11주차

데이터의 저장과 검색

- 이진검색트리
- 최대값 및 최소값 검색

		강의 주제	주차
1부 컴퓨팅 사고력	컴퓨팅 사고력의 소개	IT 사회, 소프트웨어 세상, 컴퓨팅 사고력의 소개, 컴퓨팅 사고력의 개념	1
		컴퓨팅 사고력의 개념, 주위에서 볼 수 있는 컴퓨팅 사고력, 문제해결 방법	2
	문제해결 방법, 컴퓨터	문제해결 방법, 문제해결 과정 예, 문제해결을 위한 소프트웨어 설계 사상, 컴퓨터의 특징, 소프트웨어, 유한상태기계	3
2부 소프트웨어	알고리즘	알고리즘 소개, 알고리즘의 표현 방법, 의사코드, 흐름도	4
	프로그램	프로그램의 기능, 함수, 컴파일러	5
	파이썬	파이썬 소개 및 설치, 변수에 값 저장, 입력, 출력, 조건부 수행	6
		반복, 리스트, 함수, 출력 형식	7
3부 컴퓨팅 사고력 활용하기	데이터의 표현	이진수, 아스키코드, 오디오 데이터, 이미지 데이터, 자료구조	8
		인코딩 및 압축, 오류확인	9
	데이터의 저장과 검색	배열 및 연결 리스트, 선형검색, 이분검색, 색인순차검색, 해싱	10
		이진검색트리, 최대값 및 최소값 검색	11
	알고리즘설계	정렬, 분할정복 알고리즘, 탐욕적 알고리즘	12

이진검색트리

Binary Search Tree

[problem]

트리를 이용하여 데이터를 쉽게 저장하고, 검색할 수 있는 방법을 고안하시오.

● 문제 해결 과정

1.문제를 이해한다
2.계획을 세운다
3.계획을 실행한다
4.풀이과정을 재점검한다

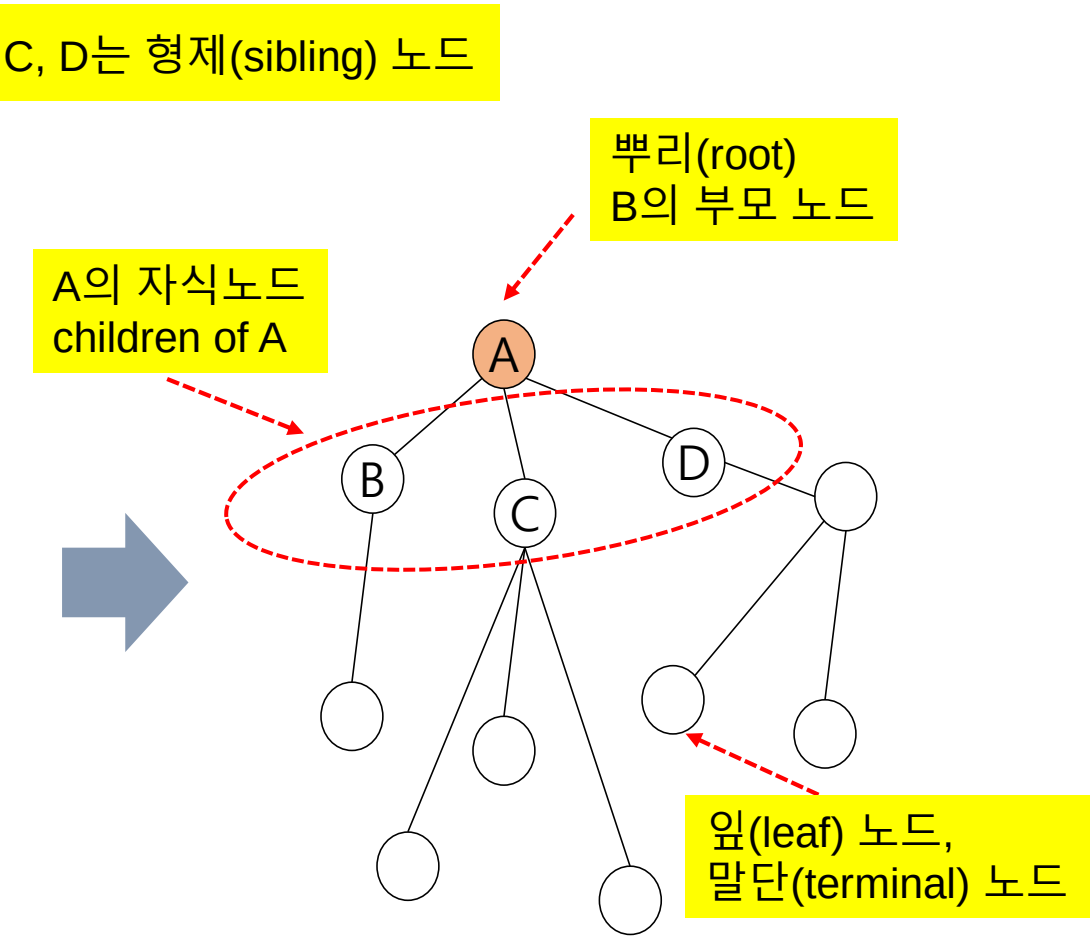
● Computational Thinking

- 분해
- 패턴 확인
- 추상화
- 알고리즘 설계
- 데이터 모음
- 데이터 표현
- 데이터 분석
- 모의실험
- 자동화
- 병렬화
- 패턴 일반화

● 문제 해결 전략

- 추정과 확인
- 순서 리스트 작성
- 가능성들을 제거
- 대칭 성질을 이용
- 특별한 경우를 고려
- 직접 추론 사용
- 방정식을 푼다
- 패턴을 찾는다
- 그림을 그린다
- 단순한 문제 해결 시도
- 모델 사용
- 역방향으로 시도
- 공식 사용
- 창조적 생각
- 잔꾀를 사용

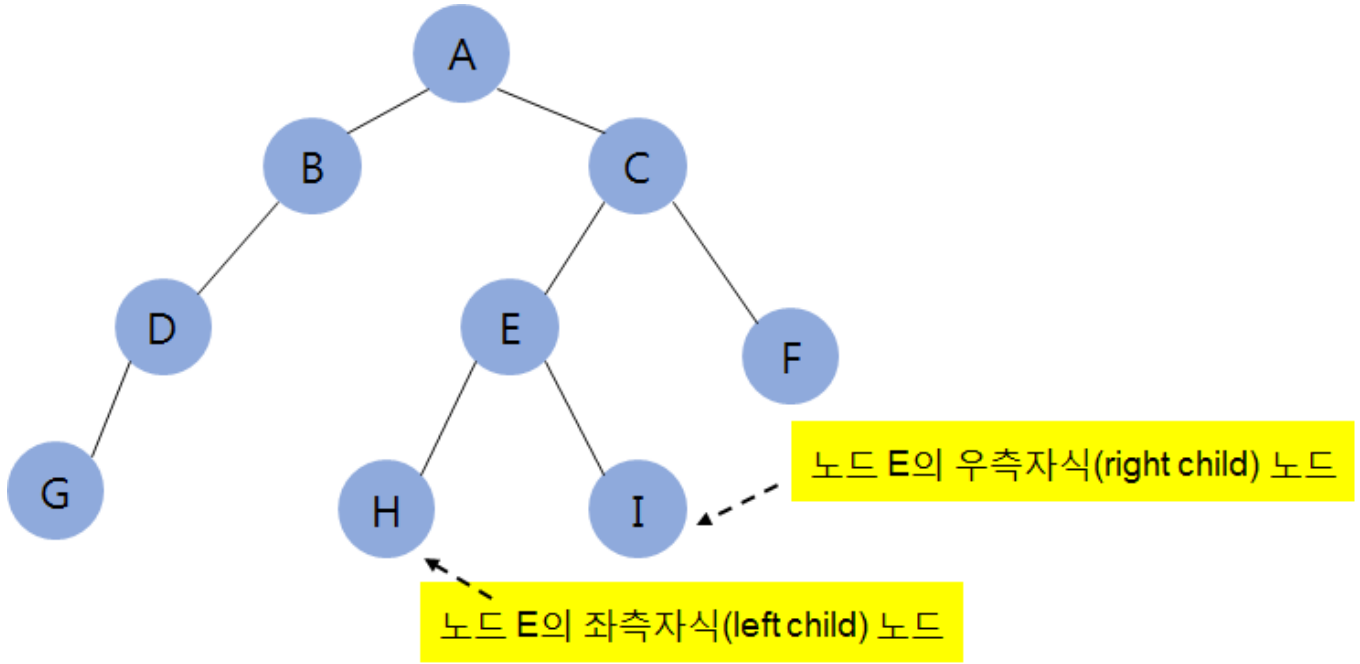
트리(Tree)



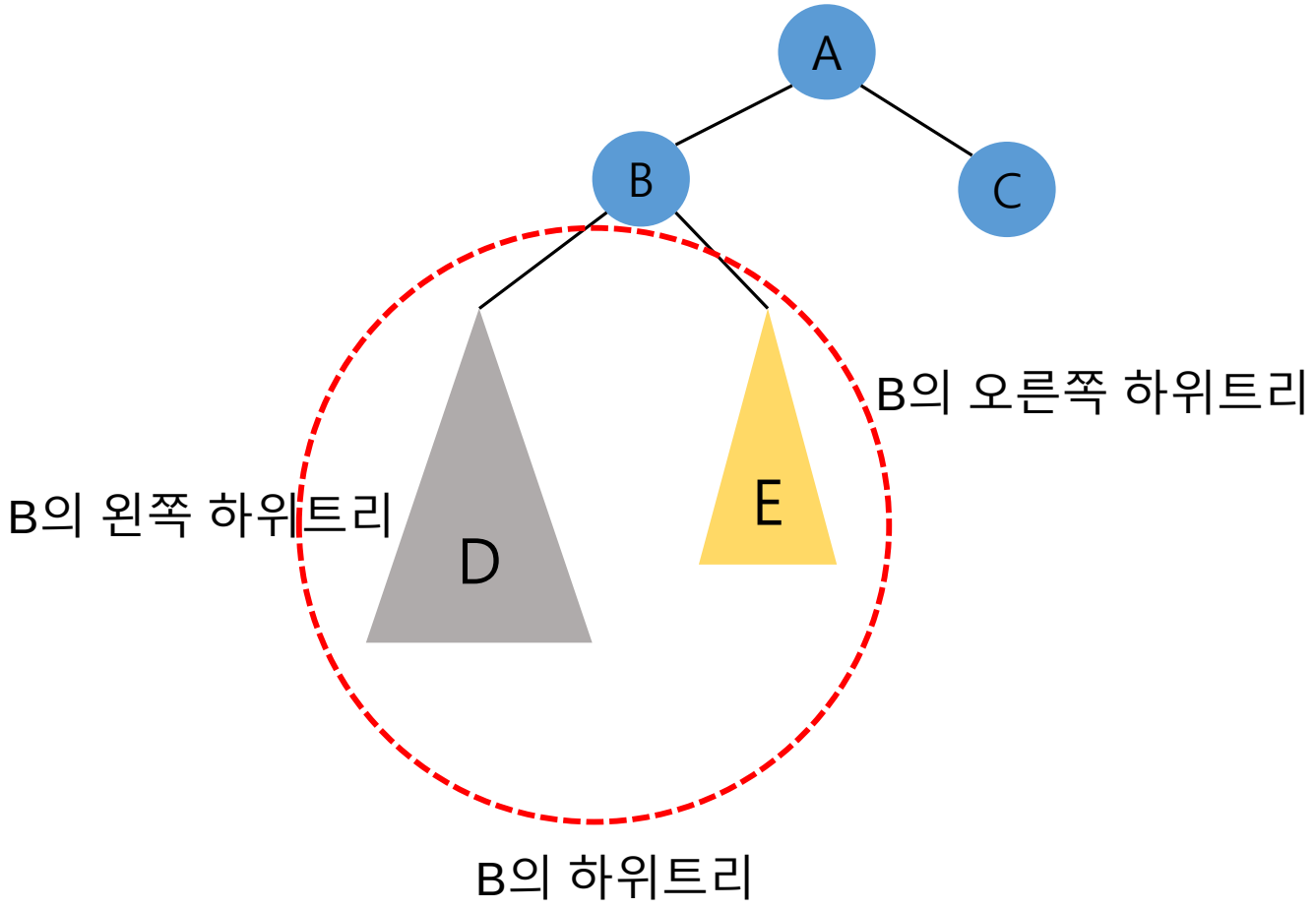
데이터의 계층적인 구조를 나타낼 수 있는 자료 구조

이진트리 (Binary tree)

