1강

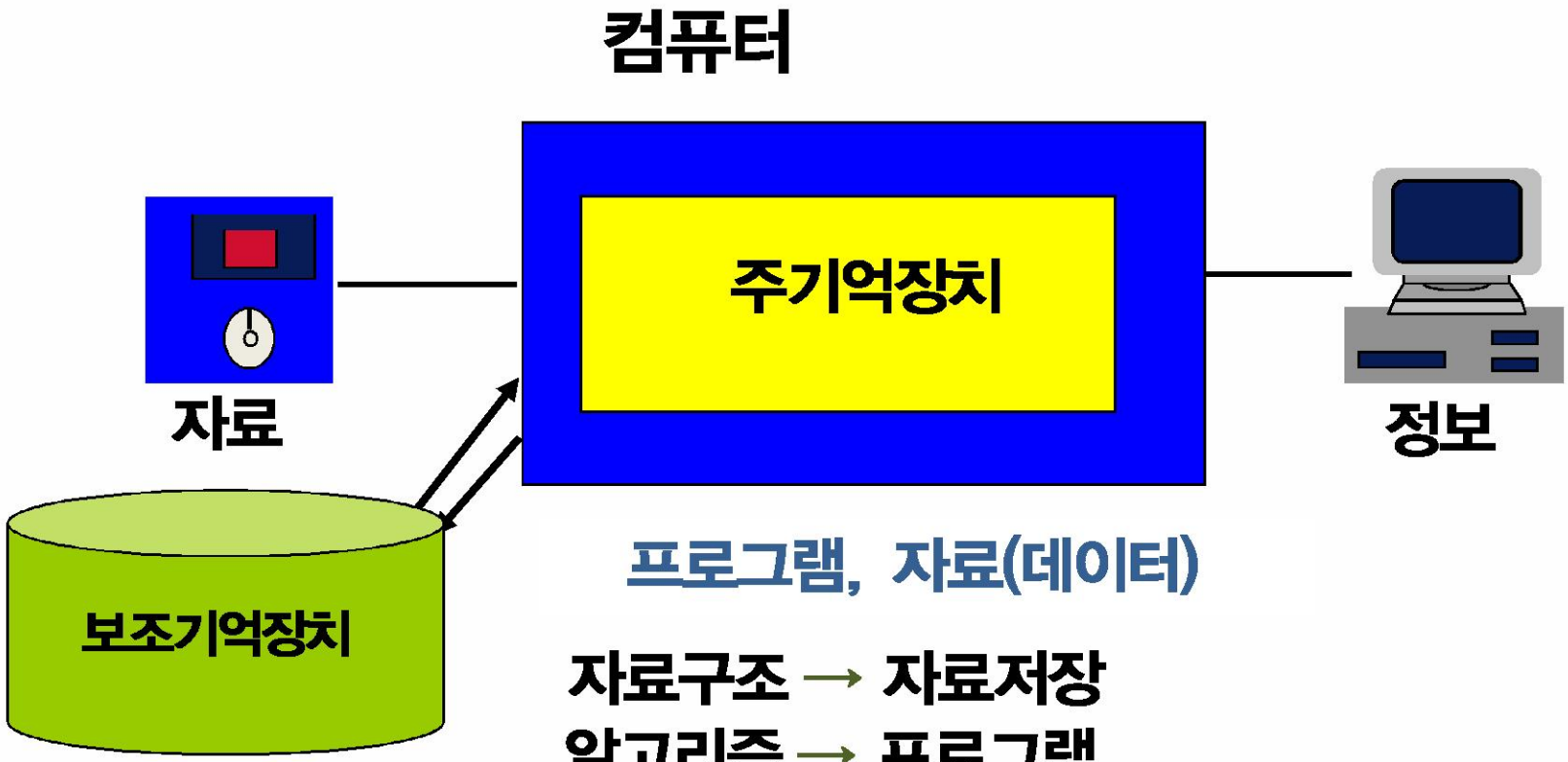
기본개념

컴퓨터과학과 이언배 교수

이 시간 강의내용

- 자료구조와 알고리즘
- 단순자료
- 추상 자료형
- 순환 알고리즘
- 알고리즘의 성능분석

자료와 정보



프로그램, 자료(데이터)

자료구조 → 자료저장
알고리즘 → 프로그램

I = P(D)

