보고서 pdf + exe파일 + 소스코드

불량한 코딩스타일 X 주석, 의미없는것 X, 블록, ... (가독성↑)

6~7장까지가 중간고사

Data Structures

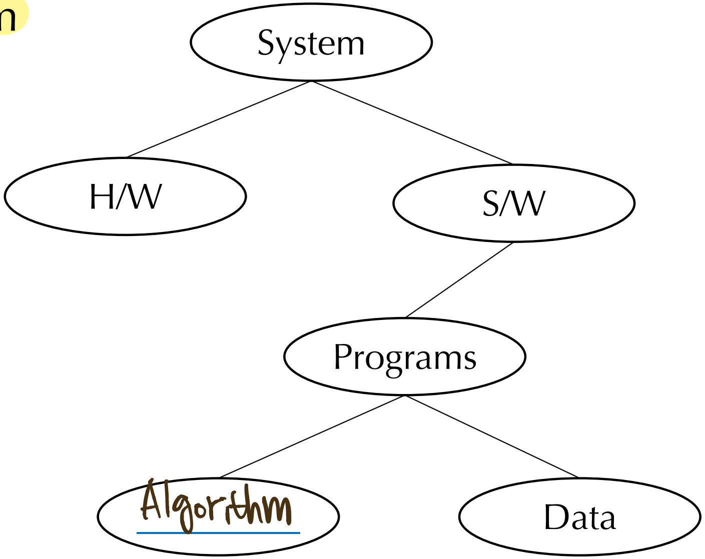
1. Introduction

Introduction

1. Software and Data Structure
2. Algorithm
3. System Life Cycle
4. Data Type
5. Abstract Data Type

Computer System

- Algorithms 문제해결을 위한 절차
  - Instructions for solving a problem
- Data
  - A collection of values measured



Definition

방법론 구조체

- Ways of, or structures for efficiently processing and organizing an large amount of ____data____ using computer system

- What to learn
  - ____Concepts____ and Properties of Data structures
  - ____Algorithms____ for handling Data structures

Algorithm vs Program

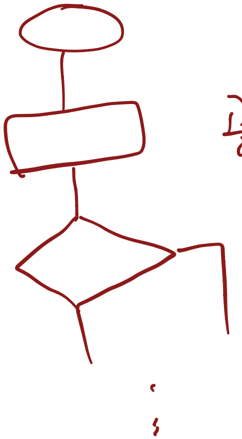
- Definition
  - A finite set of instructions for accomplishing a particular task
- Criteria

  명시적 묵시적 (범위에해당하는등)

  - Input : no explicit input (requires implicit input)
  - Output : at least one output
  - 명확성 Definiteness : specific and unambiguous instructions
    - ex) input some large number
  - 유한성 Finiteness : terminates after finite steps
  - Effectiveness : executable, implementable 실현가능
    - cf) count all prime numbers

Algorithm

- Description
  - Pseudo code 의사코드(중간)
  - Natural language
  - Flow chart
  - Programming language



공간을 많이 차지함

- Algorithm types
  - Search
  - Sort
  - Compute
  - Decision

5주

Algorithm - Prime Numbers

- Natural numbers that are greater than 1, and not divisible, except 1 or itself
- Problem) find all prime numbers between 2 and n

교재 program 1.1 (p5)

```
int prime (int n)
{
    if ( n<2 ) return 0;  거짓
    for (i = 2 ; i < n; i++)
        if ( n%i==0 ) return 0; 나누어지므로 거짓

    return 1;        ★ n=2인경우 주의
}
```

교재상의 j부분

Huffman Coding Tree

압축기법 빈도

- Text compression method by frequency
- Example : "time and tide wait for no man"