- 트리의 단순한 형태
- 각 노드는 최대 2개의 자식노드(children node)를 가질 수 있다.
- DB의 자료 저장 자료구조로 활용

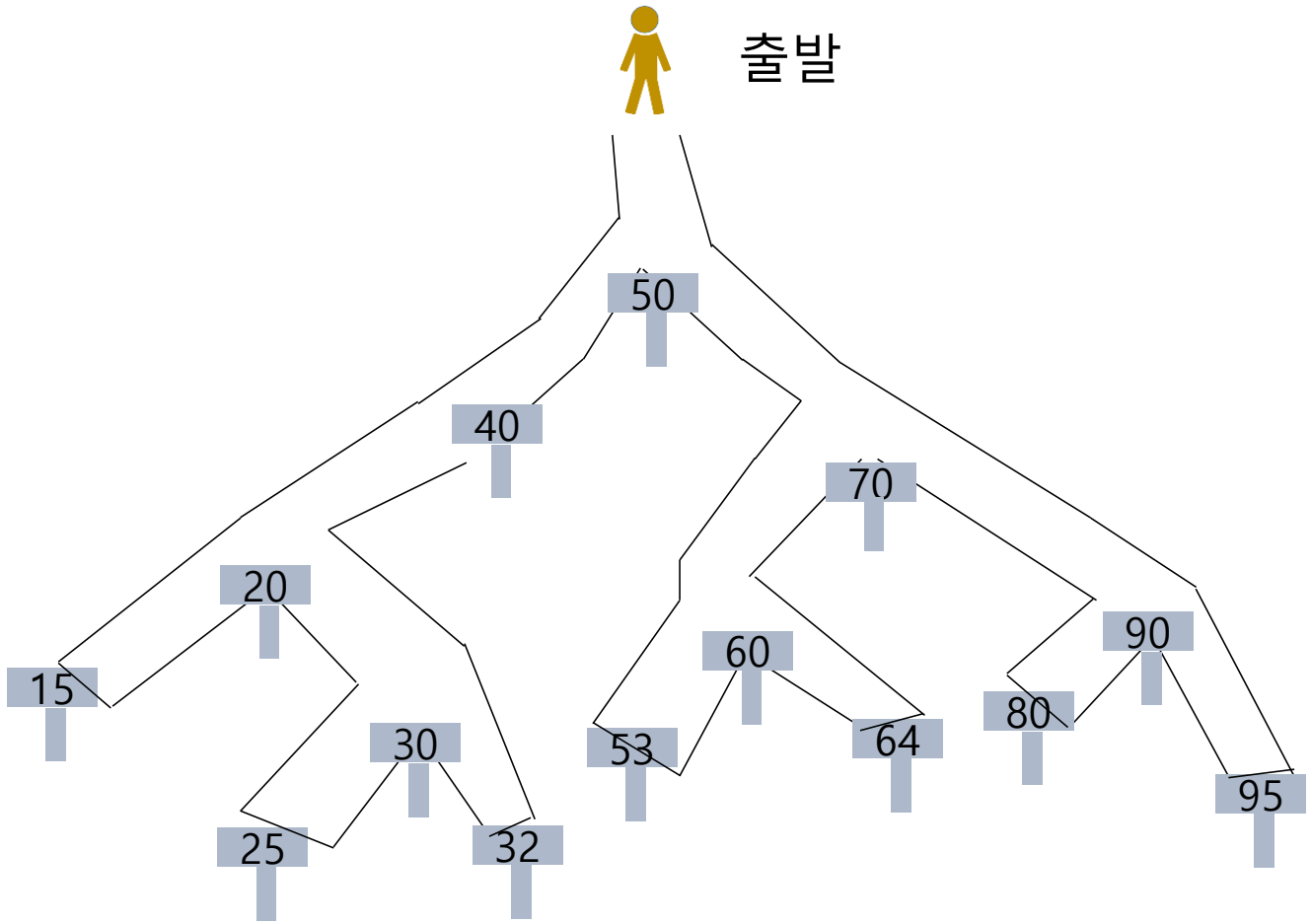


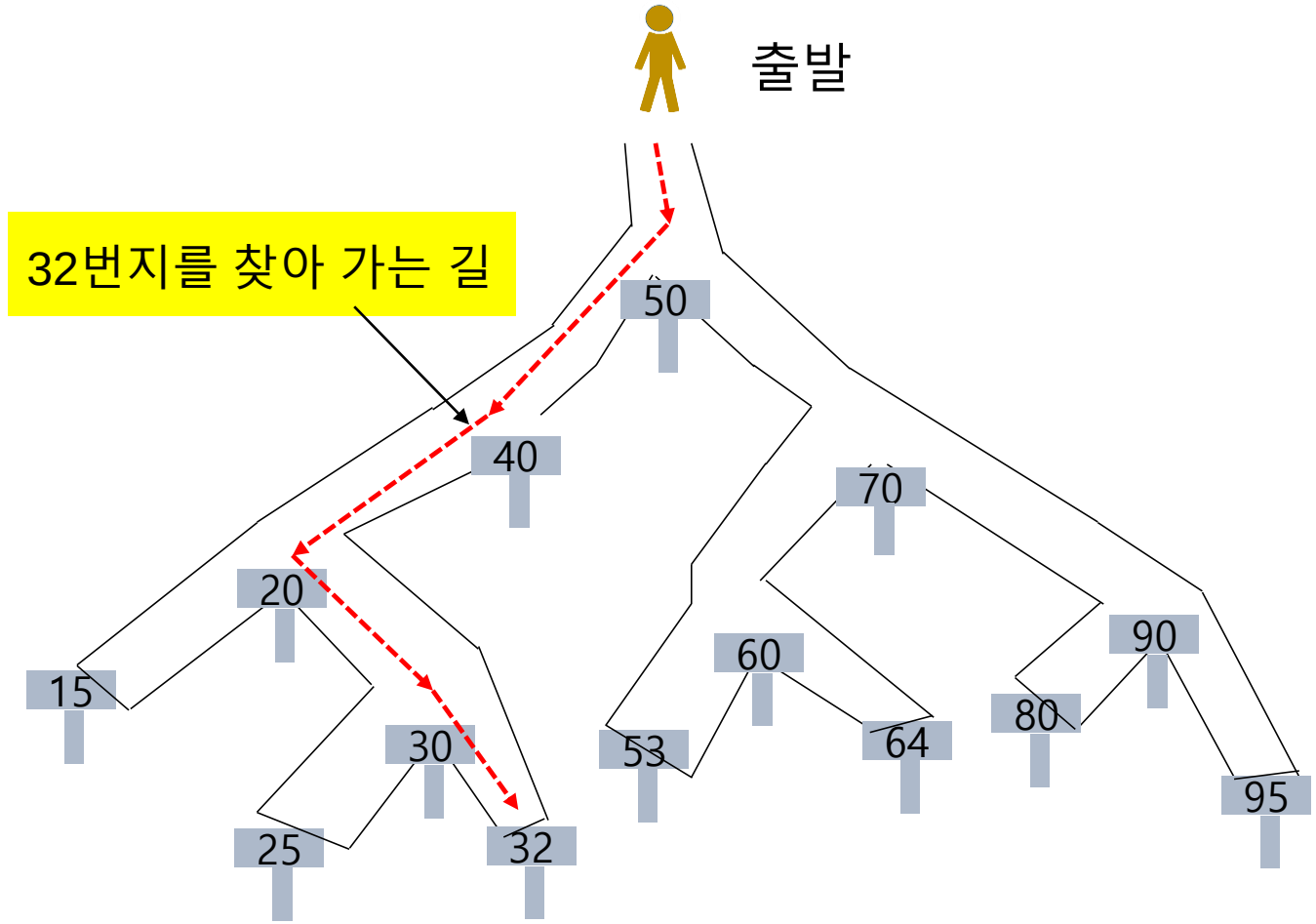
하위트리(subtree)



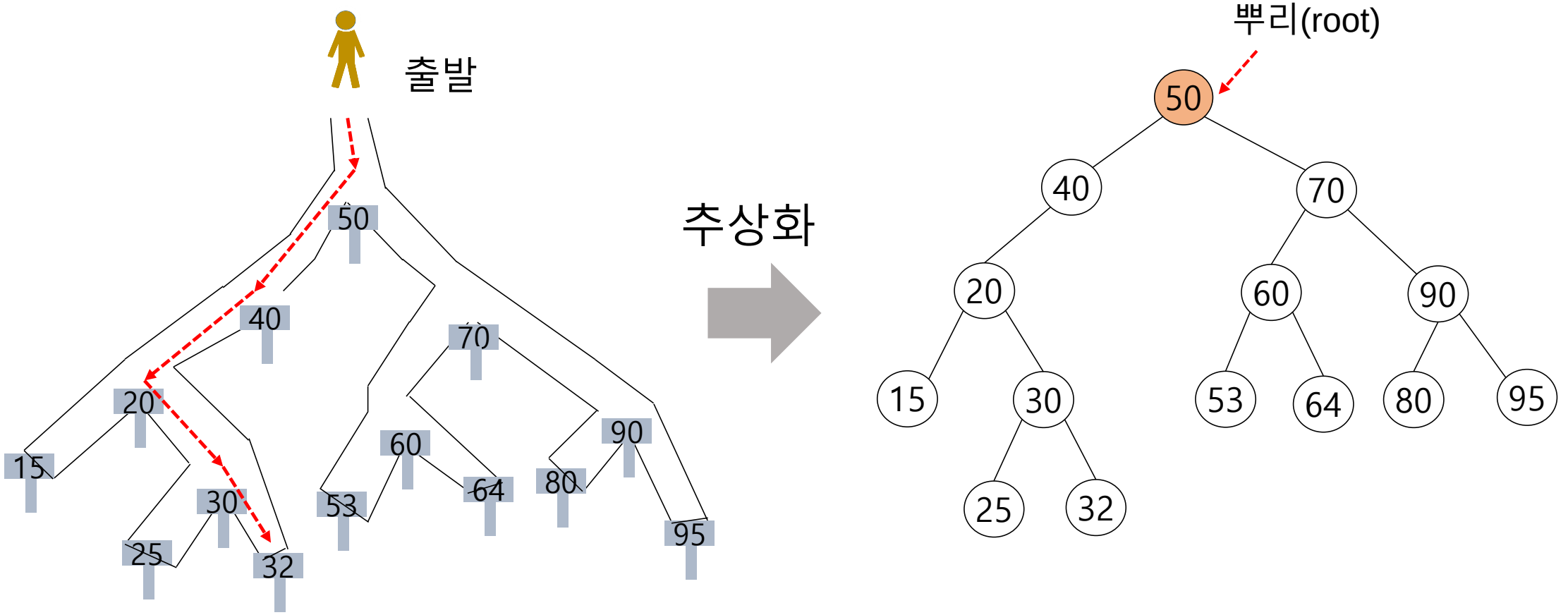
[problem] 안내판을 보고 특정 번지수를 찾아 간다

- 갈림길과 막다른 길에는 안내판이 있다.
- 그림에 있는 안내판들의 위치와 숫자의 관계를 알 수 있나?



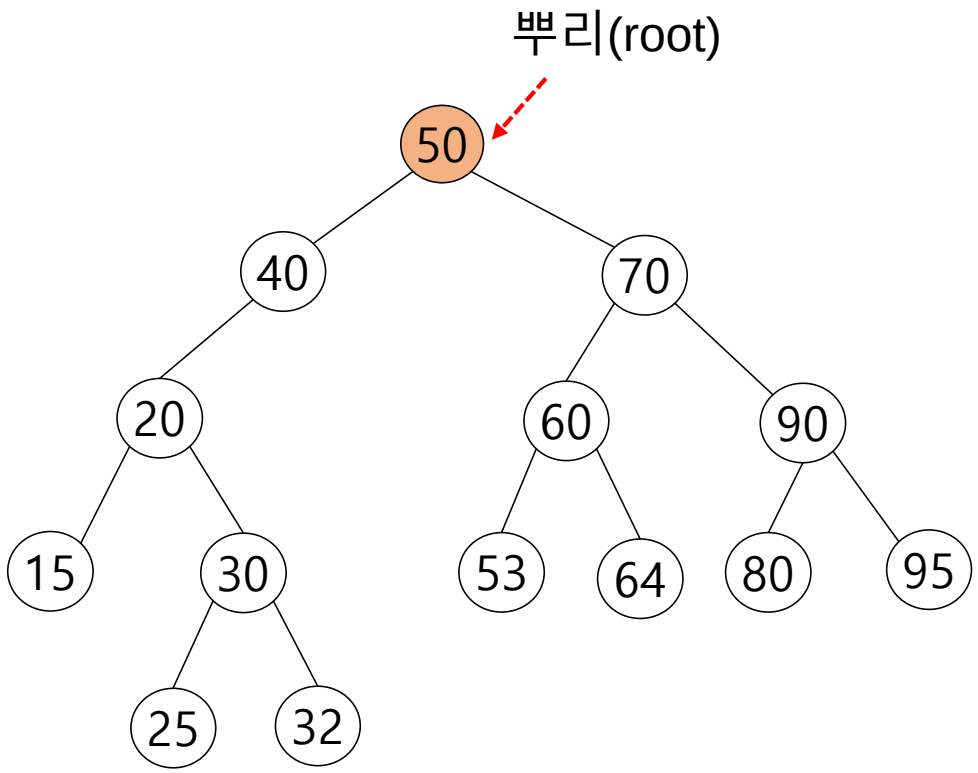


- 우리가 보는 방향에서, 안내판의 왼쪽으로 가게 되면 안내판보다 작은 번지수 들이 위치
- 안내판의 오른쪽으로 가게 되면 안내판보다 큰 번지수들이 위치
- 갈 갈림길마다 동일한 성질이 적용

