TOPCIT 학습법 특강: 소프트웨어 개발 영역

Sep 23, 2024

Myoungsung You (famous77@kaist.ac.kr)

Network and System Security (NS^2) Lab

School of Electrical Engineering at KAIST

NS2 Network and System Security Laboratory KAIST

Index

강사 소개

유명성, Myoungsung You

famous77@kaist.ac.kr

- KAIST 전기및전자공학부 박사과정
- KAIST 전산학부 석사과정 졸업
- 충북대학교 컴퓨터공학과 졸업
- **Research interests**: Cloud Computing, Programmable Data Plane, and Network Security
- **Recent publications**:
  - (**IEEE TCC volume 12, Issue 3**) *Hyperion*: Hardware-based High-performance and Secure System for Container Networks, main author
  - (**IEEE ICDCS 24**) *HardWhale*: A Hardware-Isolated Network Security Enforcement System for Cloud Environments, main author
  - (**ACM SoCC 23**) *HELIOS*: Hardware-assisted High-performance Security Extension for Cloud Networking, main author
  - (**ACM SoCC 23**) *Cryonics*: Trustworthy Function-as-a-Service using Snapshot-based Enclaves, 2nd author

1. TOPCIT 이란?

1. TOPCIT

TOPCIT (Test Of Practical Competency in IT)

- 실무 중심의 IT 역량 평가를 위한 검정 시험으로 문제 **은행 방식**이며 **폭 넓은 시험 범위**를 갖음
  - 소프트웨어 개발뿐만 아니라 프로젝트 관리, 데이터베이스, 네트워크, 보안 등 다양한 주제를 다룸
  - 단답형, 서술형, 실기형 등 다양한 유형의 문제가 출제됨
- 시험 방식: CBT (Computer Base Testing)
- 시험 시간:
  - 입실: 09:10까지, 시험 시간: 09:30 ~ 12:00 (2시간 반)

1. TOPCIT

TOPCIT 시험 평가 영역

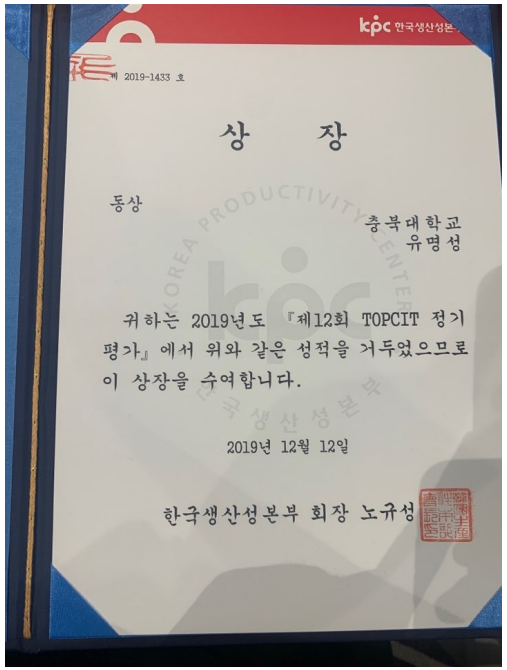
- **기술 영역**
  - 소프트웨어 개발 (09.23, 18:00~18:50)
  - 데이터 이해와 활용 (09.23, 19:00~19:50)
  - 시스템 아키텍처 이해와 활용 (09.25, 18:00~18:50)
  - 정보보안 이해와 활용 (09.25, 19:00~19:50)

- **비즈니스 영역**
  - IT 비즈니스 및 윤리의 이해 (10.02, 18:00~18:50)
  - 프로젝트관리와 테크니컬 커뮤니케이션의 이해 (10.02, 18:00~18:50)

1. TOPCIT

고득점 수상 특전

- 각 대학별 최고 득점자: 특별상 (대학총장상, 중복 수상 가능)
- 전체 최고 득점자 1 ~ 4 등: 해외교육 프로그램 참여 기회 제공 (실리콘 벨리 기업 및 CES) 및 상장 수여



2. 부상(해외교육프로그램) 안내

가. 대상 : 2024년 정기평가 대학생 대·금·은·동상 및 군인 수상자 등 10명

나. 일정 : 2025년 1월 예정(세부 내용은 대상자 추후 별도 안내)

다. 내용 : Consumer Electronics Show (CES, 소비자 가전 박람회) 2025 참관

* 위 사항은 진행 상황에 따라 변동될 수 있습니다.
* 개인사정 등 해외교육프로그램 참여 불가한 경우에 대체로 지급되는 부상은 없으며 참여기회는 타인에게 임의로 양도 불가합니다.

1. TOPCIT

고득점 수상 특전

2. 소프트웨어 개발 영역 학습 방법

2. 소프트웨어 개발 영역 학습 방법

소프트웨어 개발 영역의 평가 목표

- **소프트웨어 개발을 위한 방법론 및 배경지식 이해**
  - 소프트웨어 개발 프로세스
  - 자료구조 및 알고리즘

- **소프트웨어의 분석/설계/구현/유지보수 업무 수행 지식 습득**
  - 객체지향 설계
  - 소프트웨어 유지관리
  - 소프트웨어 테스팅 / 리펙토링

2. 소프트웨어 개발 영역 학습 방법

소프트웨어 개발 영역 특징

- TOPCIT 시험에서 가장 넓은 범위와 높은 배점을 가진 영역
- 문항수: 17, 배점: 260점

객관식 (19%)
4지 선다, 단일 선택형 및 복수 선택형 문제

단답형 (12%)
1개의 단어나 50자 이내의 짧은 문장으로 대답할 수 있는 문제

서술형 (21%)
450자 이내로 자신의 의견이나 지식을 대답할 수 있는 문제

수행형 (48%)
프로그램 코딩, 설계 등 실제적인 작업을 수행할 수 있도록 구성된 실기 문제

2. 소프트웨어 개발 영역 학습 방법

효율적인 TOPCIT 준비 방법

- 기본 학습 자료 활용
  - TOPCIT 에센스, 샘플 문제 및 기출 문제
  - TOPCIT 온라인 학습 센터 강의 수강 (비전공자 추천)

- 문제풀이 연습
  - 정보처리기사 (산업기사) 기출 문제 활용
    - 1과목: 소프트웨어 설계, 4과목: 프로그래밍 언어 활용
  - 정보보안기사 (산업기사) 기출 문제 활용
  - 한경닷컴 기출 문제 활용

3. 소프트웨어 개발 영역 주요 개념

자료구조와 알고리즘

- **자료구조**
  - 선형 자료구조
  - 비선형 자료구조
- **알고리즘**
  - 알고리즘 표기법
  - 정렬 알고리즘
  - 탐색 알고리즘