자료구조의 영역

알고리즘과 프로그램

- 알고리즘(algorithm)
  - 특정 문제를 해결하기 위해 기술한 일련의 논리의 순서
- 프로그램(program)
  - 알고리즘을 컴퓨터가 이해하고 실행할 수 있는 특정
  - 프로그래밍 언어로 표현한 것
  - program = algorithm + data structure

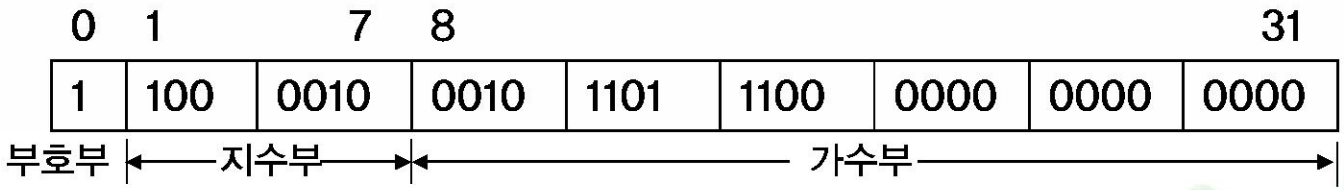
알고리즘의 특성

- **입력**
  - 외부에서 제공되는 자료가 있을 수 있다.
- **출력**
  - 적어도 한가지 결과를 생성한다.
- **명확성**
  - 각 명령들은 명확하고 모호하지 않아야 한다.
- **유한성**
  - 알고리즘의 명령대로 수행하면, 어떤 경우에도 한정된 수의 단계 뒤에는 반드시 종료한다.
- **유효성**
  - 기본적으로 모든 명령들은 종이와 연필 만으로 수행될 수 있어야 한다. 알고리즘은 각 연산이 명확해서만 안되고 반드시 실행 가능해야 한다.

자료의 종류와 표현(1)

- 정수형 자료
  - 4byte (short : 2byte, long : 4byte)
  - 10진수 35표현

- 실수형 자료
  - 4byte (float : 4byte, double과 long double : 8byte)
  - 10진수 –45.75표현



자료의 종류와 표현(2)

- 문자형 자료
  - 영문자 : 1byte, 한글문자 : 2byte
  - A는 ASCII 코드인 '01000001'이 저장
- 논리형 자료
  - 참(true, 1) 또는 거짓(false, 0)
  - &&(AND), ||(OR), !(NOT)
- 포인터형 자료
  - int i, *pi : pi = &i : *pi = 10 :

자료와 자료형

- 자료
  - 현실 세계의 관찰이나 측정을 통해 수집된 값이나 사실
  - 프로그램의 처리 대상이 되는 모든 것
- 자료형
  - 자료의 집합과 이 자료에 적용할 수 있는 연산의 집합
  - 시스템의 정의 자료형

    원시 자료형(단순 자료형), 복합 자료형(구조화 자료형)
  - 사용자 정의 자료형

    예) integer 자료형

    자료:정수(…–2,–1,0,1,2…), 연산자: +, –, *, /, mod

추상 자료형

- 추상화(abstraction)
  - 자세한 것은 무시하고, 필수적이고 중요한 속성만으로 단순화시키는 과정
- 추상 자료형(abstraction data type : ADT)
  - 자료형의 논리적 정의
  - 자료와 연산의 본질에 대한 명세만 정의한 자료형
  - 자료가 무엇이고, 각 연산의 기능을 정의
- 추상화와 구체화의 관계

	자료	연산
추상화	추상 자료형	알고리즘
구체화	자료형	프로그램

자연수(Natno) 추상 자료형

```
ADT Natno
object : { i | i ∈ integer, i ≥ 0}
function :  for all x, y ∈ Natno
        zero() ::= return 0;
        isZero(x) ::= if (x=0) then return true
                      else return false;
        succ(x) ::= return x + 1;
        add(x, y) ::= return x + y;
        subtract(x, y) ::= if (x < y) then return 0
                           else return x − y;
        equal(x, y) ::= if (x = y) then return true
                        else return false;
End Natno
```

순환(recursion)

- 순환의 종류
  - 직접순환 : A → A'
  - 간접순환 : A → B → A
- 순환 방식의 적용
  - 분할 정복(divide and conquer)의 특성을 가진 문제에 적합
  - 어떤 복잡한 문제를 직접 간단하게 풀 수 있는 작은 문제로 분할하여 해결하려는 방법

순환 알고리즘과 비순환 알고리즘

- n!의 계승(factorial) 함수를 정의

$$n! = \begin{cases} 1 & n \le 1\text{일 때} \\ n \cdot (n-1) \cdots 2 \cdot 1 & n > 1\text{일 때} \end{cases}$$

