2025 겨울방학 알고리즘 스터디

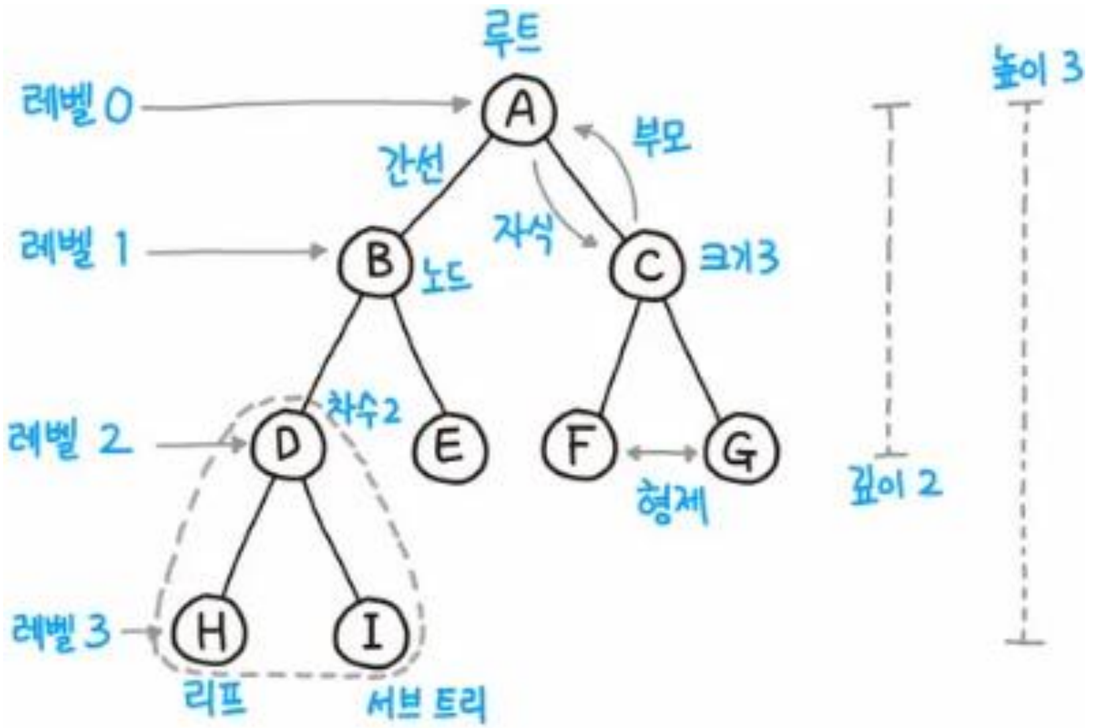
이진 트리의 종류와 이진 검색 트리

컴퓨터 공학과 20230546 서보경

목차

1. 기본적인 이진 트리들
2. 이진 검색 트리란?
3. 이진 검색 트리의 삽입과 검색
4. 이진 검색 트리의 삭제

이진 트리의 특징



계층적 구조로 부모와 자식 관계를 가진다.
(한 노드의 부모는 루트 노드가 아닌 이상 무조건 하나다.)

각 노드는 **최대 2개**의 자식을 가진다.

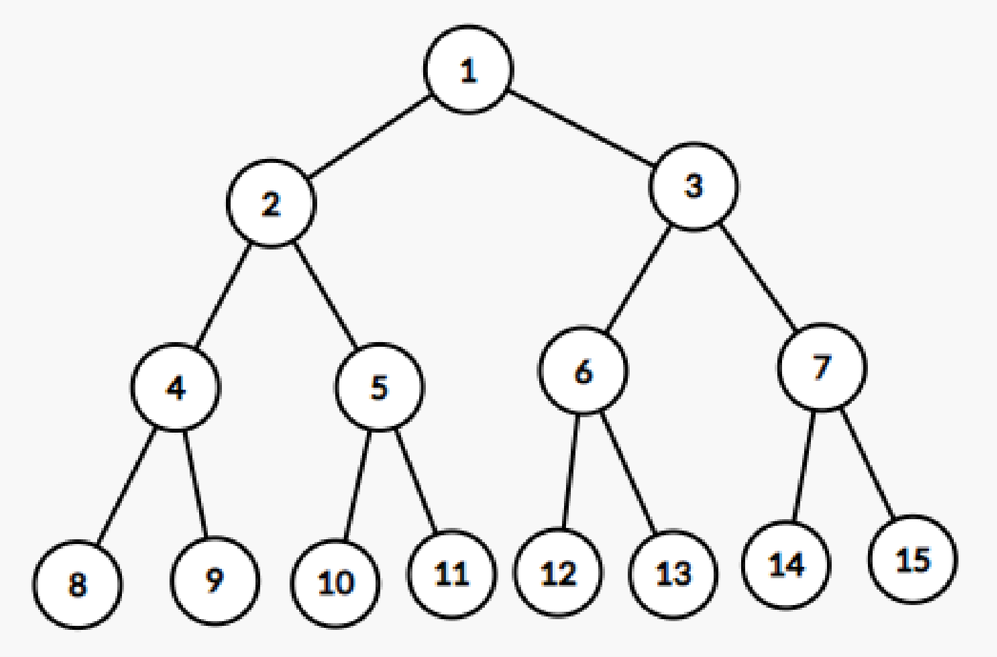
최대 차수가 2이다.