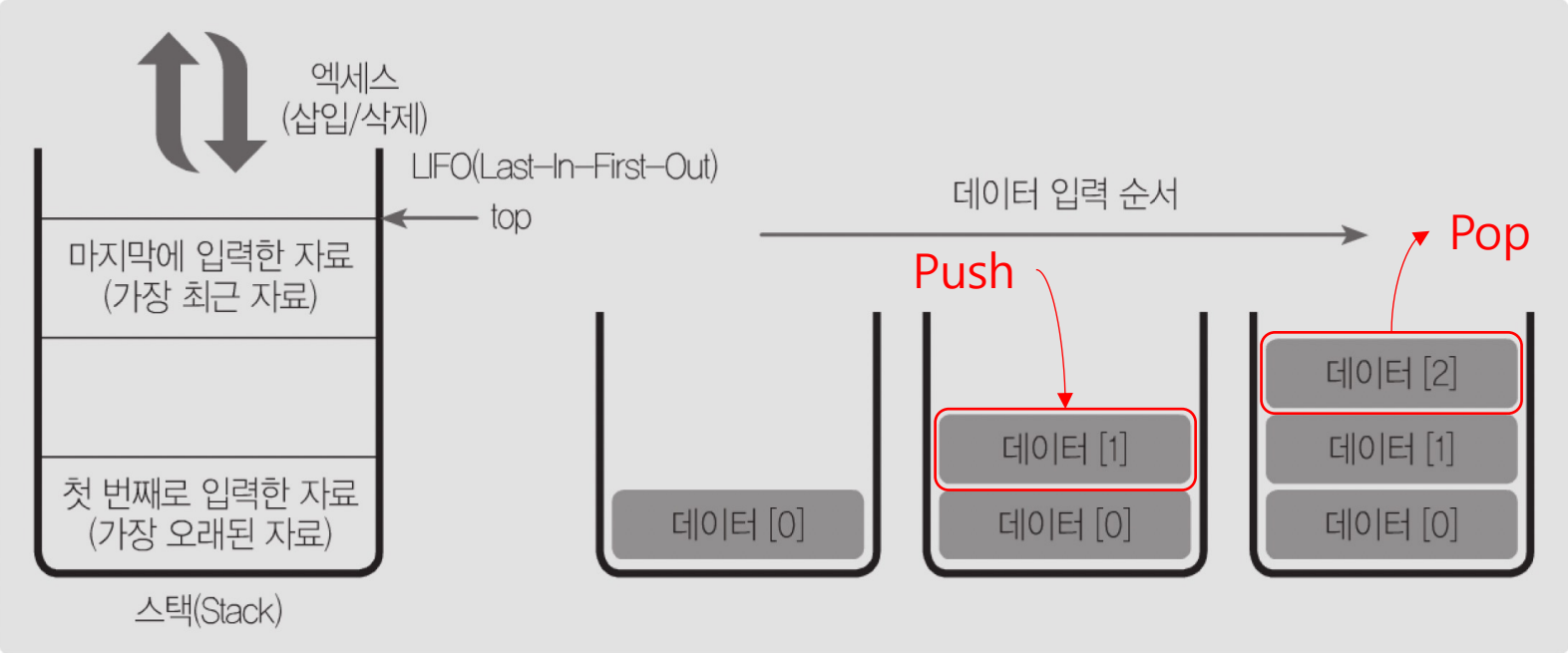
자료구조의 분류

- **선형(Linear)** 자료구조: 데이터가 일렬로 연결되어 있는 자료구조
  - 순차(Sequential) 자료구조: 배열
  - 연결(Linked) 자료구조: 연결 리스트, 더블 연결 리스트

- **비선형(Non-linear)** 자료구조: 데이터가 계층적이거나 그래프 형태로 연결되어 있는 자료구조
  - 그래프, 트리

스택 (stack)

- 선형 자료구조로 LIFO 형식으로 동작, 데이터의 입/출력이 한 곳 (top)에서 이루어짐
- 백 트래킹 알고리즘, 함수 콜 스택 등에 활용됨



큐 (queue)

- 선형 자료구조로 FIFO 형식으로 동작, 데이터의 입력은 rear (tail)에서 출력은 front (head)에서 이루어짐
- 다양한 스케쥴링 알고리즘에 사용됨 (우선순위 큐 등)

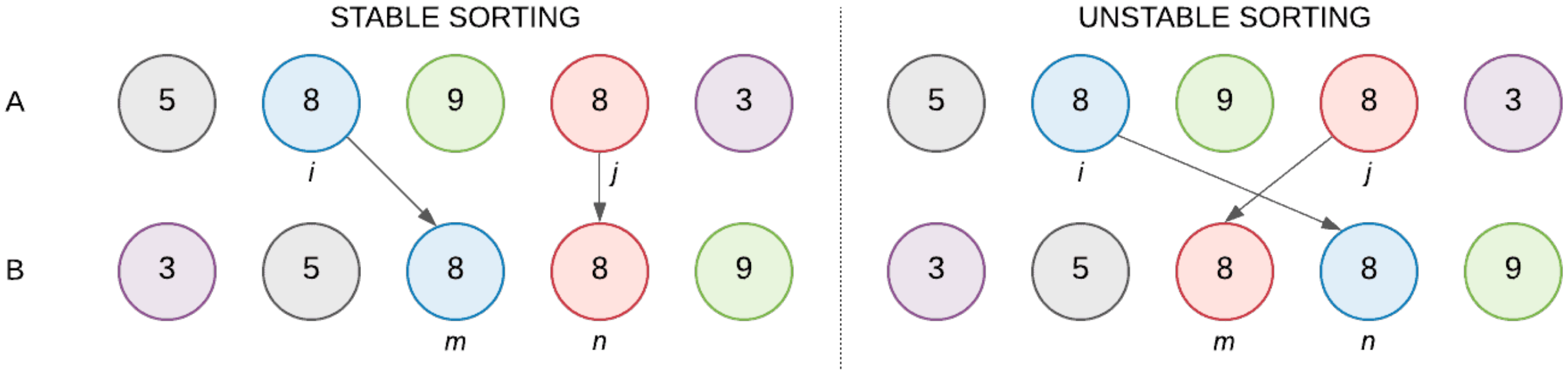
정렬 알고리즘

정렬 알고리즘의 분류

- **내부 정렬**: 정렬할 데이터를 모두 메인 메모리에 올려서 정렬
- **외부 정렬**: 데이터를 여러 개의 서브 데이터로 나눈 뒤, 일부 서브 데이터를 메인 메모리에서 정렬 (내부정렬)하고 보조 기억장치에서 서브 데이터들을 병합
- **안정 정렬**: 정렬되지 않은 상태에서 같은 키 값을 가진 원소의 순서가 정렬 후에도 보존됨
- **불안정 정렬**: 정렬되지 않은 상태에서 같은 키 값을 가진 원소의 순서가 정렬 후에 유지되지 않음