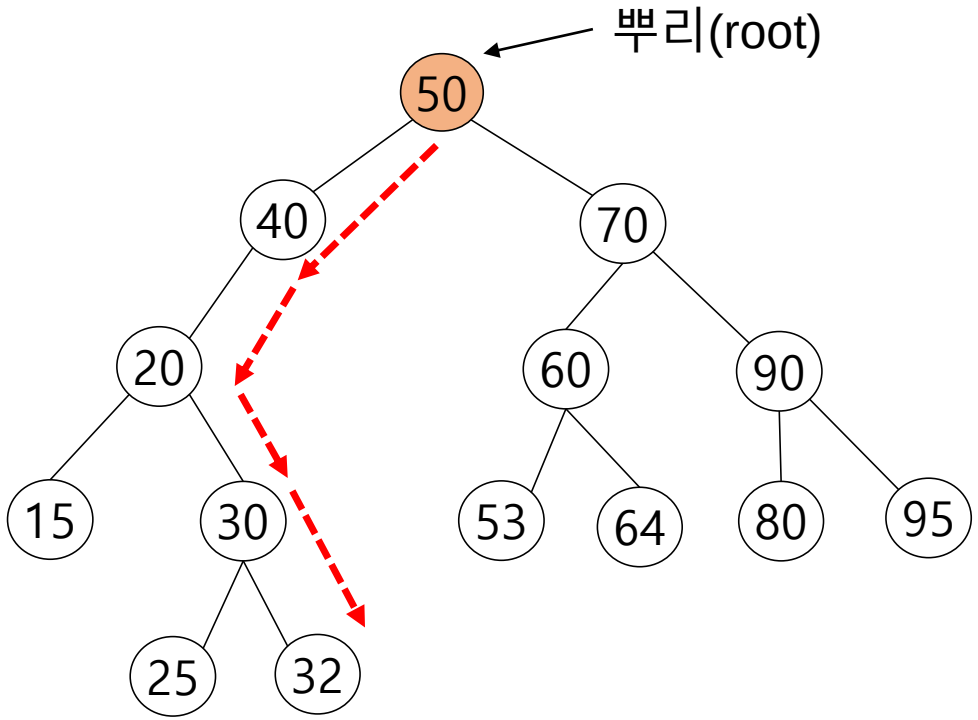

출발
50
40
70
20
60
90
15
30
53
64
80
95
25
32
추상화
뿌리(root)
50
40
70
20
60
90
15
30
53
64
80
95
25
32

이진검색트리 (Binary Search Tree)

- 트리 구조를 이용해서 데이터의 성질을 고려한 저장 구조
- 노드 A의 왼쪽 하위트리의 노드들에는 A보다 작은 데이터가 저장되고, 노드 A의 오른쪽 하위트리의 노드들에는 A보다 큰 데이터가 저장된다.
- 이 성질을 각 노드 마다 재귀적으로(recursive) 적용한 트리


뿌리(root)
50
40
70
20
60
90
15
30
53
64
80
95
25
32

- 데이터 32를 찾는 과정을 확인하시오.
  - ✓ 검색은 항상 뿌리(root)에서 출발한다.



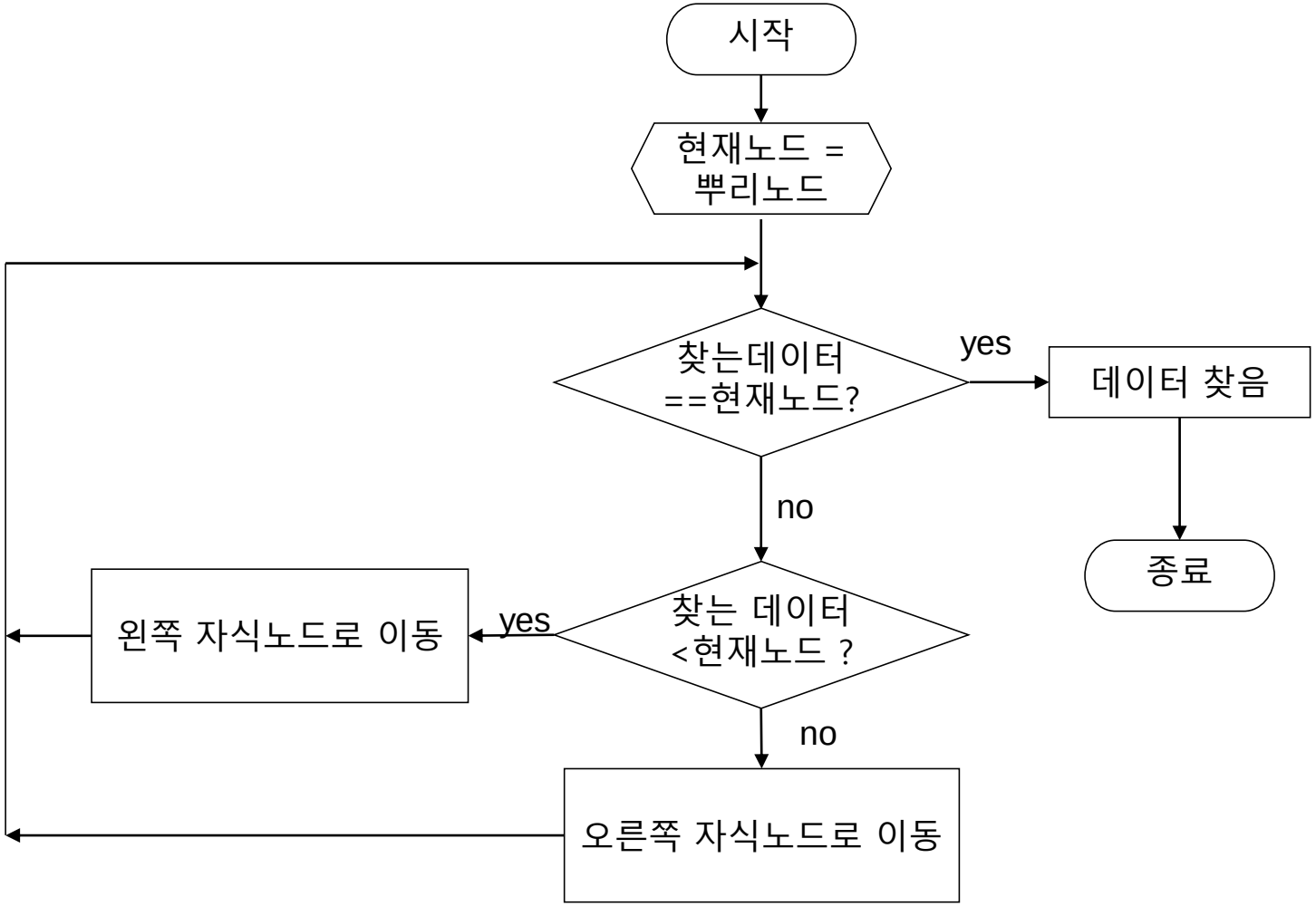
이진검색트리에서 데이터를 찾는 알고리즘

- 데이터가 트리 내부에 있을 경우

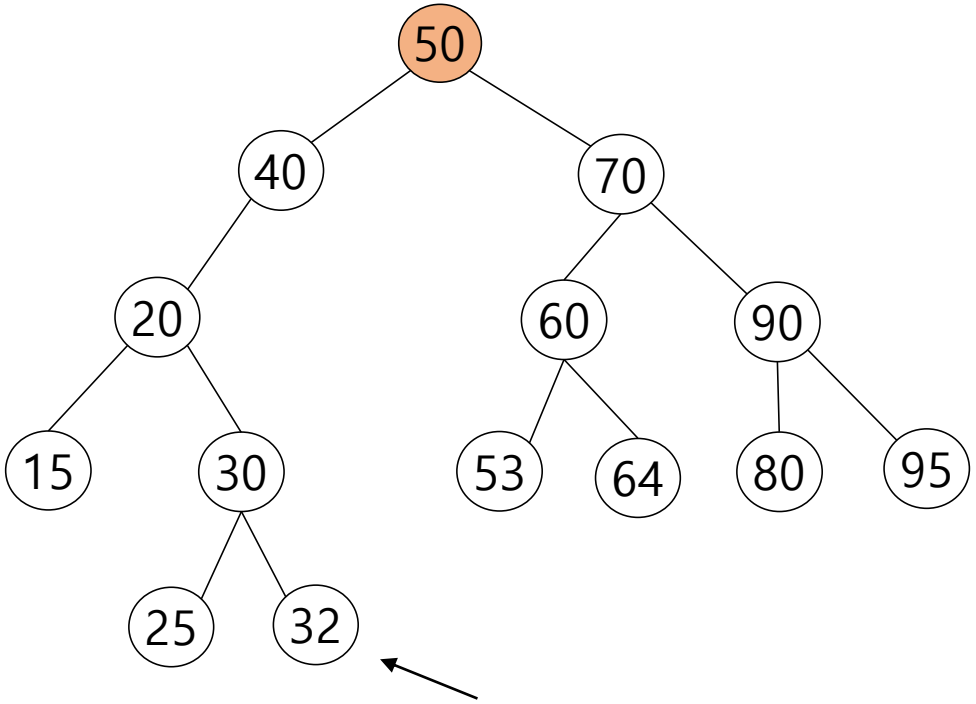
```
현재노드=뿌리노드
while (아직 찾지 못했음)
    if (찾는데이터 == 현재노드)
        데이터를 찾았다.
    else if (찾는데이터 < 현재노드)
        현재노드의 왼쪽 자식노드로 이동
    else
        현재노드의 오른쪽 자식노드로 이동
```

이진검색트리에서 데이터를 찾는 알고리즘

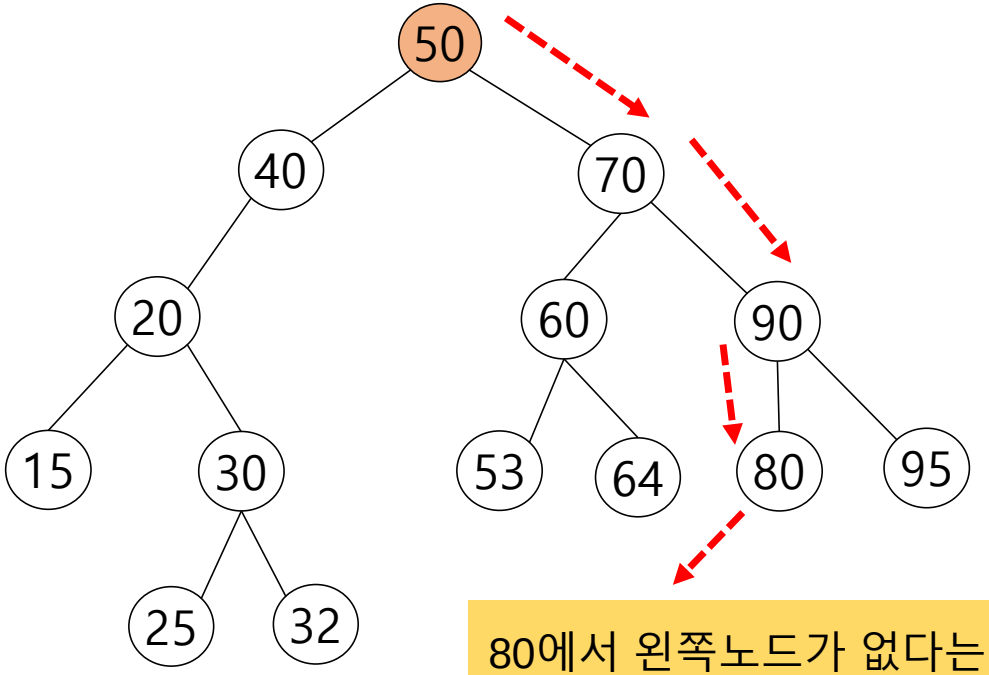
- 데이터가 트리 내부에 있을 경우



- 뿌리에 있는 데이터를 찾을 경우 1회의 비교만 하면 되고 (예, 50 찾기)
- 뿌리에서 제일 먼 곳에 있는 데이터를 찾을 경우 제일 많은 비교횟수가 필요하다. (예, 32 찾기)



- 이진검색트리에 없는 데이터를 찾을 때
- [예] 79를 찾는 경우



80에서 왼쪽노드가 없다는 것을 확인해야 79가 없다는 것을 확인할 수 있다.