- 순환 정의

$$n! = \begin{cases} 1 & n \le 1\text{일 때} \\ n \cdot (n-1)! & n > 1\text{일 때} \end{cases}$$

Factorial 순환함수

- 순환 알고리즘

```
Factorial(n)
    if (n <= 1) then return 1
                else return (n · factorial(n–1));
    end Factorial()
```

Factorial(4) = (4 · Factorial(3))
= (4 · (3 · Factorial(2)))
= (4 · (3 · (2 · Factorial(1))))
= (4 · (3 · (2 · 1)))
= (4 · (3 · 2))
= (4 · 6)
= 24

Factorial 의 비순환적 표현

```
n_Factorial(n)
    if (n <= 1) then fact←1;
          else for(i←2; i<= n; i← i+1) do
                fact←fact*i ;
        return  fact;
    end n_Factorial()
```

(순환적 표현)

```
Factorial(n)
    if (n <= 1) then return 1
                else return (n · factorial(n−1));
    end Factorial()
```

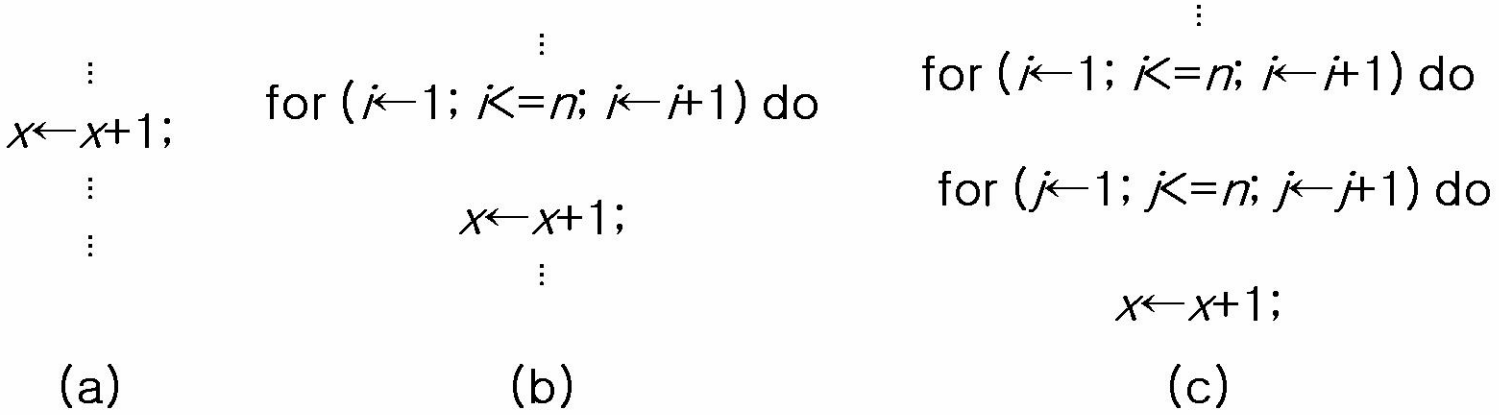
성능분석과 성능측정

- 성능 분석(performance analysis)
  - 프로그램을 실행하는데 필요한 시간과 공간 추정
- 성능 측정(performance measurement)
  - 컴퓨터가 실제로 프로그램을 실행하는데 걸리는 시간 측정

성능분석

- 공간복잡도
  - Sp = Sc + Se
  - 고정공간 + 가변공간
- 시간복잡도
  - Tp = Tc + Te
  - 컴파일시간 + 실행시간