정렬 알고리즘

- 안정 정렬/불안정 정렬



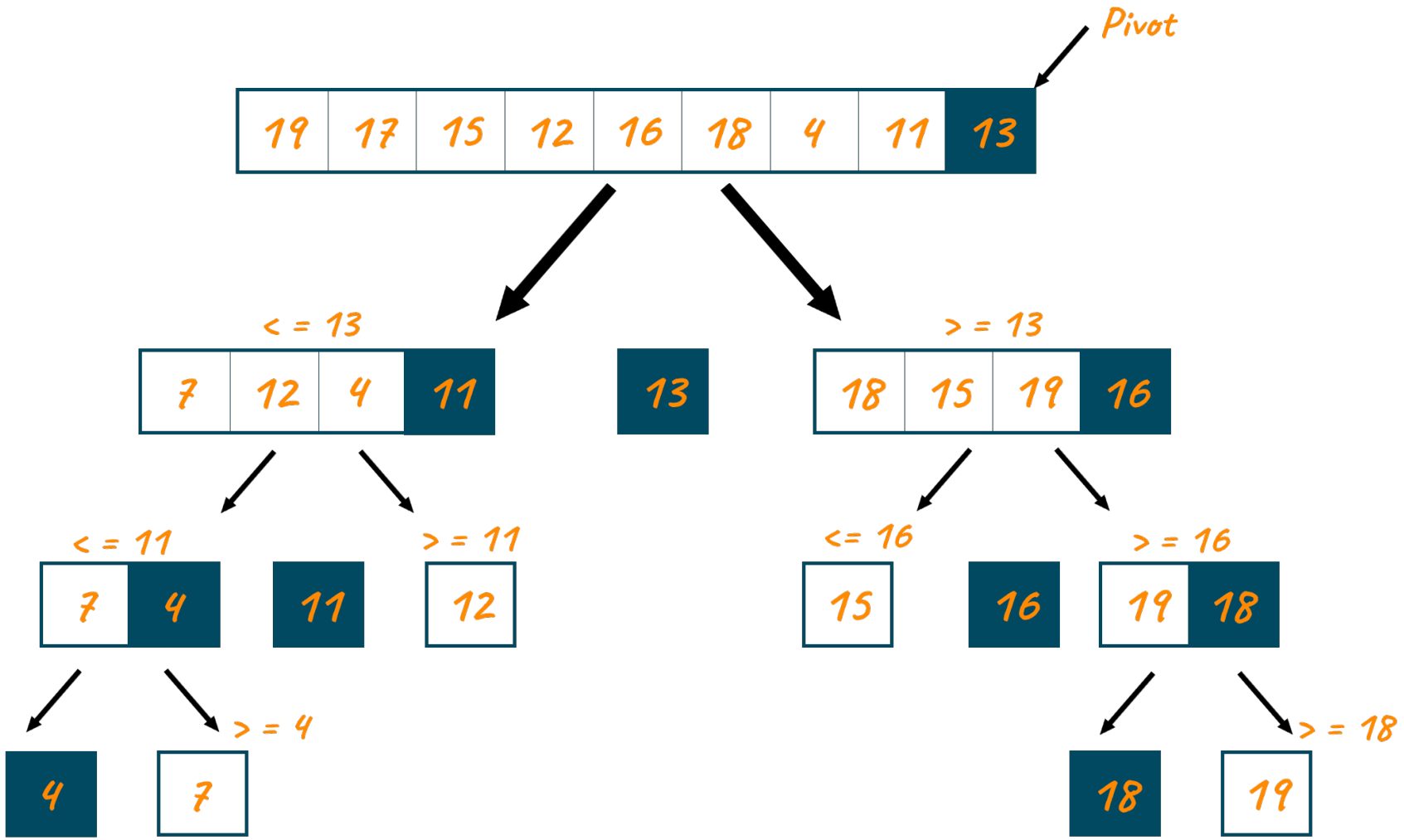
정렬 알고리즘

Quick Sort

- 분할 정복 기반의 정렬 알고리즘
  1. 리스트에서 pivot을 선택
  2. Pivot을 기준으로 리스트를 2개로 분할: Pivot보다 작은 원소는 왼쪽, 큰 원소는 오른쪽으로 이동
  3. Pivot을 제외한 왼쪽/오른쪽 부분 리스트에 대해서 다시 동일한 과정을 반복
  4. 부분 리스트가 더 이상 분할이 불가능할 때 까지 반복
- 불안정(Unstable) 정렬 알고리즘

정렬 알고리즘

- Quick Sort



정렬 알고리즘

Quick Sort 구현

```
3   int partition(int arr[], int low, int high) {
4       int pivot = arr[high];
5       int i = low - 1;
6
7       for (int j = low; j < high; j++) {
8           if (arr[j] < pivot) {
9               i++;
10              int temp = arr[i];
11              arr[i] = arr[j];
12              arr[j] = temp;
13          }
14      }
15
16      int temp = arr[i + 1];
17      arr[i + 1] = arr[high];
18      arr[high] = temp;
19
20      return i + 1;
21  }
```

```
23  void quickSort(int arr[], int low, int high) {
24      if (low < high) {
25          int pi = partition(arr, low, high);
26
27          quickSort(arr, low, pi - 1);
28          quickSort(arr, pi + 1, high);
29      }
30  }
```

```
32  int main() {
33      int arr[] = {19, 17, 15, 12, 16, 18, 4, 11, 13};
34      int n = sizeof(arr) / sizeof(arr[0]);
35
36      quickSort(arr, 0, n - 1);
37
38      printf("Sorted array: ");
39      for (int i = 0; i < n; i++) {
40          printf("%d ", arr[i]);
41      }
```

검색 알고리즘

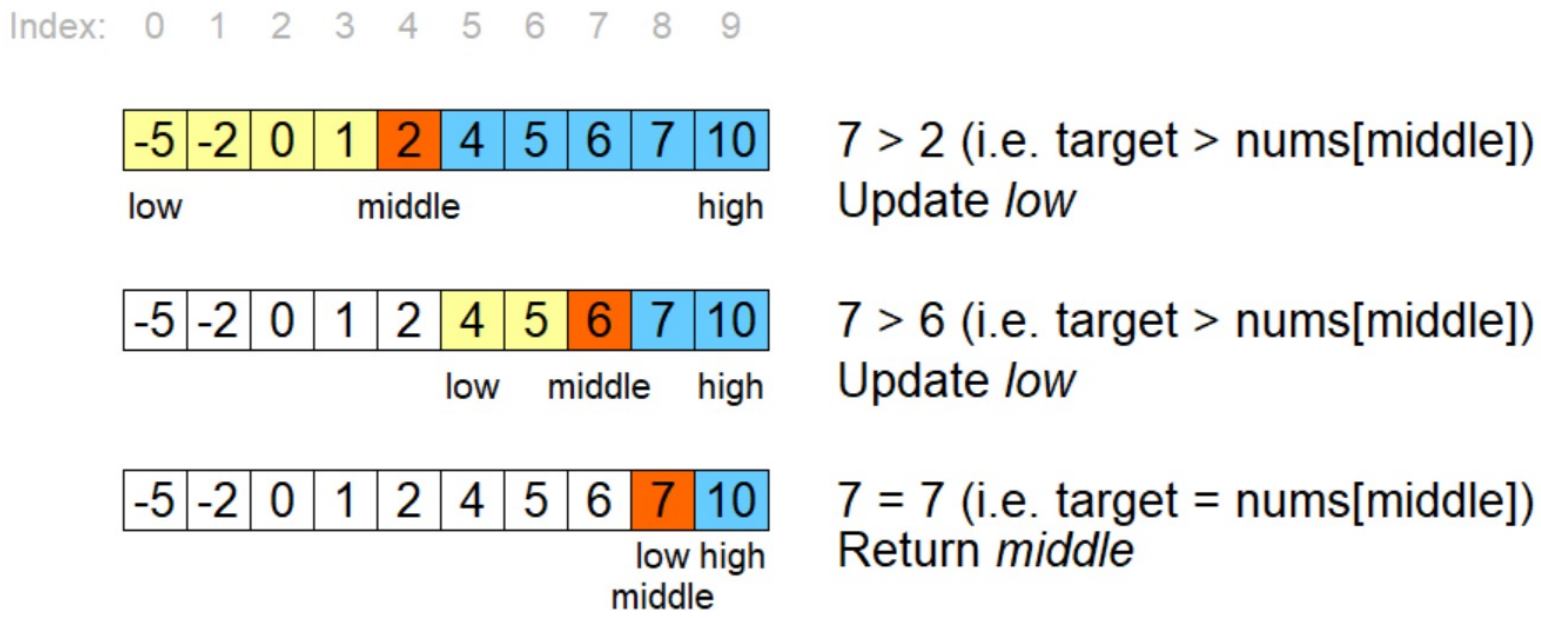
검색 알고리즘의 종류

- **선형검색**: 정렬되어 있지 않은 데이터를 처음부터 순서대로 검색, O(N)
- **제어검색**: 정렬되어 있는 데이터를 특정 알고리즘에 따라 검색, O(logN)
- **해싱**: 해시 키를 사용해 데이터를 검색, O(1)

검색 알고리즘

이진 탐색 (binary search)