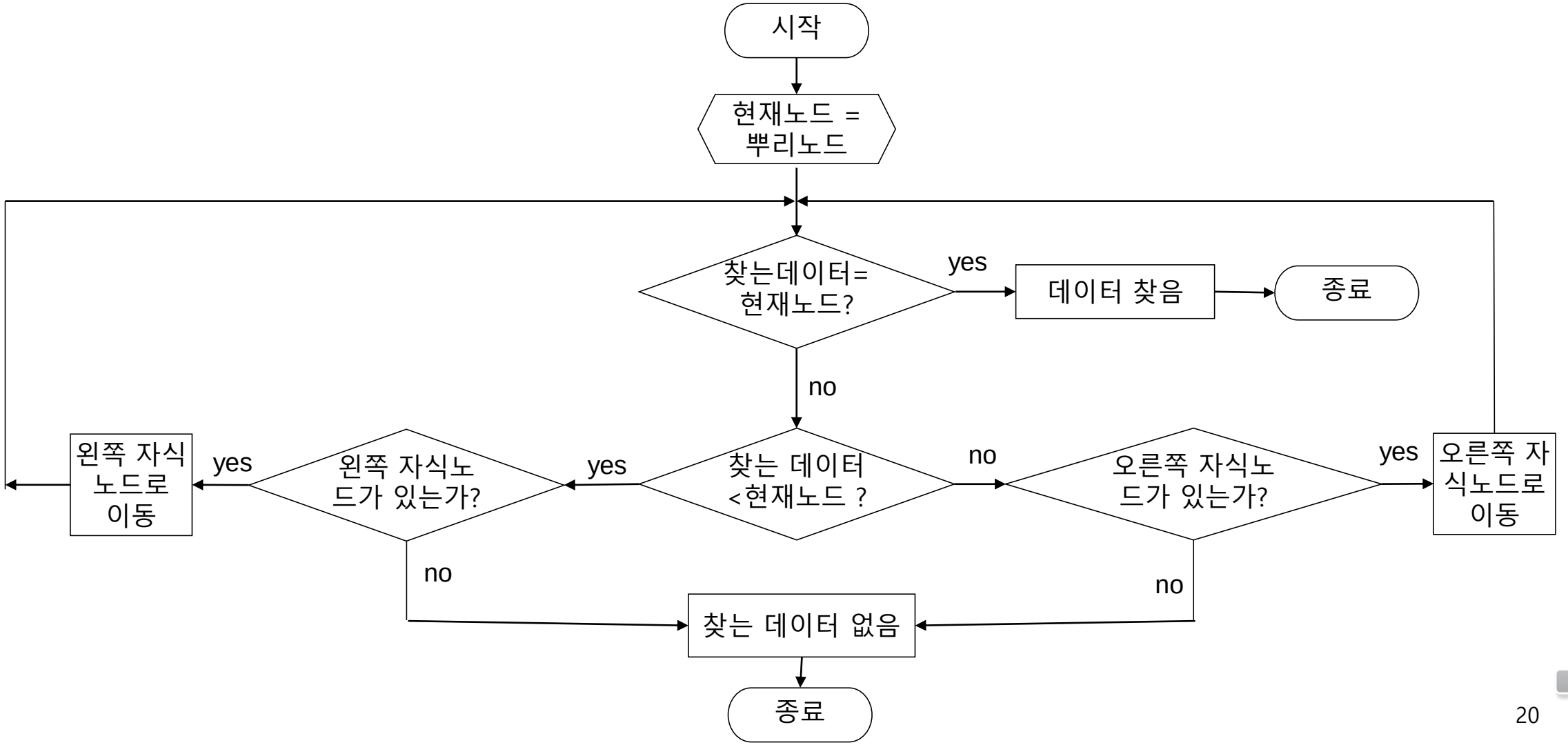
이진검색트리에서 데이터를 찾는 알고리즘

- 데이터가 트리 내부에 없을 수도 있을 경우

```
현재노드=뿌리노드
단계 1: if(찾는데이터 == 현재노드)
            현재노드와 같으면 데이터 찾았음. 종료
단계 2: else if (찾는데이터 < 현재노드)
            if (왼쪽자식노드가 있으면)
                  왼쪽자식노드로 이동
            else
                  종료
단계 3: else
            if (오른쪽자식노드가 있으면)
                  오른쪽자식노드로 이동
            else
                  종료
단계 4: 단계 1로 이동
```

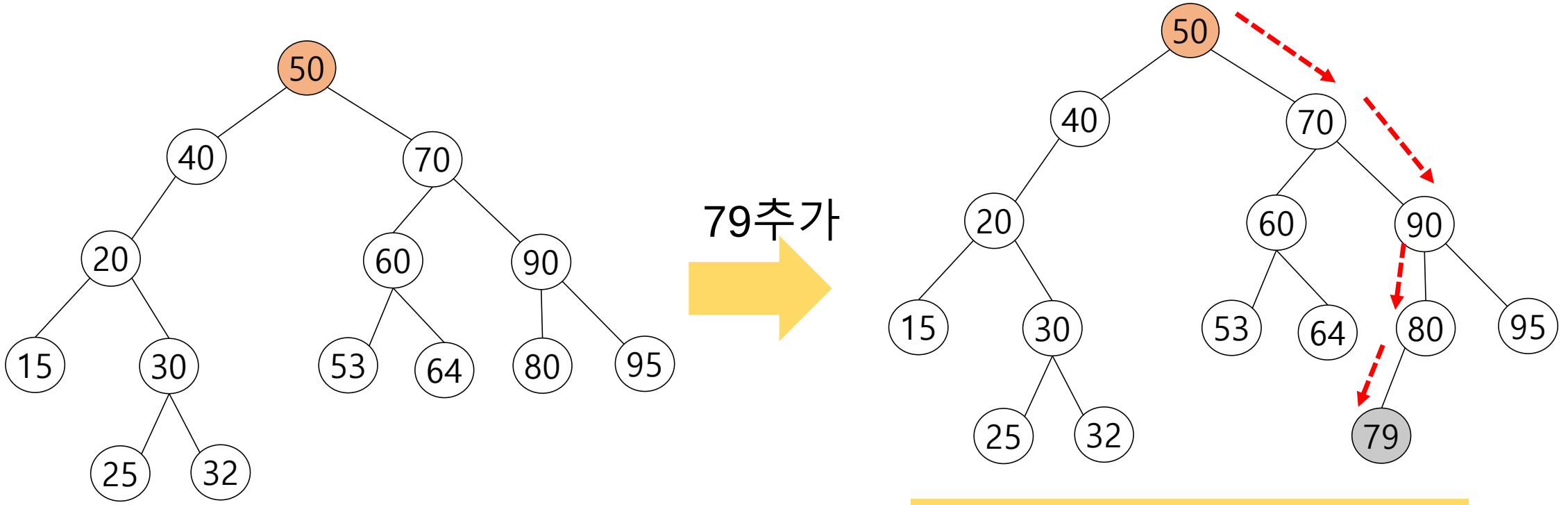

이진검색트리에서 데이터를 찾는 알고리즘

- 데이터가 트리 내부에 없을 수도 있을 경우



이진검색트리에 데이터를 추가할 때

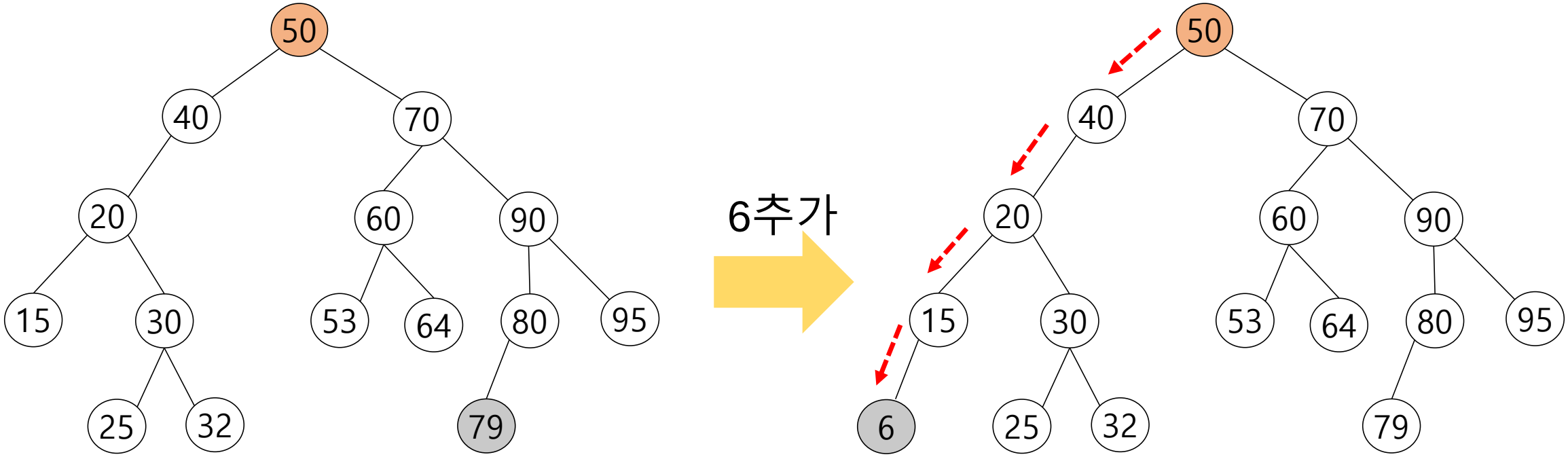
- 이미 있는 데이터인 경우는 추가안함.
- [예] 79를 추가



80에서 왼쪽노드가 없다는 것을 확인한 후 79를 80의 왼쪽자식노드로 추가한다.

이진검색트리에 데이터를 추가할 때

- 이미 있는 데이터인 경우는 추가 안함.
- [예] 6을 추가



새로운 데이터의 추가 방법

- 이미 있는 데이터는 추가 안할 경우

m = 추가할 데이터

현재노드=뿌리노드

단계 1: if(현재노드==m)

현재노드와 같으면 이미 데이터 있음. 종료

단계 2: else if (m < 현재노드)

if (왼쪽자식노드가 있으면)

왼쪽자식노드로 이동

else

m을 현재노드의 왼쪽 자식으로 연결 . 종료

단계 3: else

if (오른쪽자식노드가 있으면)

오른쪽자식노드로 이동

else

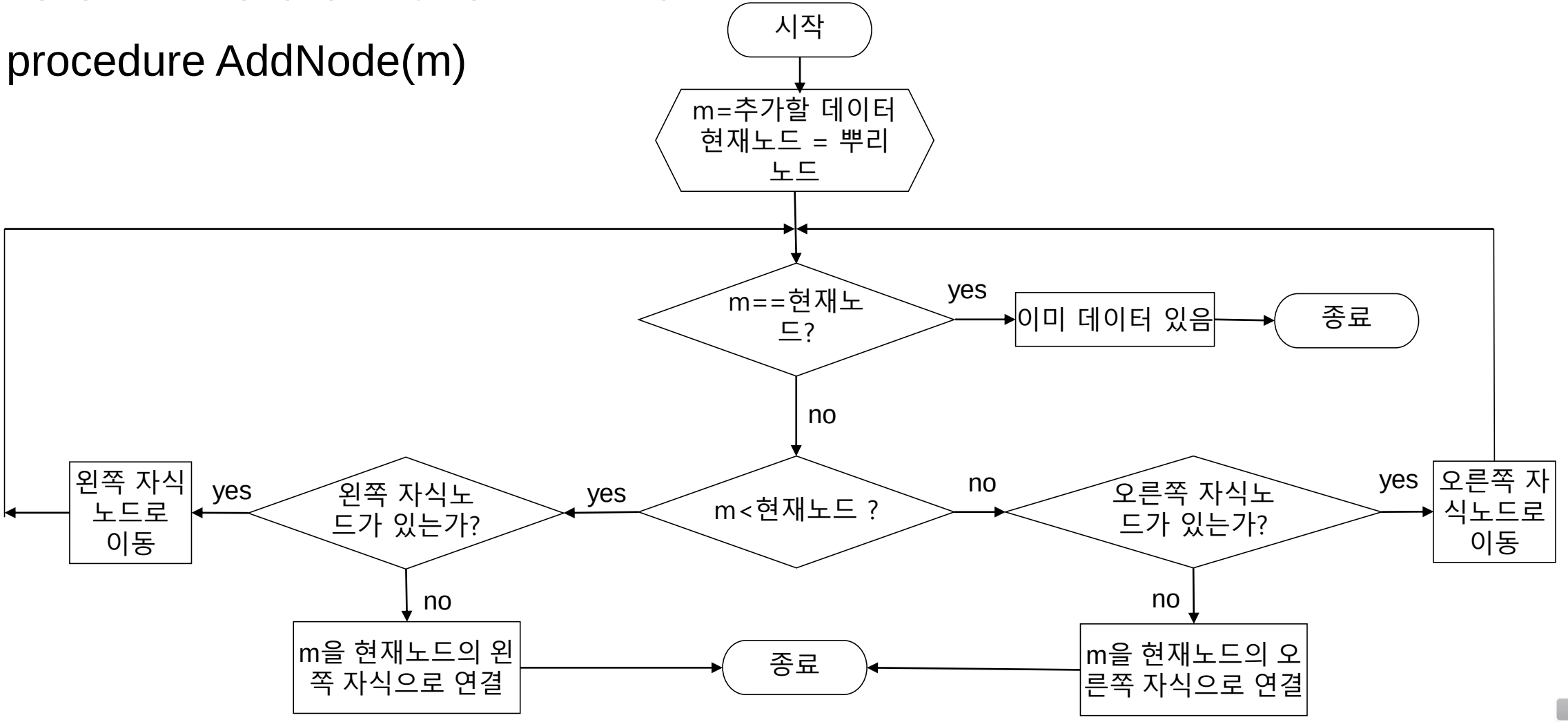
m을 현재노드의 오른쪽 자식으로 연결. 종료

단계 4: 단계 1로 이동

새로운 데이터의 추가 방법

- 이미 있는 데이터는 추가 안할 경우

- procedure AddNode(m)



데이터를 이진검색트리로 만드는 방법

- 데이터: $a_1, a_2, \ldots, a_n$
- 절차 AddNode를 각각의 데이터 $a_1, a_2, \ldots, a_n$ 에 대해 적용한다.
- 이진검색트리에 하나씩 데이터를 추가하는 절차의 반복

AddNode(a_1)

AddNode(a_2)

⋮

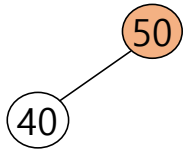
AddNode(a_n)

이진검색트리를 만들어 가는 과정

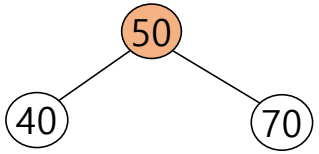
- 데이터가 한 개씩 50 40 70 20 30 25 15 90 85 60 64 53 의 순서로 발생한다고 가정
- 데이터의 저장은 트리의 뿌리에서 시작하여 합당한 자리를 찾은 후 저장한다.