프로그램의 빈도수 계산

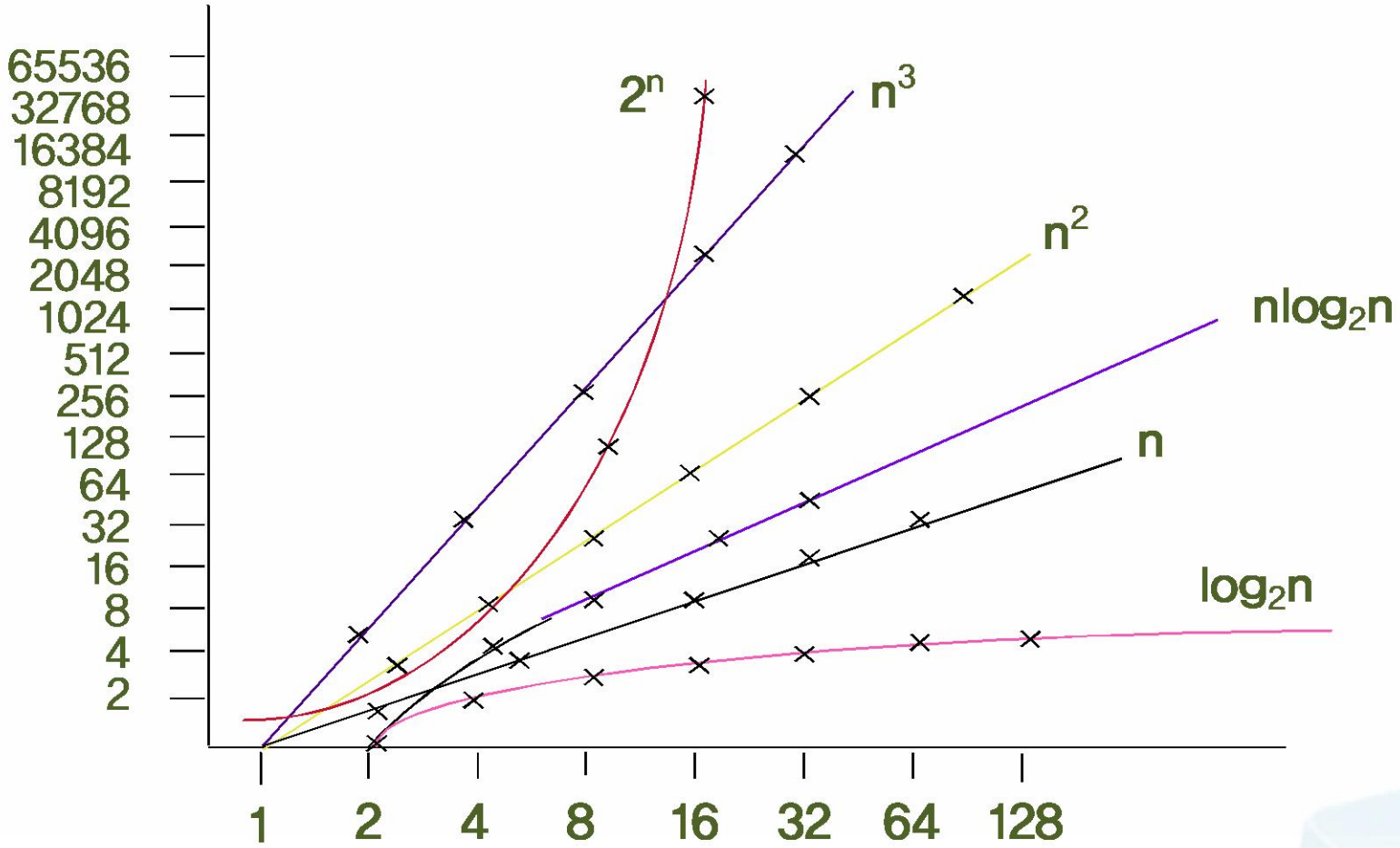


(a) (b) (c)

연산시간 표기법

- 연산시간 표기법
  - Big-Oh (O), Big-Omega (Ω), Big-Theta (θ)
- Big-Oh (O)
  - f, g가 양의 정수를 갖는 함수일 때, 두 양의 상수 a, b가 존재하고, 모든 n ≥ b에 대해 $f(n) \leq a_1 g(n)$이면,
    $f(n) = O(g(n))$
    $f(n) = 3n + 2$: $f(n) = O(n)$ (a=4, b=2)
    $f(n) = 6 \cdot 2^n + n^2$: $f(n) = O(2^n)$ (a=1001, b=100)
    $f(n) = 100$: $f(n) = O(1)$
    $f(n) = 1000n^2 + 100n - 6$: $f(n) = O(n^2)$

연산시간 함수의 변화


65536
32768
16384
8192
4096
2048
1024
512
256
128
64
32
16
8
4
2
2^n
n^3
n^2
$n\log_2 n$
n
$\log_2 n$
1
2
4
8
16
32
64
128

연산시간 표현

- $O(1)$: 상수
- $O(\log_2 n)$: 로그형
- $O(n)$: 선형
- $O(n^2)$: 평방형
- $O(n^3)$: 입방형
- $O(2^n)$: 지수형

$$O(1) < O(\log_2 n) < O(n) < O(n^2) < O(n^3) < O(2^n)$$