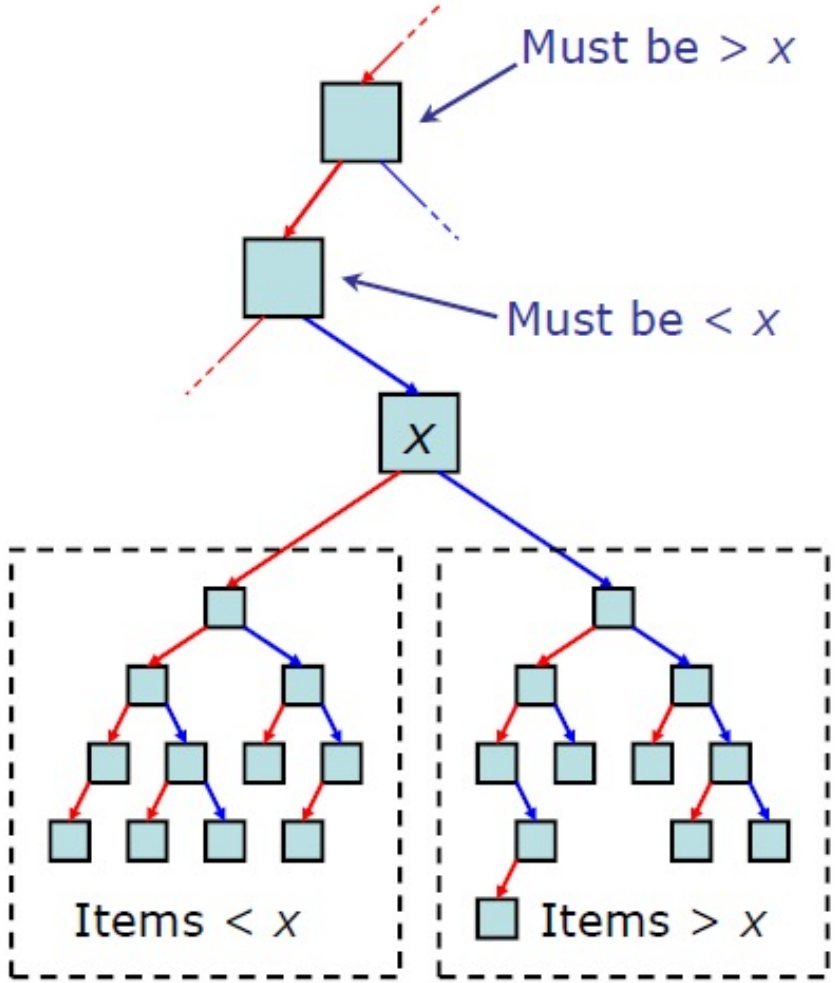
- 중간값 (middle)을 정해 중간부터 값을 찾아가며, 비교할 때 마다 탐색 범위를 ½로 줄여나가는 탐색 알고리즘
- 정렬된 데이터에만 사용 가능, O(logN)



검색 알고리즘

이진 탐색 트리 (binary search tree)

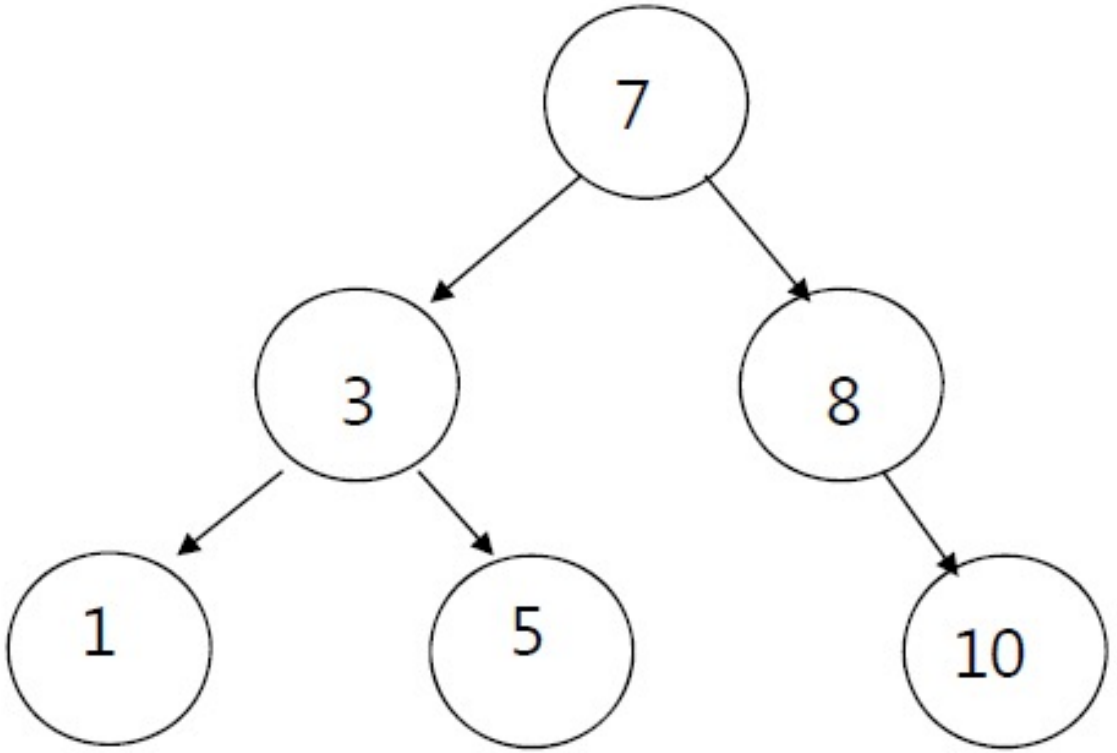
- 이진 탐색과 연결 리스트를 조합한 자료구조
- 이진 탐색의 빠른 탐색 + 빈번한 자료의 추가/삭제 용이
- 탐색/삽입/삭제: O(h)
- 이진 탐색 트리 조건
  - 왼쪽 서브트리는 해당 노드보다 작은 값의 노드만 존재
  - 오른쪽 서브트리는 해당 노드보다 큰 값의 노드만 존재
  - 중복 노드는 존재하지 않음
  - 각 서브트리 또한 이진 탐색 트리



검색 알고리즘

이진 탐색 트리 (탐색)

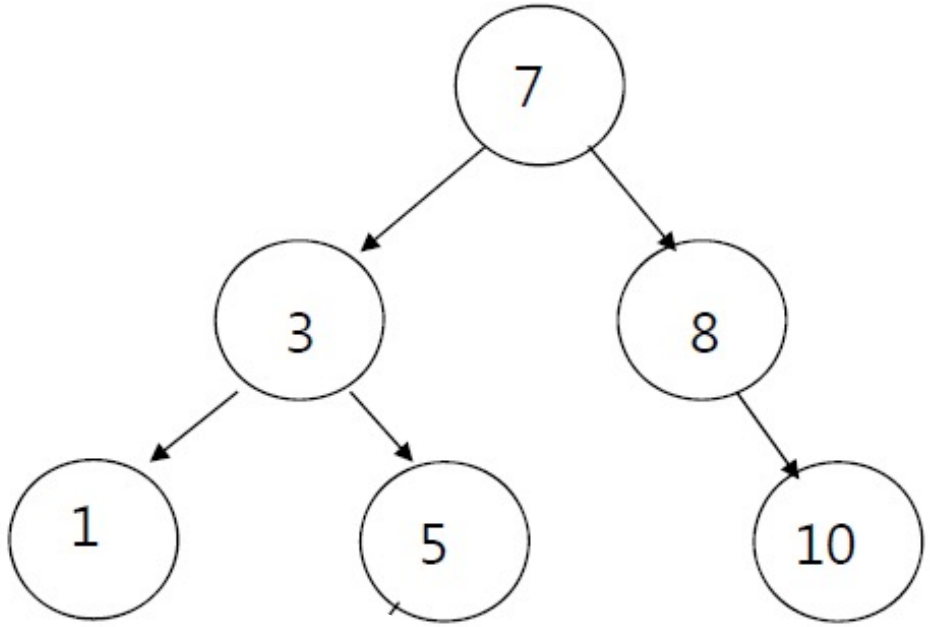
- 루트 노드부터 트리를 따라 내려오며 탐색 범위를 줄임
- 10을 탐색: 7, 8, 10
- 4를 탐색: 7, 3, 5



검색 알고리즘

이진 탐색 트리 (삽입)