최초 50

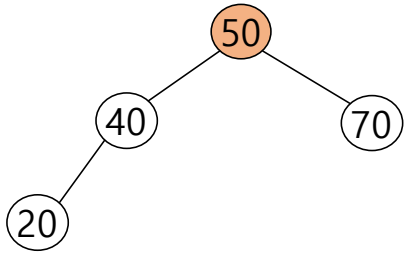
40



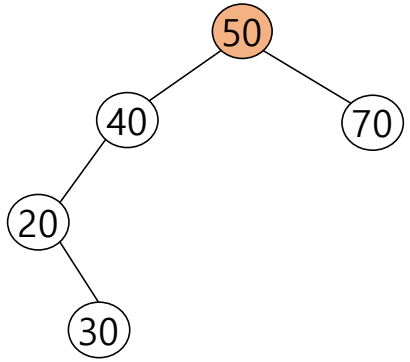
70



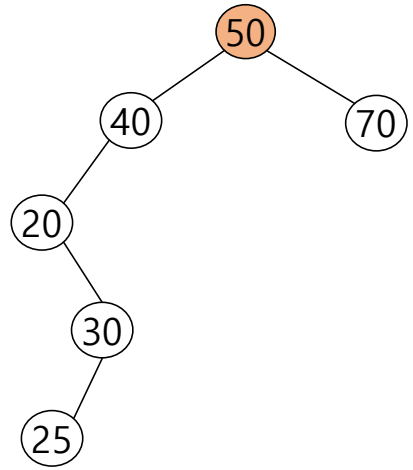
20



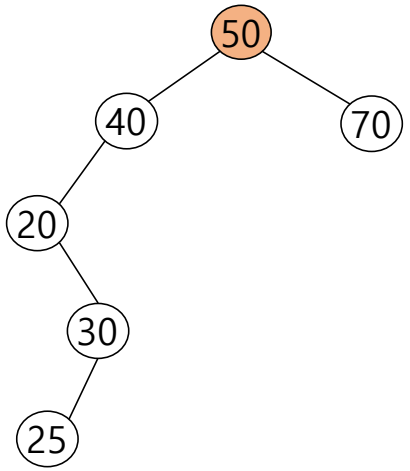
30



25



- 이후 15 90 85 60 64 53 데이터가 더 발생하여, 이진검색트리에 저장하는 과정을 그려 보시오.

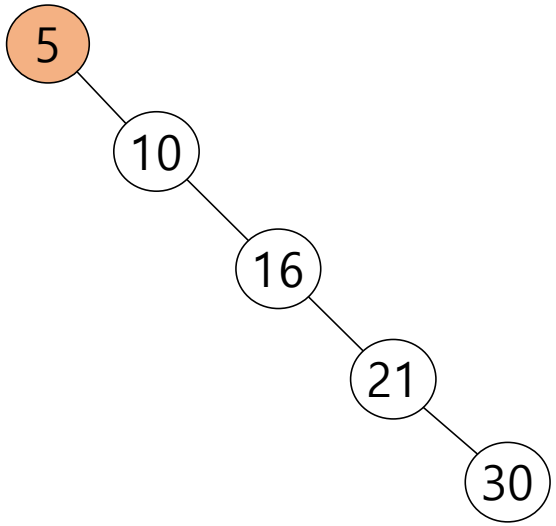


[연습문제]

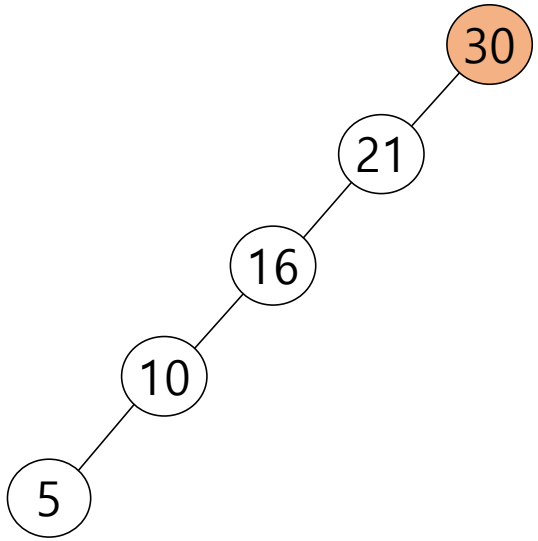
데이터 5, 1, 3, 9, 7, 2, 4, 6, 8을 이용하여 이진검색트리를 완성하시오.

- 다음의 데이터를 저장하는 이진검색트리를 작성하시오.

✓ 데이터가 5 10 16 21 30 의 순서로 발생할 때,

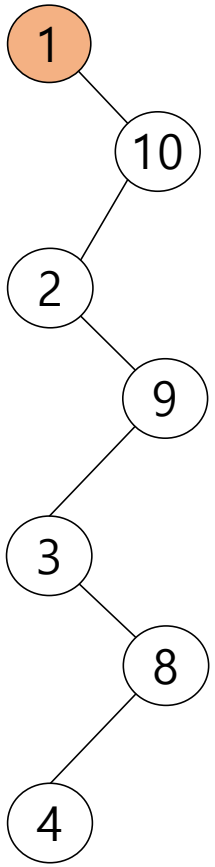


✓ 데이터가 30 21 16 10 5 의 순서로 발생할 때,



[연습문제] 데이터가 1 10 2 9 3 8 4 의 순서로 입력될 경우의 이진검색트리를 작성하시오.

[답] 데이터가 1 10 2 9 3 8 4 의 순서로 입력될 경우의 이진검색트리

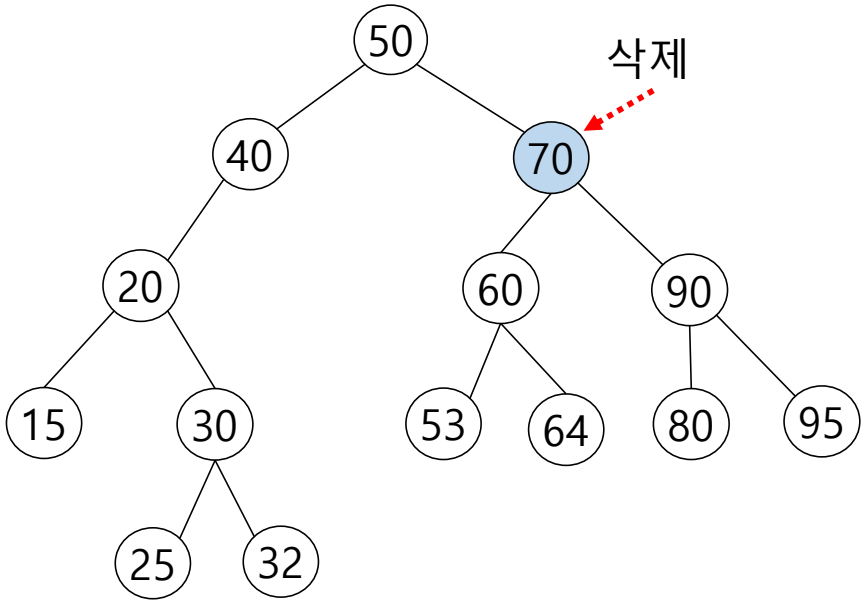


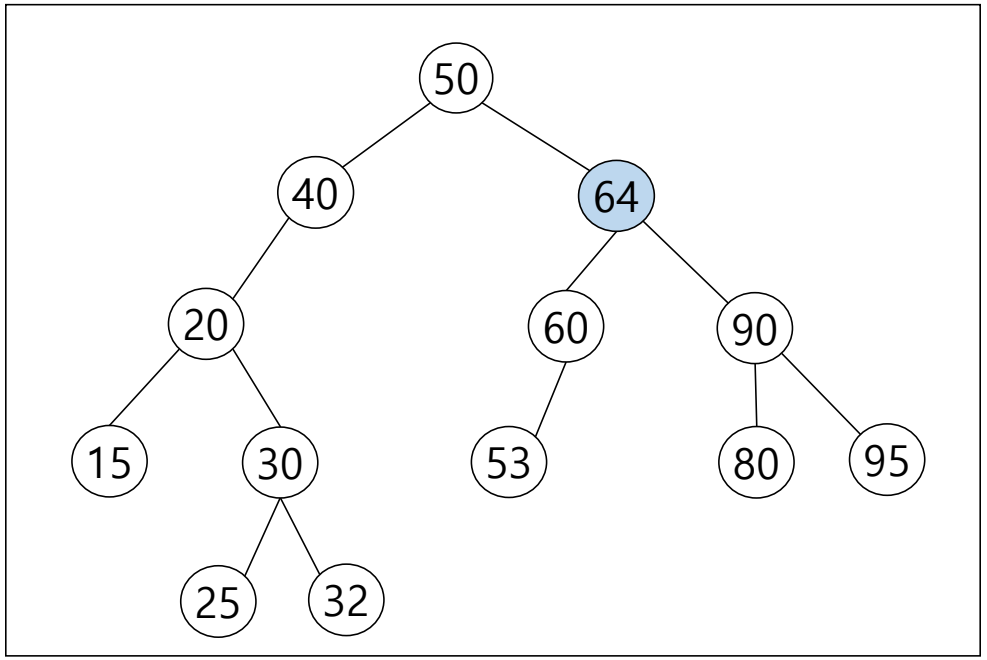
오름차순, 내림차순이 아닌 경우도 한 줄로 나열된 이진검색트리가 생성될 수 있다.

이진검색트리에서 하나의 데이터를 삭제하는 방법

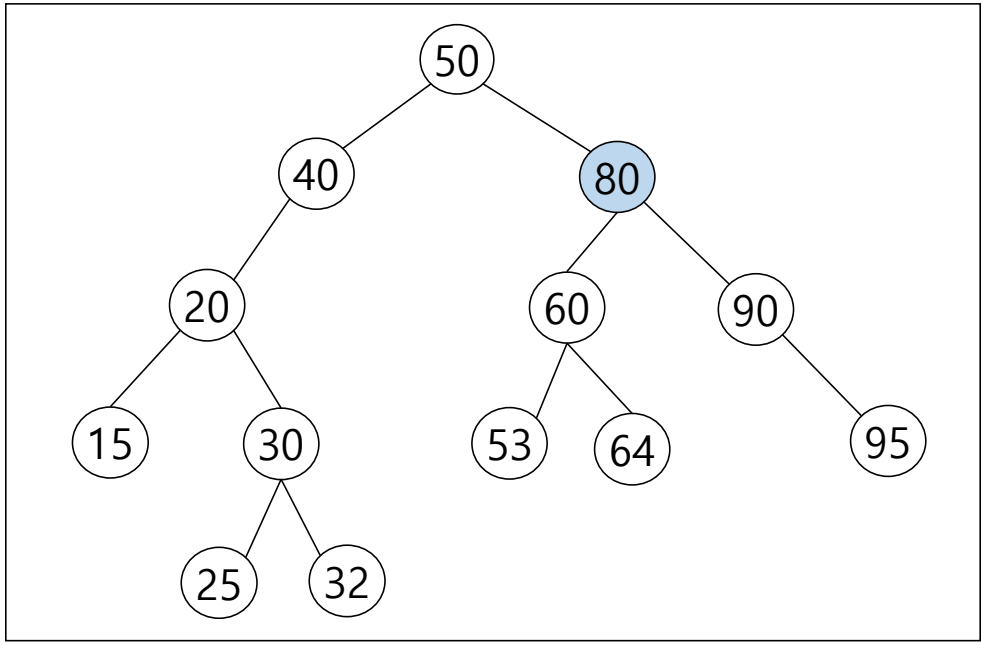
- 두 개의 자식노드가 있는 노드가 삭제될 경우
- 한 개의 자식노드가 있는 노드가 삭제될 경우