t	i	m	e	‘ ‘	a	n	d	w	f	o	r
3	3	2	2	6	3	3	2	1	1	2	1

↓

오름차순정렬

w	f	r	m	e	d	o	t	i	a	n	‘ ‘
1	1	1	2	2	2	2	3	3	3	3	6

여러가지 나올수있다

계층자료구조

root
부모
|
자식

Huffman Coding Tree

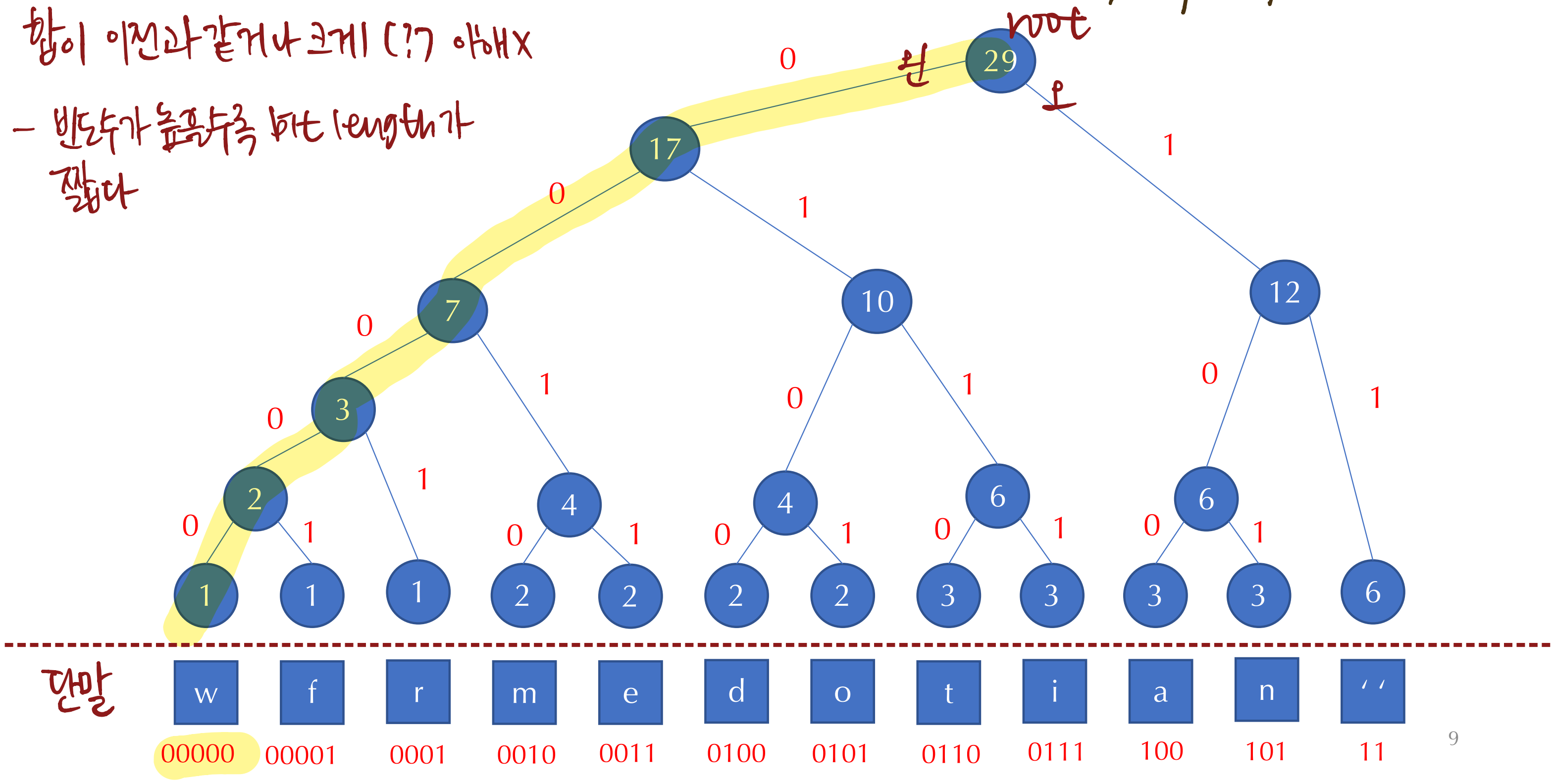
w	f	r	m	e	d	o	t	i	a	n	‘ ’
1	1	1	2	2	2	2	3	3	3	3	6