- 이진 트리의 조건을 만족하는 위치에 새로운 원소 삽입
- 반드시 잎새 노드 (leaf node)에 삽입
- 4 삽입: 7 -> 3 -> 5 -> 4



검색 알고리즘

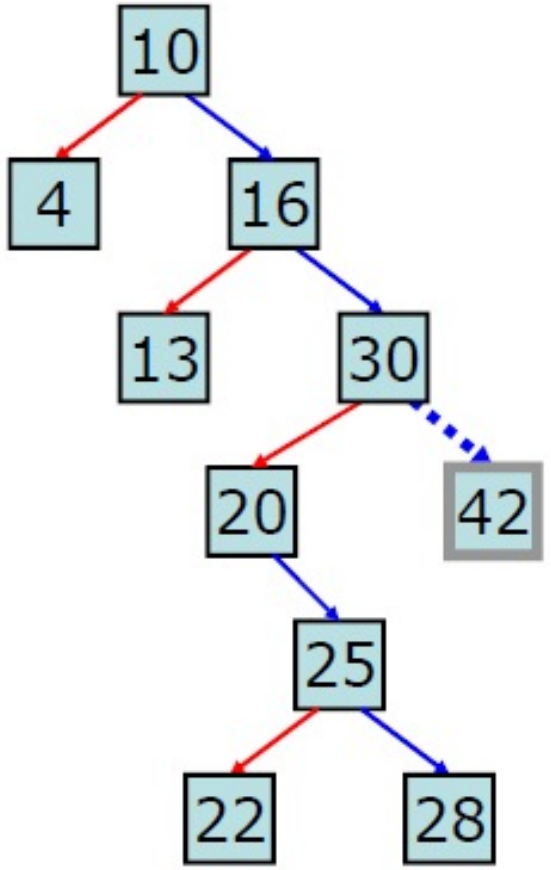
이진 탐색 트리 (삭제)

- 삭제 후에도 이진 트리의 조건을 만족하도록 보장
- 삭제할 노드의 자식 노드의 유무에 따라 동작이 상이
  - Case 1: 자식 노드가 없는 경우
  - Case 2: 자식 노드가 1개인 경우
  - Case 3: 자식 노드가 2개인 경우

검색 알고리즘

이진 탐색 트리 (삭제 Case 1)

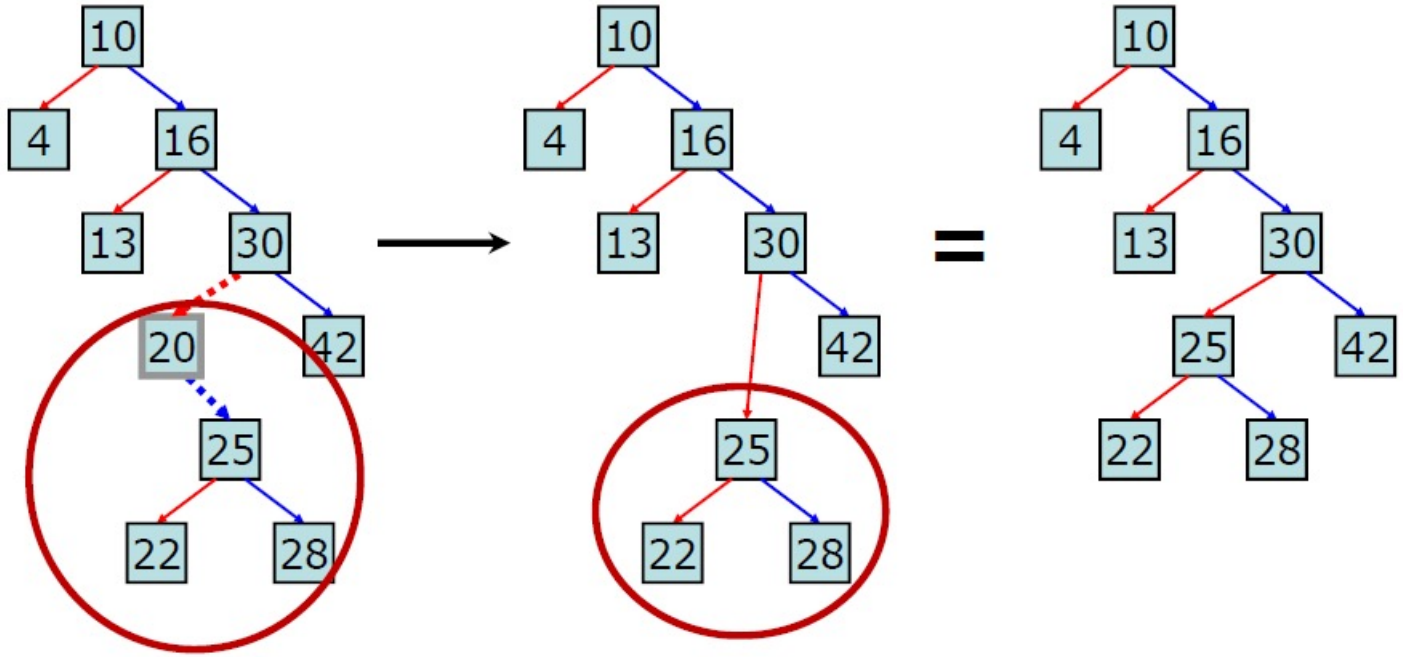
- 단순히 해당 노드를 삭제



검색 알고리즘

이진 탐색 트리 (삭제 Case 2)

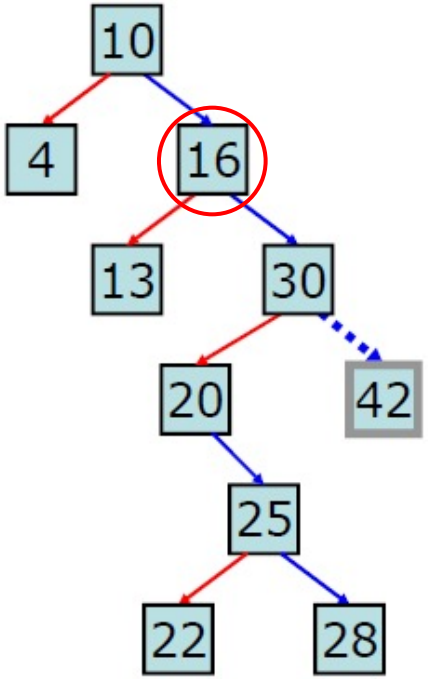
- 해당 노드 삭제 후 부모 노드와 자식 노드 연결
- 해당 노드의 하위 트리 원소들은 모두 해당 노드의 부모 보다 작거나 (왼쪽 자식) 큼 (오른쪽 자식)



검색 알고리즘

이진 탐색 트리 (삭제 Case 3)