- 두 개의 자식노드가 있는 노드가 삭제될 경우

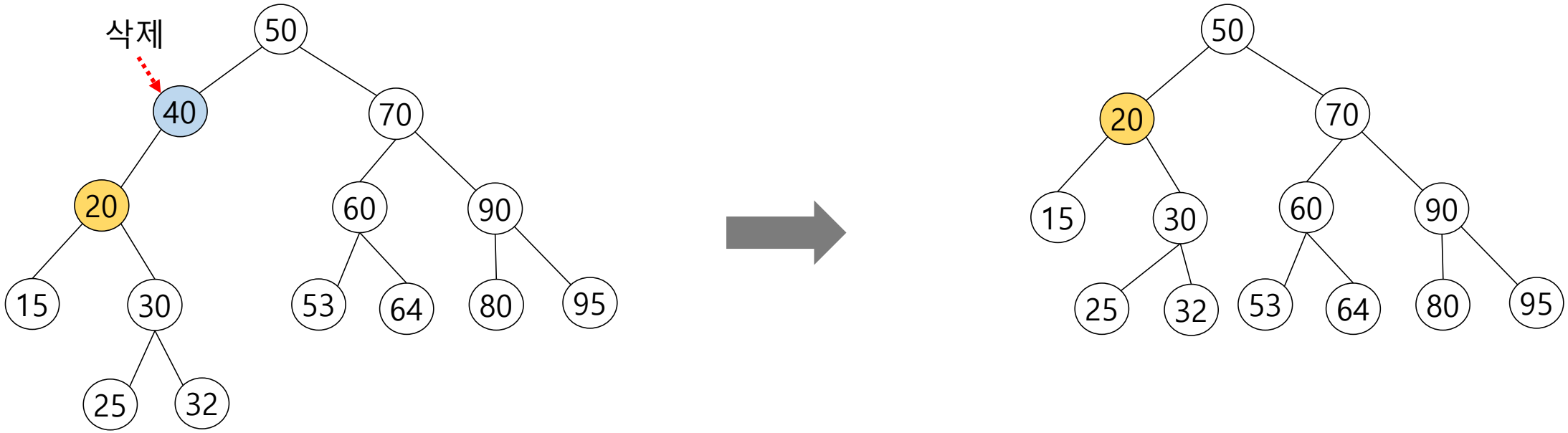




or

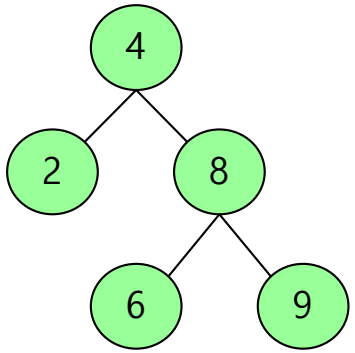


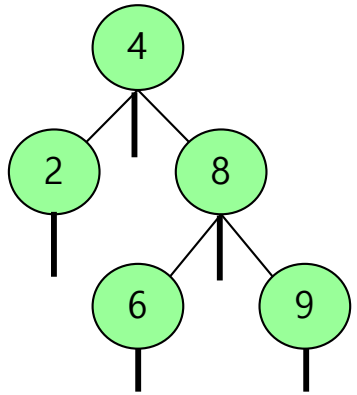
- 한 개의 자식노드가 있는 노드가 삭제될 경우



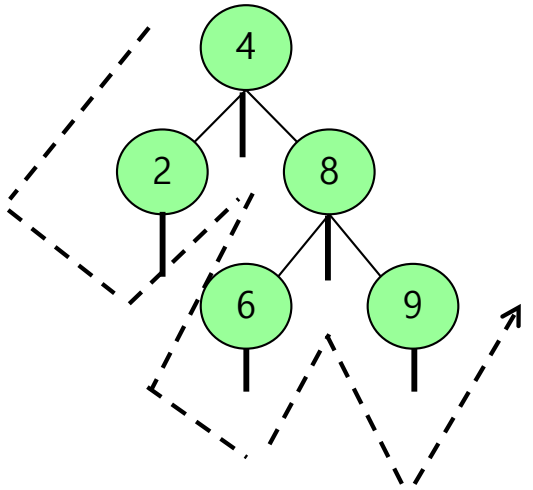
- 이진검색트리를 inorder로 방문하면 정렬된 데이터의 순서를 얻을 수 있다.

- inorder





각 노드에 아래 방향으로 막대기를 연결



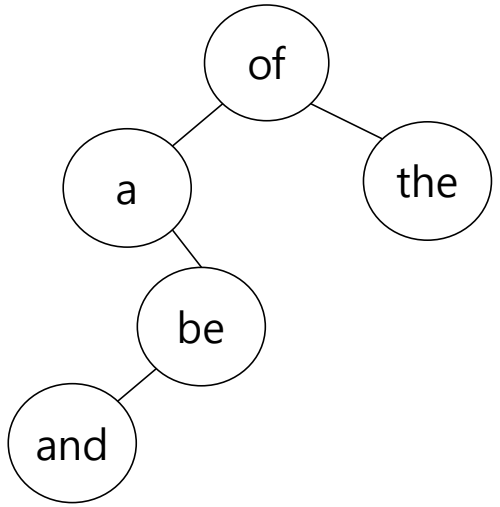
- 화살표 표시 방식으로 이동
- 이동하면서 만나는 막대기가 연결된 노드 번호를 읽는다.
- 2 4 6 8 9 의 순서를 얻는다 – 정렬된 순서

최적 이진검색트리

[문제]

- 다음의 단어를 이진검색트리에 저장한다. - the, be, and, of , a
- 어떻게 저장할 때 단어의 평균검색시간을 최소로 할 것인가?

[평균검색시간]



(1) of의 평균검색시간: 1X of를 검색할 확률

(2) a의 평균검색시간: 2X a를 검색할 확률

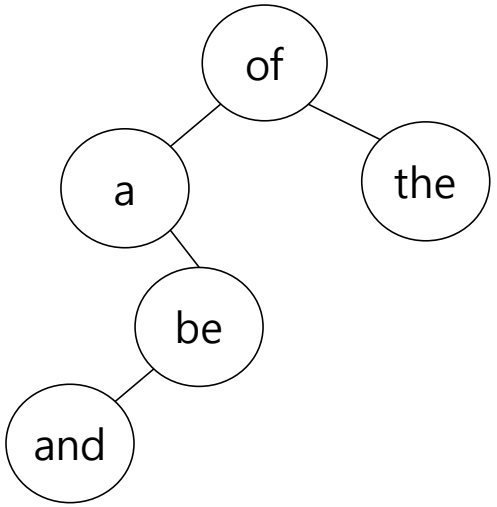
(3) the의 평균검색시간: 2X the를 검색할 확률

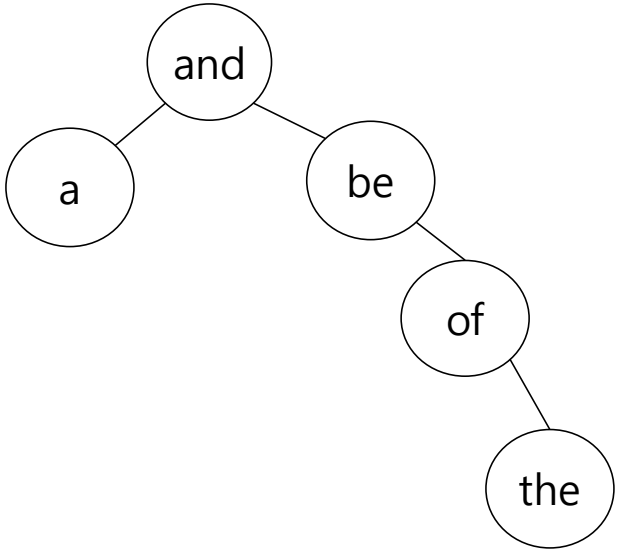
(4) be의 평균검색시간: 3X be를 검색할 확률

(5) and의 평균검색시간: 4X and를 검색할 확률

이진검색트리의 평균검색시간 = (1)+(2)+(3)+(4)+(5)

각 단어를 검색할 확률이 동일하다면,

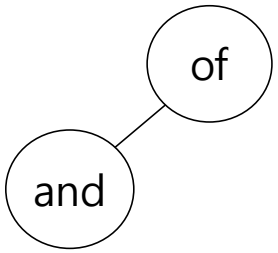




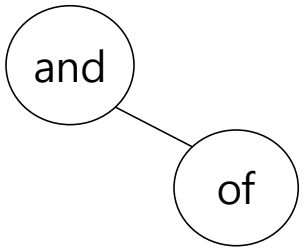
두 종류의 이진검색트리는 동일한 평균검색시간 소요

단어를 찾을 확률 - and: 0.7 of: 0.3

두 단어를 저장하는 다음의 이진검색트리 중 어느 구조가 효과적일까?



(1)



(2)

(1) 평균검색시간 = 2 X0.7 + 1 X0.3 = 1.7

(2) 평균검색시간 = 1 X0.7 + 2 X0.3 = 1.3

(2)의 구조가 효과적이다.

- 단어의 검색확률이 서로 다르다면 이진검색트리의 종류마다 평균검색시간이 달라진다.
- 실제로 단어의 검색확률이 서로 다르다.
- 최소의 평균검색시간을 제공하는 트리를 최적의 이진검색트리라고 한다.
- 해결 아이디어: 자주 검색하는 단어는 뿌리노드에 가깝게 배치, 어쩌다 검색하는 단어는 뿌리노드에서 멀리 배치
- 최적이진검색트리 찾는 문제는 매우 풀기 어려운 문제이다.

Computational thinking

- 데이터는 선형적이지 않은 어떤 구조에 저장되었는가?
- 선형적인 구조보다 데이터를 빨리 찾을 수 있는 요인은 무엇인가?
- 일상 생활에서 이진검색트리의 전략이 포함되어 있는 사례를 들 수 있나?
- 특정 데이터가 더 이상 필요 없을 때는 어떤 작업을 하여야 하나?
- 데이터를 이진검색트리에 저장하였더니, 자료가 정렬되었다. 무슨 이유에서 그렇게 될 수 있을까?

컴퓨터 과학

- 이진검색트리는 데이터베이스에서 자료를 저장하는 기본 구조로 사용된다.
- 데이터의 생성 순서에 따라 이진검색트리가 달라진다.
- 어떤 경우에는 이진검색트리가 단순히 연결리스트의 구조를 갖게 된다.
- n개의 데이터가 저장된 이진검색트리에서 하나의 데이터를 찾는 평균 시간은 $1.38 \log_2 n$ 으로 매우 우수한 성능을 보인다.

최대값(최소값) 찾기

[problem]

다음 자료에서 최대값 또는 최소값을 찾는 방법을 고안하시오.

5	17	2	66	22	91	9	14	58	36

- 우리는 자료가 한 눈에 파악할 수 있는 정도의 개수이면 쉽게 최대값(최소값)을 알 수 있다.
- 그런데, 자료의 개수가 1,000개, 2,000개가 되면 쉽게 최대값(최소값)을 파악할 수 없다.
- 효과적인 방법이 있을까?

- 문제 해결 과정
  1.문제를 이해한다
  2.계획을 세운다
  3.계획을 실행한다
  4.풀이과정을 재점검한다

- Computational Thinking
  - 분해(decomposition)
  - 패턴 확인
  - 추상화(abstraction)
  - 알고리즘 설계
  - 데이터 모음
  - 데이터 표현
  - 데이터 분석
  - 모의실험
  - 자동화
  - 병렬화
  - 패턴 일반화

- 문제 해결 전략
  - 추정과 확인
  - 순서 리스트 작성
  - 가능성들을 제거
  - 대칭 성질을 이용
  - 특별한 경우를 고려
  - 직접 추론 사용
  - 방정식을 푼다
  - 패턴을 찾는다
  - 그림을 그린다
  - 단순한 문제 해결 시도
  - 모델 사용
  - 역방향으로 시도
  - 공식 사용
  - 창조적 생각
  - 잔꾀를 사용

- 데이터는 선형적으로 저장되어 있다고 가정한다.
  - ✓ 배열 또는 연결 리스트의 형태 가정
  - ✓ 데이터 내에 같은 값을 갖는 데이터는 없다고 가정한다. 모두 다른 값이라 가정
- 최소값 찾기 문제는 최대값 찾기 문제와 유사하게 풀 수 있다
  - ✓ 데이터의 비교 시 반대로 수행한다
- 먼저 최대값의 데이터를 찾는 문제를 먼저 다룬다.
- 제일 처음에 있는 데이터가 최대이더라도 나머지 데이터를 확인하지 않고서는 최대인지 확신할 수 없다.

[최대값을 찾는알고리즘]

(단계 1) 데이터의 제일 처음(제일 왼쪽)에 있는 데이터를 임시로 최대값 데이터 M 이라고 하자

(단계 2) 오른쪽 방향으로 이동하면서, 다음 데이터와 M의 값을 비교한다. 이 때 다음 두 가지 경우가 발생한다.