101/0101/0110 not

0110/0111/0010/0011 time

합이 이전과 같거나 크게 (?) 이해X

- 빈도수가 높을수록 bit length가 짧다



Huffman Coding Tree

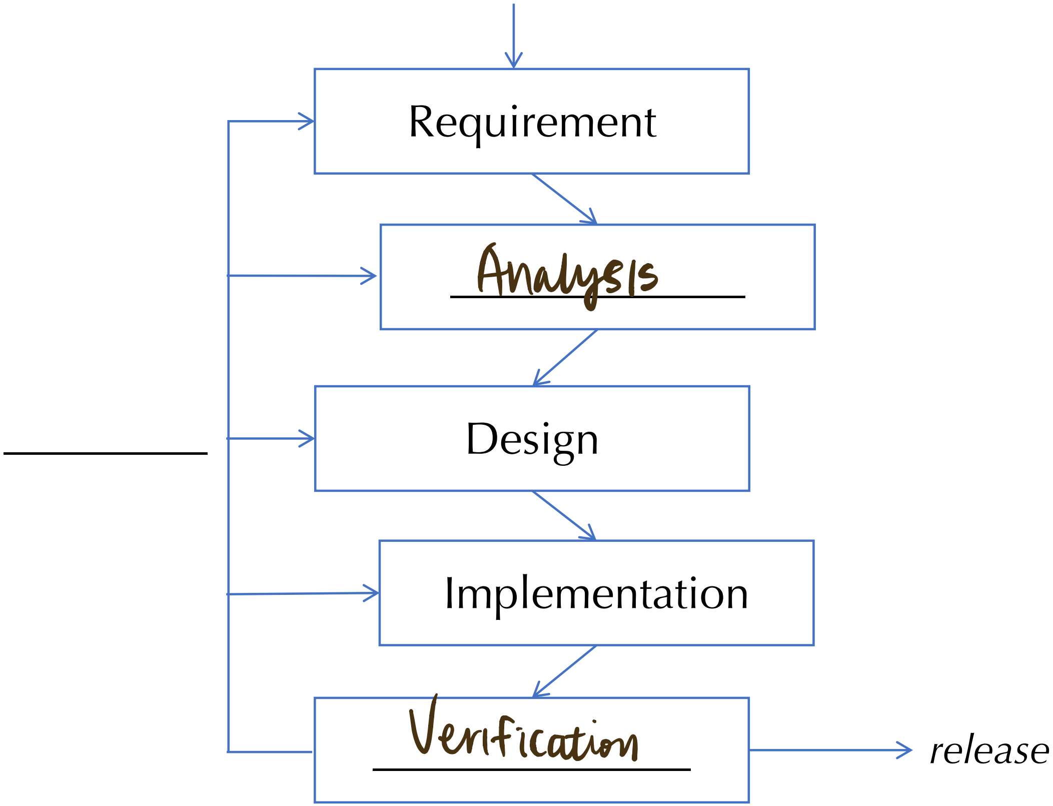
- By Non-compressed encoding
  - 29 chars => 29 bytes => 232 bits

	w	f	r	m	e	d	o	t	i	a	n	‘ ‘
Freq	1	1	1	2	2	2	2	3	3	3	3	6
bits	5	5	4	4	4	4	4	4	4	3	3	2
Freq *bits	5	5	4	8	8	8	8	12	12	9	9	12

- by Huffman coding
  - Sum (Freq * bits) = 100 bits
  - (232-100) / 232 => Save 57% space complexity

System Life Cycle

software 개발주기



System Life Cycle - Requirement

1. Requirement

목적의 문서화

- A set of specifications that define the objectives of a project
- Functions, Platform, Input, Output, Constraints, Users

추상적

- Project examples
  - Reservation System (flight ticket)
  - Billing System (mobile phone, utility)
  - Recruitment Agency

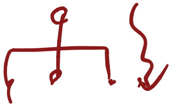
System Life Cycle - Analysis

2. Analysis

구체화

- From one big project into smaller modules
- devide and conquer Strategy
- ex) Ticket Reservation System : Clients, Contents, Payment

- Top-down approach (하향식)
  - Build the hierarchy from the primary goal to components
    - Program -> Subroutines -> Instructions



- Bottom-up approach (상향식)
  - Components comprises the entire system

System Life Cycle - Design

3. Design

- define objects and functions for each module

- ex) Ticket Reservation System
  - Clients : sign up/in, history
  - Contents
  - Payment Agency