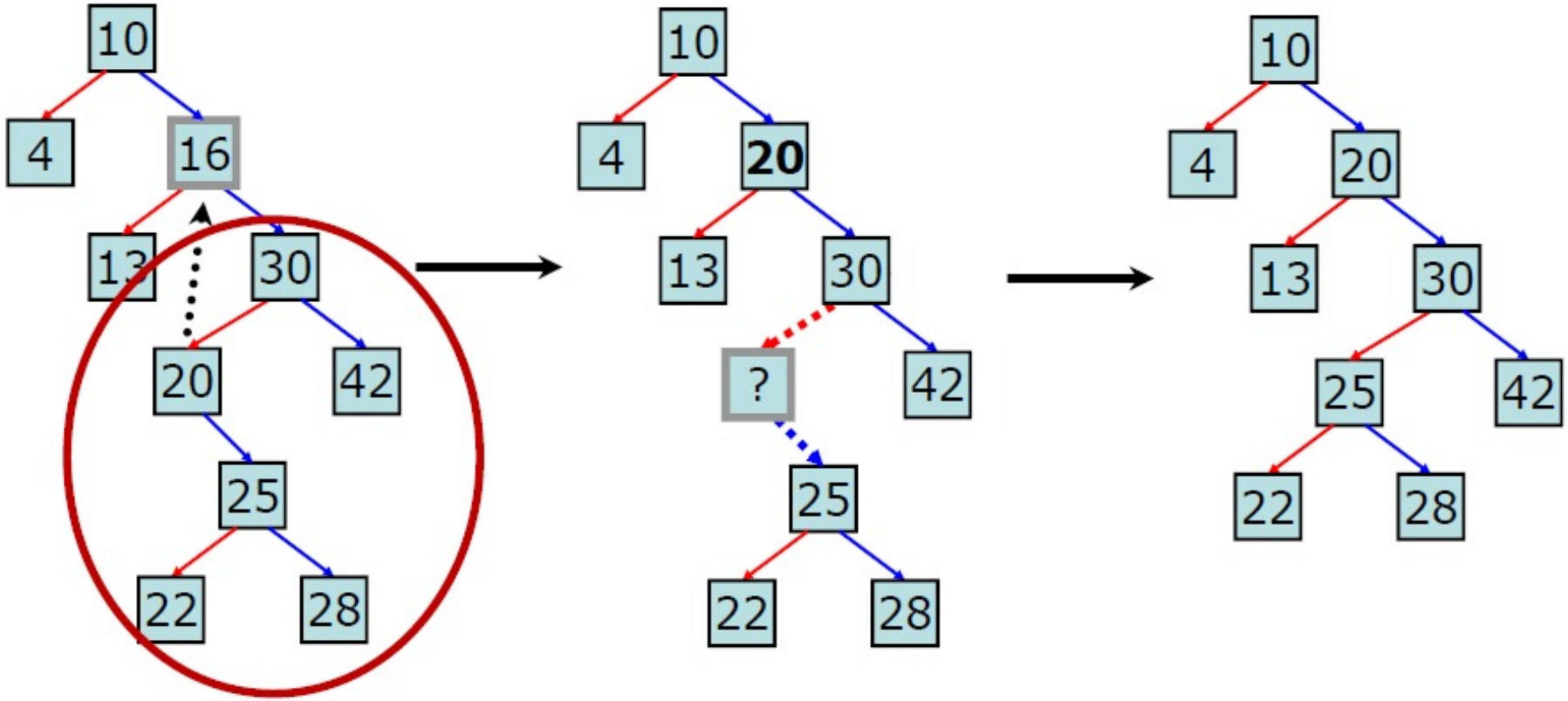
- 트리의 구조 변경을 최소화하면서 노드를 제거
- 삭제 알고리즘
  - 삭제 대상 노드의 오른쪽 서브 트리를 순회
  - Successor 노드 선정
    - **삭제 대상 노드의 오른쪽 서브 트리 최소값**
  - Successor 노드의 값을 삭제할 노드에 복사
  - Successor 노드 제거
    - Case 1 or case 2



4, 10, 13, 16, **20**, 22, 25, 28, 30, 42

검색 알고리즘

- 이진 탐색 트리 (삭제 Case 3)



객체지향 설계

Object Oriented Programming (OOP)

- 프로그램을 객체 (object)의 집합으로 보고, 객체 간의 상호작용으로 프로그램을 구현하는 프로그래밍 패러다임
- 클래스 (class): 객체 생성을 위한 틀
  - 맴버 변수 (attribute): 객체의 상태를 나타내는 데이터
  - 맴버 함수 (method): 객체의 행동을 나타내는 함수

- 장점: 코드 재사용성 증가, 유지 보수 용이, 코드의 간결성 증가
- 단점: 프로그램 처리 시간 증가, 프로그램 설계 시간 증가

객체지향 설계

OOP의 4가지 특징

- 캡슐화 (Encapsulation)
  - 데이터(속성)와 데이터를 처리하는 함수(메서드)를 하나로 묶는 것을 의미
  - 객체의 상세한 내용이 외부에 은닉되고, 정해진 인터페이스 (getter and setter)만 공개됨
    - 접근제어자

접근 제한	적용 대상	접근할 수 없는 클래스
public	클래스, 필드, 생성자, 메소드	없음
protected	필드, 생성자, 메소드	자식 클래스가 아닌 다른 패키지에 소속된 클래스
default	클래스, 필드, 생성자, 메소드	다른 패키지에 소속된 클래스
private	필드, 생성자, 메소드	모든 외부 클래스

객체지향 설계

- OOP의 4가지 특징 (캡슐화 예제)

```
4   class EncapsulationExample {
5   private:
6       int privateData;
7   protected:
8       int protectedData;
9   public:
10      int publicData;
11
12      EncapsulationExample() {
13          privateData = 10;
14          protectedData = 20;
15          publicData = 30;
16      }
17      int getPrivateData() {
18          return privateData;
19      }
20      int getProtectedData() {
21          return protectedData;
22      }
23      int getPublicData() {
24          return publicData;
25      }
26  };
```

```
28  int main() {
29      EncapsulationExample obj;
30
31      obj.privateData = 10;
32      obj.protectedData = 20;
33      obj.publicData = 30;
34      cout << "Public Data: " << obj.getPublicData() << endl;
35      return 0;
36  }
```

객체지향 설계

OOP의 4가지 특징

- 상속 (Inheritance)
  - 기존의 클래스의 특성 (맴버 변수와 맴버 함수)을 다른 클래스에게 물려주는 것
  - 코드의 재 사용성을 높이고, 기존 코드를 확장하여 새로운 기능을 추가할 수 있음
  - 부모 클래스와 자식 클래스가 있으며, 자식 클래스는 부모 클래스의 모든 특성을 상속받음
    - 이를 바탕으로 자신만의 추가 기능을 작성 가능

객체지향 설계

- OOP의 4가지 특징 (상속 예제)

```
4   class Animal {
5   public:
6       void sound() {cout << "Some sound" << endl;}
7   };
8
9   class Dog : public Animal {
10  public:
11      void bark() {cout << "Woof! Woof!" << endl;}
12  };
13
14  int main() {
15      Dog myDog;
16      myDog.sound();
17      myDog.bark();
18
19      return 0;
20  }
```

객체지향 설계

OOP의 4가지 특징

- 다형성 (Polymorphism)
  - 같은 이름의 메서드나 연산자가 서로 다른 클래스에서 다르게 동작하는 것
  - 코드의 유연성 및 재사용성, 가독성을 높일 수 있음
  - 함수 오버로딩, 상속과 함수 오버라이딩을 통해 구현됨

객체지향 설계

함수 오버로딩 (overloading)

- 같은 이름을 가진 함수 또는 연산자를 여러 개 정의하는 것
- 오버로딩된 함수들은 서로 다른 매개변수 타입, 개수, 순서를 가질 수 있음
- 컴파일러는 호출된 함수의 ***시그니처***(이름, 매개변수 유형)에 따라 오버로딩된 함수를 선택
- 리턴 타입은 함수 시그니처에 포함되지 않음

객체지향 설계

- 함수 오버로딩 (overloading)

```
4   int add(int a, int b) {
5       return a + b;
6   }
7
8   double add(double a, double b) {
9       return a + b;
10  }
11
12  int main() {
13      int intResult = add(3, 4);
14      double doubleResult = add(3.5, 4.5);
15
16      cout << "Integer addition result: " << intResult << endl;
17      cout << "Double addition result: " << doubleResult << endl;
18
19      return 0;
20  }
```

함수 오버라이딩 (overriding)

- 부모 클래스에서 정의한 함수를 자식 클래스에서 재정의하는 것
- 자식 클래스에서 동일한 부모 클래스의 함수와 ***동일한 시그니처***를 가진 함수를 작성
- 주로 가상 함수와 함께 사용되며, 런 타임에 가상 함수 테이블을 통해 적절한 함수에 매핑됨