(경우 1) 만일 다음 데이터가 M보다 작으면 오른쪽 다음 데이터로 이동

(경우 2) 만일 다음 데이터가 M보다 크면 그 데이터를 M이라고 재설정한 후 오른쪽 다음 데이터로 이동

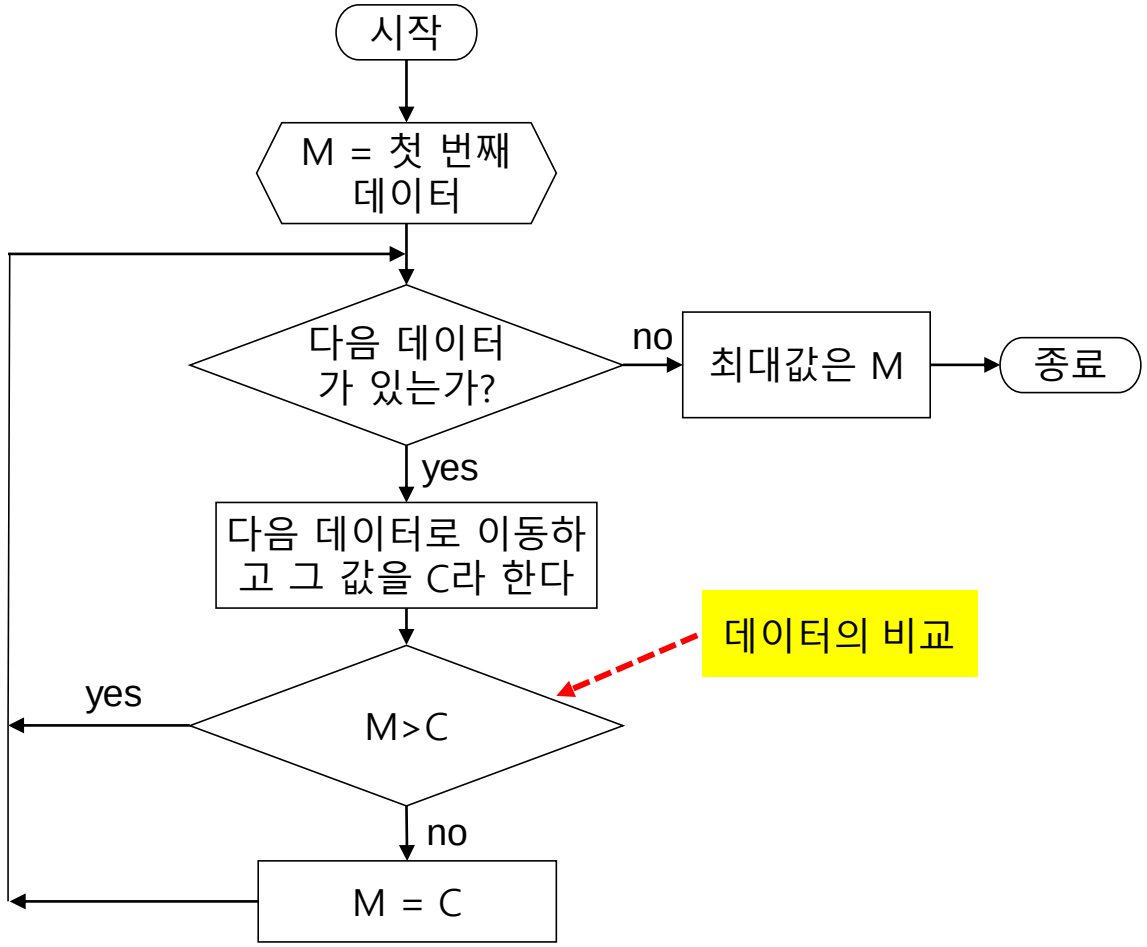
(단계 3)

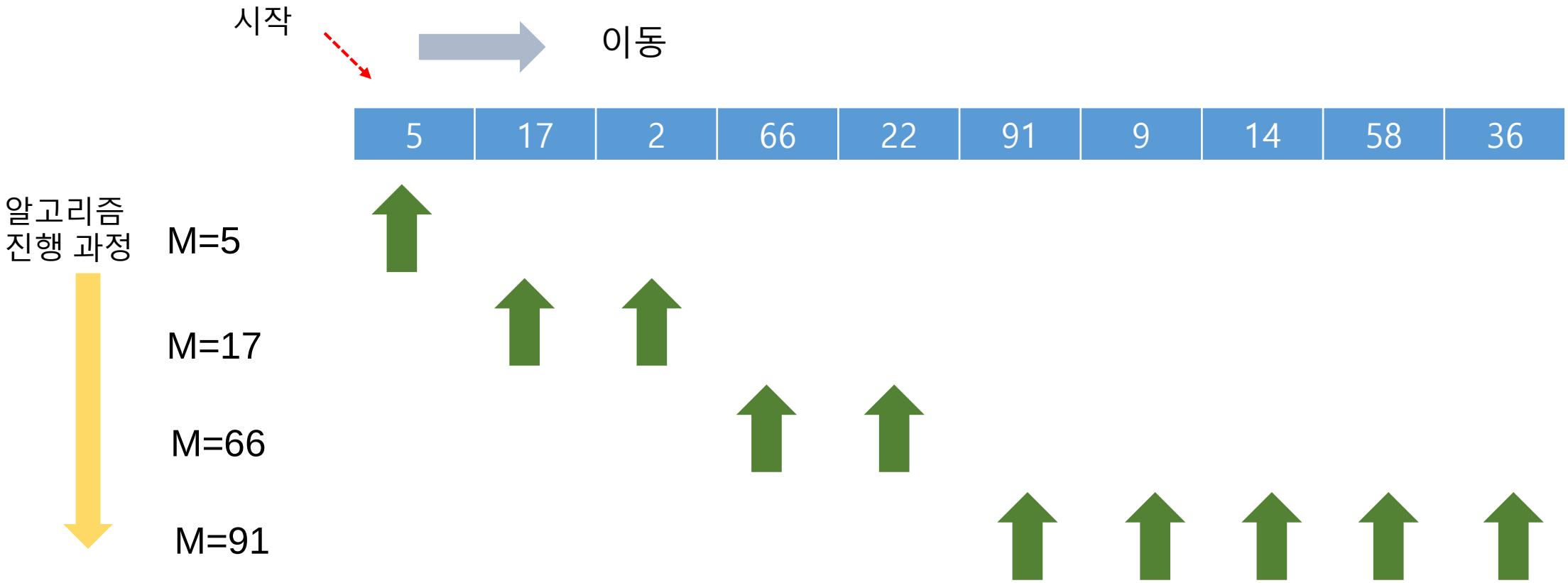
아직 확인해야 할 데이터가 남아 있으면 (단계 2)를 반복한다.

더 이상 데이터가 없으면 종료

- M은 자료 중의 최대값이다.

최대값을 찾는 알고리즘의 흐름도




시작
이동
5
17
2
66
22
91
9
14
58
36
알고리즘
진행 과정
M=5
M=17
M=66
M=91

[최소값을 찾는 알고리즘]

- 최대값을 찾는 방법에 일부 내용을 수정하면 완성할 수 있다.
- 최초 값을 최소값이라 설정한다.
- 이동하면서 데이터를 비교하는 과정에서 클 때 하는 작업을 작을 때에 수행한다.
- 최종 얻는 데이터 M은 자료 중의 최소값이다.

✓ 직접 알고리즘을 수정해 보시오

[최소값을 찾는 알고리즘]

(단계 1) 데이터의 제일 처음(제일 왼쪽)에 있는 데이터를 임시로 최소값 데이터 M이라고 하자

(단계 2) 오른쪽 방향으로 이동하면서, 다음 데이터와 M의 값을 비교한다. 이 때 다음 두 가지 경우가 발생한다.

(경우 1) 만일 다음 데이터가 M보다 크면 오른쪽 다음 데이터로 이동

(경우 2) 만일 다음 데이터가 M보다 작으면 그 데이터를 M이라고 재설정한 후 오른쪽 다음 데이터로 이동

(단계 3)

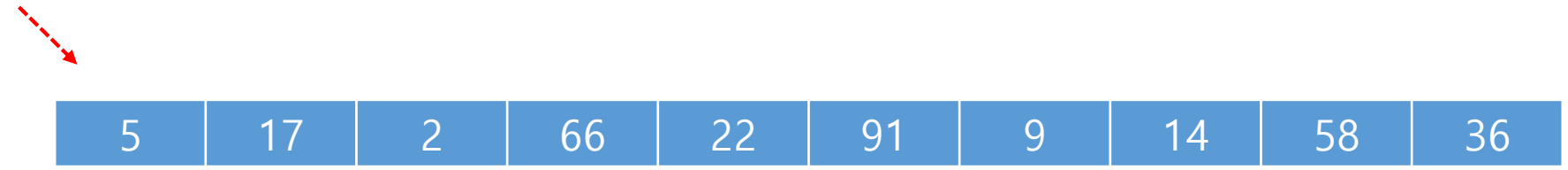
아직 확인해야 할 데이터가 남아 있으면 (단계 2)를 반복한다.

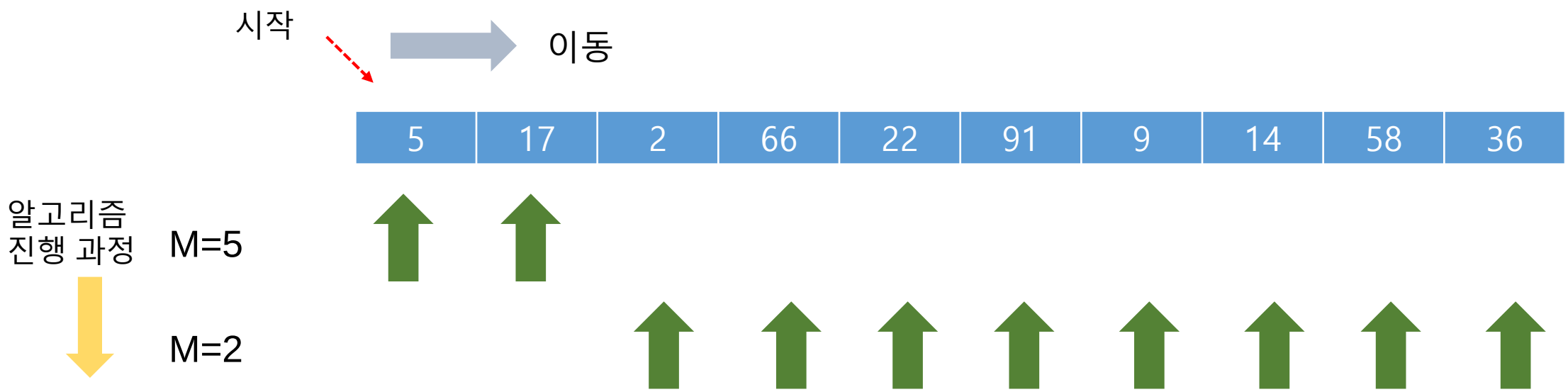
더 이상 데이터가 없으면 종료

- M은 데이터 중의 최소값이다.

다음 자료에 대해 최소값을 구하는 방법을 적용해 보시오.

시작




시작
이동
5
17
2
66
22
91
9
14
58
36
알고리즘
진행 과정
M=5
M=2

Computational thinking

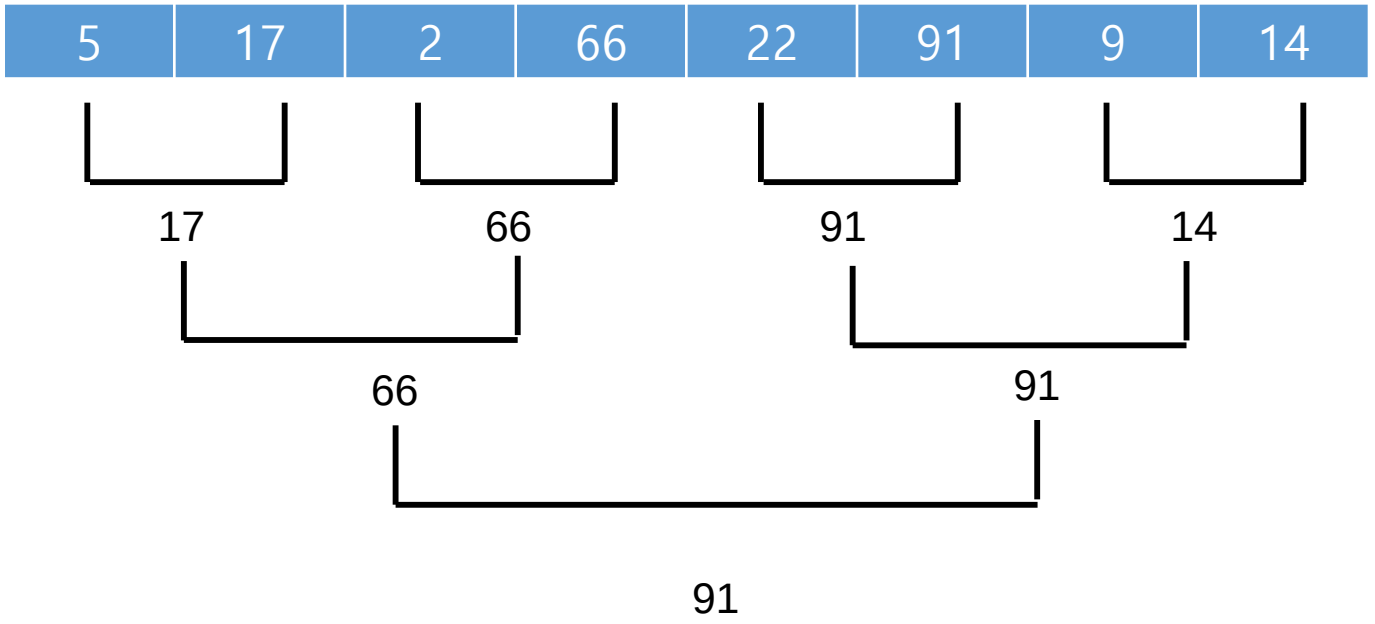
- 실 생활에서 이러한 방법을 쓰는 경우를 생각해 보시오.
- 데이터가 선형적이지 않은 구조에 저장되어 있을 때는 어떻게 하면 최대값(최소값) 데이터를 찾을 수 있을까?
- 데이터의 개수가 n일 때, 최대값(최소값)을 찾으려면 몇 번의 데이터 비교가 필요할까?
- 데이터가 선형적으로 저장되어 있을 때, 여기서 설명한 방법보다 더 빨리 최대값(최소값) 데이터를 찾는 방법이 있을까?

[풀이과정 재점검]

- 한 번 확인한 데이터는 더 이상 확인하지 않는다.
- 최대값(최소값)을 구하는 문제는 서로 대칭적인 문제이므로, 최대값을 구하는 해법은 간단한 변경으로 최소값을 찾는 문제에도 활용할 수 있다.