System Life Cycle - Implementation

4. Implementation

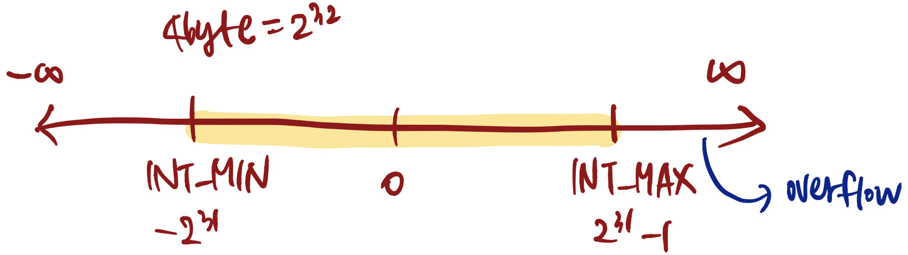
- Write executable codes for objects and algorithms
- Platforms : Web, Mobile App, Package, Embedded

System Life Cycle - Verification

5. Verification

- Correctness proof of algorithms and program test
- Testing
  - Black box test : through only inputs and outputs
  - White box test : internal codes as well as black box test 코드까지 고려
- Debugging
  - Version 3.1.1

Data Types



- Definition
  - a collection of objects and associated operation that act on those objects
- ex) Integer data type
  - objects = {INT_MIN, …, -2, -1, 0, 1, 2, … , INT_MAX}
  - operations = {+, -, *, /, %, …}
    - INT_MAX (4 bytes) = 2^{31} - 1 = 2,147,483,647
- Built-in data types
  - Basic type : char, int, float
  - Composite type : array, structure
  - Pointer type
  - User-defined data type : object type
  - Program 1.2 P11

Algorithm - Factorial

- Factorial
    - Calculate the factorial of an integer
    - 0! = 1! = 1
    - n! = 1 * 2 * … * (n - 1) * n
- Program 1.3

p12

main에서 불러내서 사용

```
int factorial (int n) {
    int s = 1;                    정의x
    if (   n<0       ) return -1;   // not defined
    if (  n==0   ) or (   n==1    ) return 1;
    for (i = 2 ; i <= n; i++)   ) 교재의 j loop에 해당
        s = s * i;

    return s;
}
```

Abstract Data Type

추상자료형

- User-defined data type (ADT) ex. class를 정의하는 것
  - Data type that abstracts objects that have similar properties and operations
  - Include Object specification and Operation implementation

    (properties) 속성 함수(method)
  - Object and instance (cf. class)
  - Ex) Bank account, Student, Subject

    object : Student (객체)

    instance : 학생개인
  - ex) OOPL 객체지향
    - C++, C#, Java, Python

    object oriented programming language

Bank Account ADT

수도코드

ADT 정의

```
class BankAccount
{
```

property(속성)

```
  int account_id;
  int account_type;
  char owner_name[20];
  float balance = 0;
```

생성자(constructor)

```
  init (name, type, money)
  {
     owner_name = name;
     account_type = type;
     deposit(money);
  }
```

information hiding

공개된 interface로 접근 (객체지향 programming)

method

```
  deposit(amount)
  {
     balance = balance + amount;
  }
```

method

```
  withdraw(amount)
  {
     balance = balance – amount;
  }
}
```

instance create

```
BankAccount myaccount ("Kim", "checking", 1000);
```

자연수 → 양의 int

Natural Number ADT

structure NaturalNum is

Objects : an ordered subrange of integers [0, INT_MAX]

Functions: for all x, y ∊ NaturalNum, TRUE, FALSE ∊ Boolean and
where +, -, <, = are integer operations

is defined as

NaturalNum Zero( ) ::= 0 0 반환하라

Boolean Is_Zero(x) ::= if (x) return FALSE else return TRUE 0 인지?

혹은그냥 error출력.

NaturalNum Add (x, y) ::= if ((x + y) <= INT_MAX) return x + y else return INT_MAX

NaturalNum Subtract(x, y) ::= if (x < y) return 0 else return x – y

하한선반환

Boolean Equal(x, y) ::= if (x == y) return TRUE else return FALSE

NaturalNum Successor(x) ::= if (x == INT_MAX) return x else return x + 1

x다음숫자가뭐냐 상한선 반환

end NaturalNum

자동차라는 추상자료형