객체지향 설계

- 함수 오버라이딩 (overriding)

```
4   class Animal {
5   public:
6       virtual void sound() {cout << "Some sound" << endl;}
7   };
8
9   class Dog : public Animal {
10  public:
11      void sound() override {cout << "Woof! Woof!" << endl;}
12  };
13
14  class Cat : public Animal {
15  public:
16      void sound() override {cout << "Meow! Meow!" << endl;}
17  };
```

```
19  int main() {
20      Animal* animals[2];
21      animals[0] = new Dog();
22      animals[1] = new Cat();
23
24      for (int i = 0; i < 2; i++) {
25          animals[i]->sound();
26      }
27
28      return 0;
29  }
```

객체지향 설계

OOP의 4가지 특징

- 추상화 (Abstraction)
- 복잡한 시스템을 간단한 개념으로 표현하는 것을 의미
- 필요한 속성과 기능만을 표현하여 객체의 복잡성을 줄일 수 있음
  - 순수 가상 함수: 구현이 없는 함수로 하위 클래스에서 반드시 오버라이딩 해야함
  - 추상 클래스: 1개 이상의 순수 가상 함수를 갖는 클래스로, 객체 생성 불가

객체지향 설계

- OOP의 4가지 특징 (추상화 예제)

```
4   class Shape {
5   public:
6       virtual void draw() = 0;
7   };
8
9   class Circle : public Shape {
10  public:
11      void draw() override {cout << "Drawing Circle" << endl;}
12  };
13
14  class Square : public Shape {
15  public:
16      void draw() override {cout << "Drawing Square" << endl;}
17  };
```

```
19  int main() {
20      Circle circle;
21      Square square;
22
23      circle.draw();
24      square.draw();
25
26      return 0;
27  }
```

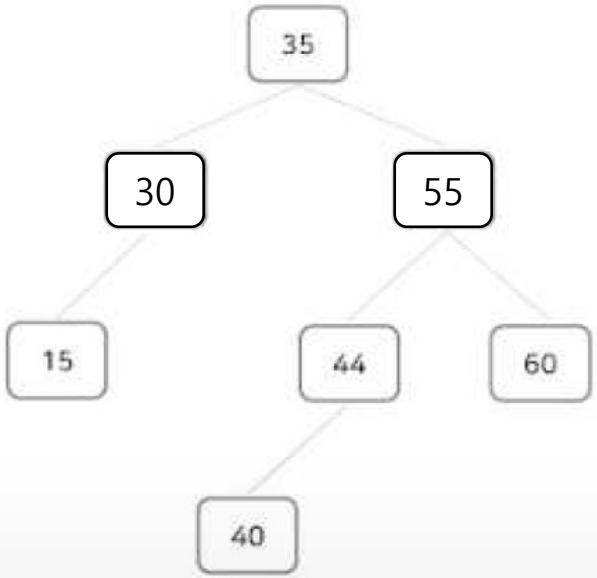
4. 기출문제 풀이

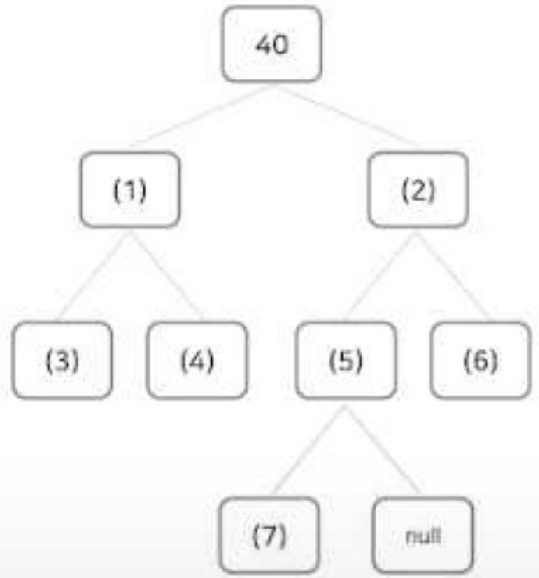
4. 기출문제 풀이

Q1

다음은 이진탐색트리이다. 왼쪽 트리의 35를 삭제하고, 이진트리의 조건에 따라 32를 추가한 후의 결과가 오른쪽 트리이다. 괄호의 번호 순서대로 알맞은 값을 적으시오.
(값이 없을 경우 null로 표시할 것)

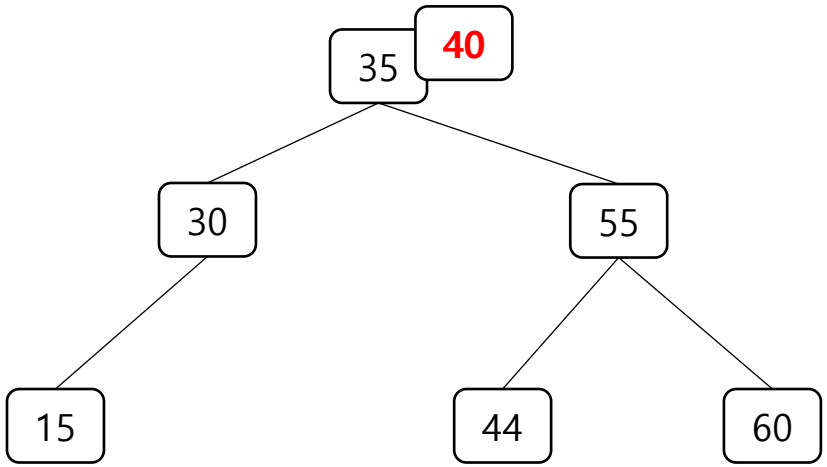
[보기]

