$  에 대해  $c_1 |g(n)| \leq f(n) \leq c_2 |g(n)|$  을 만족하는 상수  $c_1, c_2$  와  $n_0$  가 존재하면  $f(n) = \Theta(g(n))$  이다.



$$f(n) = O(g(n))$$



$$f(n) = \Omega(g(n))$$

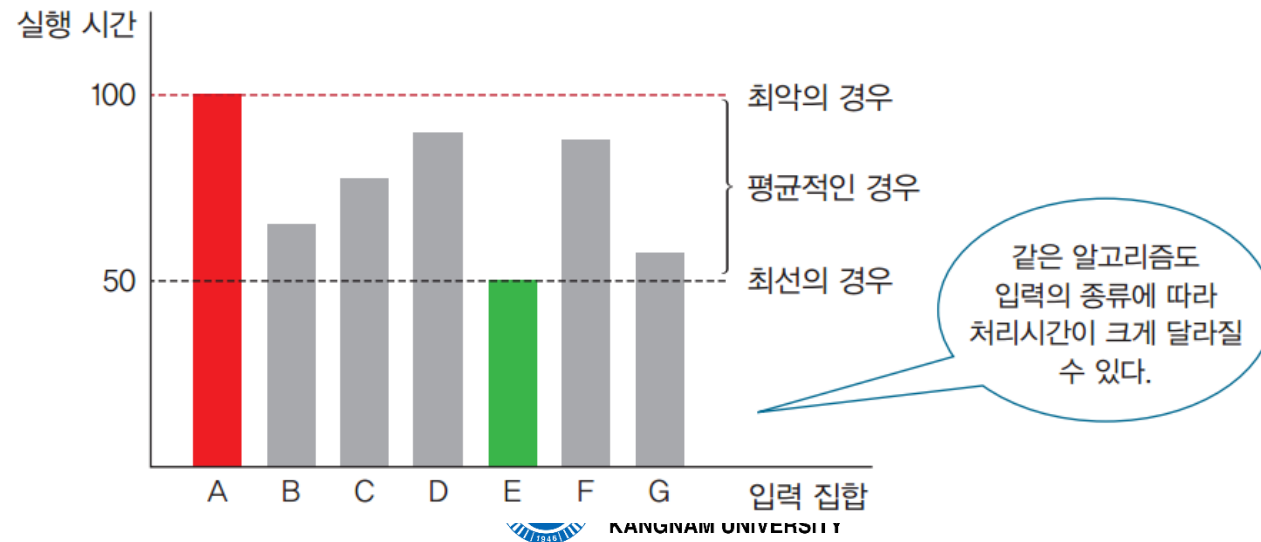


$$f(n) = \Theta(g(n))$$



# 최선, 평균, 최악의 경우

- 실행시간은 입력 집합에 따라 다를 수 있음
  - ✓ 최선의 경우(best case): 수행 시간이 가장 빠른 경우
    - 의미가 없는 경우가 많다.
  - ✓ 평균의 경우(average case): 수행시간이 평균적인 경우
    - 계산하기가 상당히 어려움.
  - ✓ 최악의 경우(worst case): 수행 시간이 가장 늦은 경우
    - 가장 널리 사용됨. 계산하기 쉽고 응용에 따라서 중요한 의미를 가짐. (예) 비행기 관제업무, 게임, 로봇틱스





# 과제

- 아래의 영상을 보고 느낀 점을 간단히 써서 올리세요.

✓ <https://www.youtube.com/watch?v=okklyWhN0iQ>