- 최대값(최소값)을 구하는 이 방법은 가장 기본적인 방법이다.
- 다른 방법으로 최대값(최소값)을 구할 수는 있으나, 이 방법보다 데이터의 비교횟수가 작은 방법은 존재하지 않는다.
- n개의 데이터가 있을 때 n-1번의 데이터 비교가 있어야만, 최대값(최소값)을 찾을 수 있다.
- n이 큰 경우, 최대값(최소값)을 찾는 시간이 많이 소요되므로, 다른 자료구조를 이용해서 데이터를 저장한다. – 트리 구조 사용

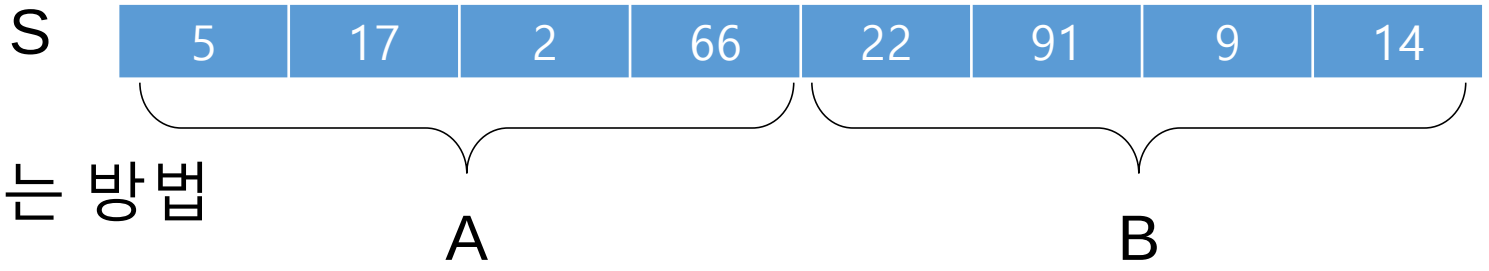
토너먼트 방법을 이용한 최대값 찾기



✓ 총 7회의 비교 필요

n개의 데이터 중에서 최대값을 찾는 알고리즘은
반드시 n-1 번의 데이터 비교가 필요하다.

[연습문제]



- 재귀를 이용하여 최대값을 찾는 방법을 설계해 보시오.
- 입력: S[1,…,n]

```
find_max(S)
    if |S|=2 then 두 원소 중 큰 값을 return
    else
        S를 두 배열 A, B로 나눈다. A[1,…,n/2], B[n/2+1,…,n]
        최대값 = max(find_max(A), find_max(B))
        return 최대값
```

✓ 재귀(recursion) 사용

✓ n-1 번의 데이터 비교 필요

최대값과 최소값을 모두 찾기

[problem]

데이터에 있는 최대값, 최소값을 모두 찾을 수 있는 효과적인 방법을 고안하시오.

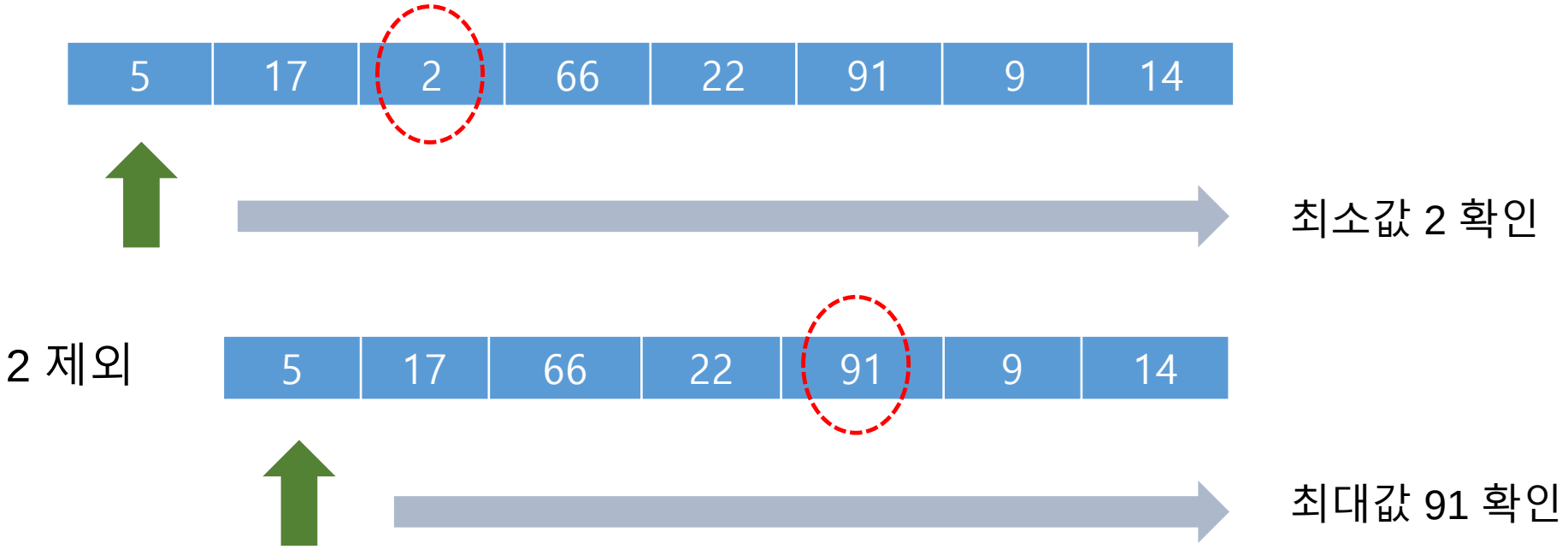
- 문제 해결 과정
  1.문제를 이해한다
  2.계획을 세운다
  3.계획을 실행한다
  4.풀이과정을 재점검한다

- Computational Thinking
  - 분해(decomposition)
  - 패턴 확인
  - 추상화(abstraction)
  - 알고리즘 설계
  - 데이터 모음
  - 데이터 표현
  - 데이터 분석
  - 모의실험
  - 자동화
  - 병렬화
  - 패턴 일반화

- 문제 해결 전략
  - 추정과 확인
  - 순서 리스트 작성
  - 가능성들을 제거
  - 대칭 성질을 이용
  - 특별한 경우를 고려
  - 직접 추론 사용
  - 방정식을 푼다
  - 패턴을 찾는다
  - 그림을 그린다
  - 단순한 문제 해결 시도
  - 모델 사용
  - 역방향으로 시도
  - 공식 사용
  - 창조적 생각
  - 잔꾀를 사용

[방법 1]

- 최소값을 먼저 찾고, 나머지 데이터에서 최대값을 찾는다.
- 이전에 설명한 방법을 적용한다



- 총 몇 번의 데이터 비교가 있었나? 7+ 6 = 13회

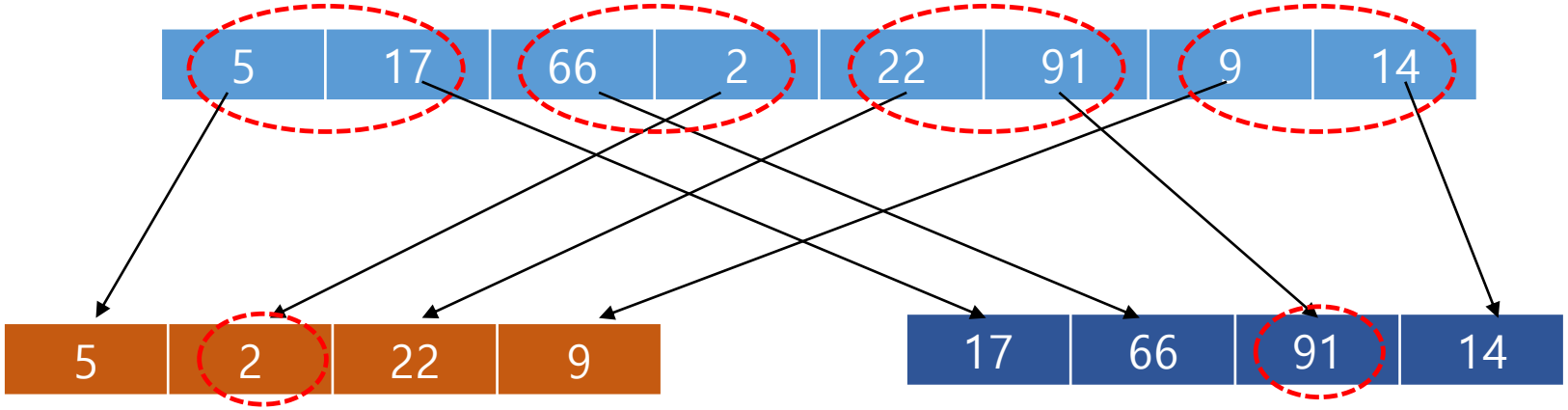
- 데이터의 비교 횟수를 줄일 수 있는 방법이 있을까?

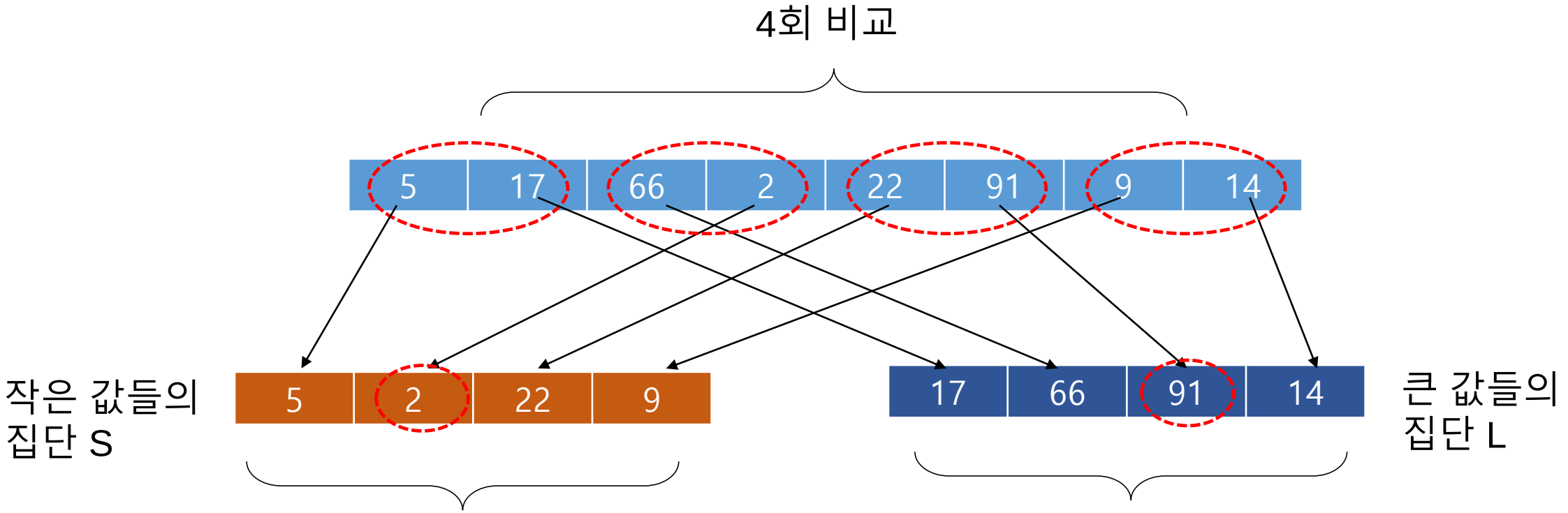
[방법 2]

(단계 1) 모든 데이터에 대해, 인접한 두 개의 데이터를 비교하여, 둘 중 작은 값, 큰 값을 구분한다.

(단계 2) 작은 값들의 집단 S, 큰 값들의 집단 L을 구성한다.

(단계 3) S에서 최소값을, L에서 최대값을 찾는다. – 일종의 함수




4회 비교
5
17
66
2
22
91
9
14
작은 값들의
집단 S
5
2
22
9
17
66
91
14
큰 값들의
집단 L
최소값 찾는데
3회 비교
최대값 찾는데
3회 비교

- 총 10회 비교

결론

- [방법 1]은 13회, [방법 2]는 10회의 데이터 비교가 필요하다.
- [방법 2]가 [방법 1] 보다 효과적이다.
- 대규모의 데이터가 있는 경우는 알고리즘 성능의 작은 차이도 문제해결능력에서 큰 차이를 발생시킬 수 있다.

[연습문제]

4, 3, 7, 8, 10, 2, 5, 1, 6, 9에서 최대값, 최소값을 찾으시오.

Computational thinking

- 두 방법간의 효율성의 차이는 어디로부터 기인했는지 생각해 보시오.
- 알고리즘의 초기 단계에서 작은 값들의 집단, 큰 값들의 집단으로 나누어서 불필요한 비교를 없앤다 – why?
- 이 경우 복합문제(최소값과 최대값을 동시에 구하는 문제)는 단위 문제(최소값 또는 최대값을 구하는 문제)의 합이 아니었다.

- n개의 데이터가 있는 자료에서

  [방법 1] 은 2n-3 회의 데이터 비교가 필요하고

  [방법 2]는 $\frac{3}{2}n-2$ (n이 짝수), $\frac{3}{2}n-\frac{3}{2}$ (n이 홀수)회의 데이터 비교가 필요하다.

11주차 강의 요약

데이터의 저장과 검색

- 이진검색트리
- 최대값 및 최소값 검색


11주차 끝
수고하셨습니다.