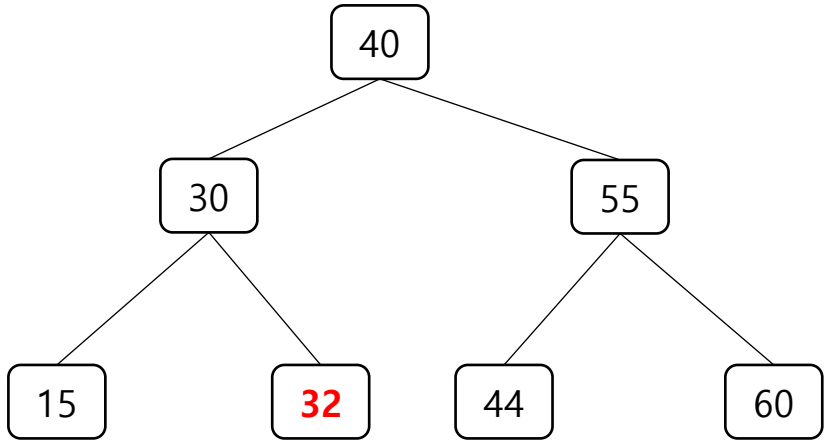




4. 기출문제 풀이

정답





<35 삭제>

30, 55, 15, 32, 44, 60, null

<32 추가>

4. 기출문제 풀이

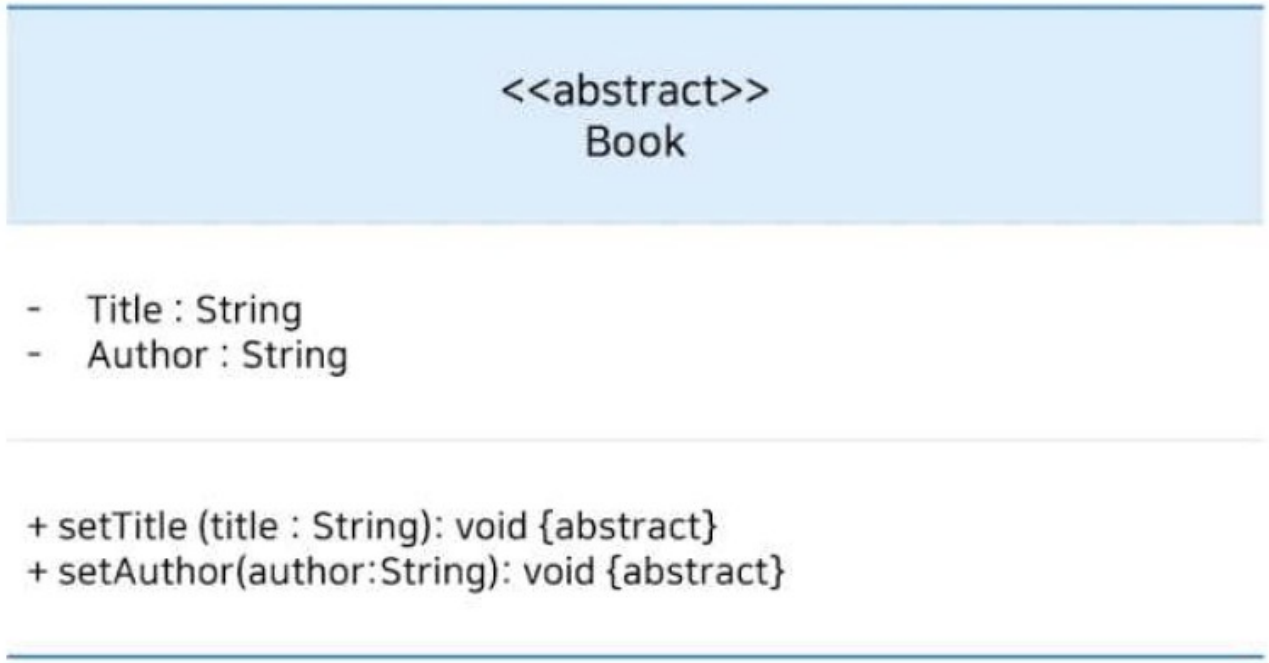
Q2

다음 클래스 다이어그램은 Book에 대한 추상클래스이다. 다음을 참고하여 아래의 물음에 답하시오.

1) [보기]에서 setTitle() 메소드의 접근제한 속성에 대하여 설명하시오(10점).

2) [보기]의 클래스 다이어그램을 JAVA 언어 코드로 작성하시오(30점).

[보기]



4. 기출문제 풀이

Q2

다음 클래스 다이어그램은 Book에 대한 추상클래스이다. 다음을 참고하여 아래의 물음에 답하시오.

1) [보기]에서 setTitle() 메소드의 접근제한 속성에 대하여 설명하시오(10점).

2) [보기]의 클래스 다이어그램을 JAVA 언어 코드로 작성하시오(30점).

1) "+" Public 접근 제한 속성으로, 객체 외부에서도 접근 가능

2) 구현 예시

```
1   Public abstract class Book{
2       private String title;
3       private String author;
4
5       public abstract void setTitle(String title);
6       public abstract void setAuthor(String author);
7   }
```

4. 기출문제 풀이

소프트웨어 개발 객관식

- **UML 다이어그램 중 순차 다이어그램(Sequence Diagram)에 대한 설명으로 틀린 것은?**

  ① 객체 간의 동적 상호작용을 시간적 개념을 중심으로 모델링한다.

  ② 각 객체 간의 메시지 교환을 통해서 시스템의 행위를 표현한다.

  ③ 객체 간의 제어 흐름을 보여주는 다이어그램이다.

  ④ 객체 간의 관계를 표현하는 정적 다이어그램이다.

- 정답: **4**, 순차 다이어그램은 동적 다이어그램으로, 객체 간의 상호작용을 시간 순서대로 표현한다. 정적 다이어그램은 클래스 다이어그램이나 객체 다이어그램에 해당한다.

4. 기출문제 풀이

소프트웨어 개발 객관식

- **객체지향 개념에서 다형성(Polymorphism)에 대한 설명으로 옳은 것은?**

  ① 하나의 객체가 여러 가지 타입을 가질 수 있는 것을 의미한다.

  ② 상위 클래스의 특성을 하위 클래스가 물려받는 것을 의미한다.

  ③ 필요한 정보만 외부에 노출하는 것을 의미한다.

  ④ 데이터와 데이터를 처리하는 함수를 하나로 묶는 것을 의미한다.

- 정답: **1**, 다형성은 같은 이름의 메서드가 클래스나 객체에 따라 다르게 동작할 수 있는 능력을 의미한다.

4. 기출문제 풀이

소프트웨어 개발 객관식

- **애자일(Agile) 프로세스 모델에 대한 설명으로 틀린 것은?**

  ① 고객의 요구사항 변화에 유연하게 대응할 수 있다.

  ② 시스템 개발 주기를 짧게 반복하면서 개발을 진행한다.

  ③ 개발 초기에 요구사항을 명확히 정의하는 것이 중요하다.

  ④ 고객과 개발자 사이의 의사소통이 매우 중요하다.

- 정답: **3**, 애자일 방법론은 변화에 유연하게 대응하는 것을 중요시한다. 따라서 초기에 모든 요구사항을 명확히 정의하는 것보다는, 개발 과정에서 지속적으로 요구사항을 수정하고 반영하는 것이 특징이다.

4. 기출문제 풀이

소프트웨어 개발 객관식

- **소프트웨어 아키텍처 모델 중 MVC(Model-View-Controller)에 대한 설명으로 틀린 것은?**

  ① Model은 애플리케이션의 정보, 데이터를 나타낸다.

  ② View는 사용자에게 보여지는 화면을 나타낸다.

  ③ Controller는 데이터와 비즈니스 로직 간의 상호작용을 관리한다.

  ④ View와 Model은 서로 의존성을 가지며 직접 통신한다.

- 정답: **4**, MVC 패턴에서 View와 Model은 직접 통신하지 않는다. Controller가 Model과 View 사이의 중개자 역할을 한다. Model의 데이터가 변경되면 Controller를 통해 View에 알려지고, 사용자의 입력은 View에서 Controller를 통해 Model에 전달되고, 이를 통해 Model과 View의 독립성을 유지하고 결합도를 낮춘다.

4. 기출문제 풀이

소프트웨어 개발 객관식

- **테스트 케이스에 대한 설명으로 틀린 것은?**

  ① 테스트 케이스는 입력 값, 실행 조건, 기대 결과로 구성된다.

  ② 테스트 케이스는 요구사항 명세서를 기반으로 작성한다.

  ③ 테스트 케이스는 개발 단계에서만 사용된다.

  ④ 테스트 케이스는 소프트웨어의 품질을 향상시키는데 도움을 준다.

- 정답: **3**, 테스트 케이스는 개발 단계뿐만 아니라 소프트웨어 개발 생명주기의 여러 단계에서 사용된다. 요구사항 분석 단계에서 시작하여 설계, 구현, 테스트, 유지보수 단계에 이르기까지 전 과정에서 사용된다.

4. 기출문제 풀이

소프트웨어 개발 객관식

- **객체지향 개념을 활용한 소프트웨어 구현과 관련한 설명으로 옳은 것은?**

  ① 캡슐화는 객체의 속성과 메소드를 하나로 묶고 실제 구현 내용을 외부에 감추는 것이다.

  ② 추상화는 공통된 속성과 기능을 묶어 이름을 붙이는 것이다.

  ③ 다형성은 하나의 객체가 여러 가지 형태로 재구성되는 것을 의미한다.

  ④ 상속은 기존의 클래스를 재사용하여 새로운 클래스를 작성하는 것이다.

- 정답: **1**, 캡슐화는 객체지향 프로그래밍의 핵심 개념 중 하나로, 데이터(속성)와 그 데이터를 처리하는 메소드를 하나의 단위로 묶고, 실제 구현 내용을 외부로부터 숨기는 것을 의미한다.

famous77@kaist.ac.kr