자식은 **왼쪽 자식**과 **오른쪽 자식**으로 구분한다.

트리의 높이는 루트 노드에서 가장 깊은 리프 노드까지의 거리

등등...

포화 이진 트리

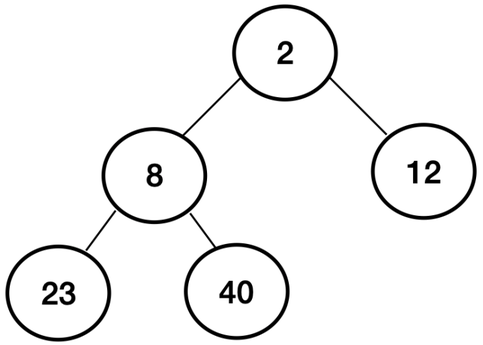


모든 노드가 0개 또는 2개의 자식을 가진다.

모든 리프 노드가 동일한 깊이에 존재한다.

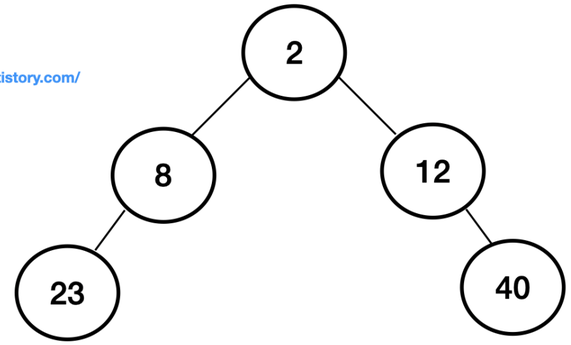
노드 개수 : 2^(h+1) – 1
*h = 트리의 높이

완전 이진 트리



완전 이진 트리(O)

https://heytech.tistory.com/



완전 이진 트리(X)

마지막 레벨을 제외한 모든 레벨이 완전히 채워진 형태.

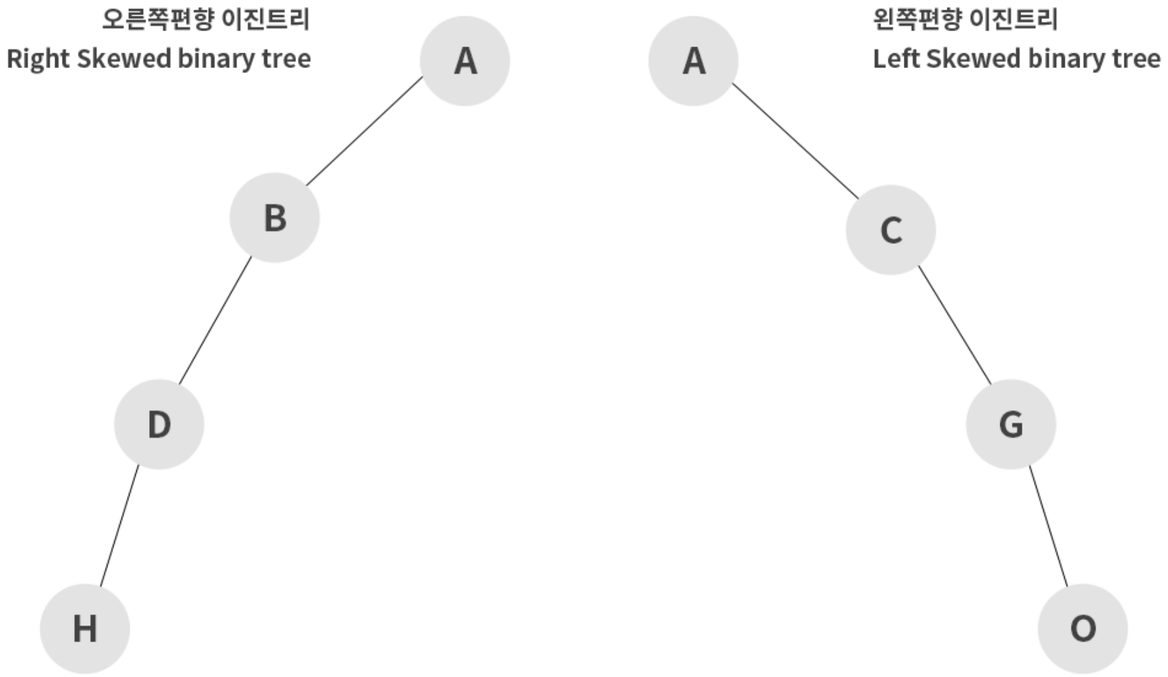
마지막 레벨은 왼쪽부터 차례대로 채워진다.

주로 배열로 구현한다.

Heap 구조의 기반이다.

완전 이진 트리 ⊆ 포화 이진 트리
(역은 성립하지 않음)

편향 이진 트리



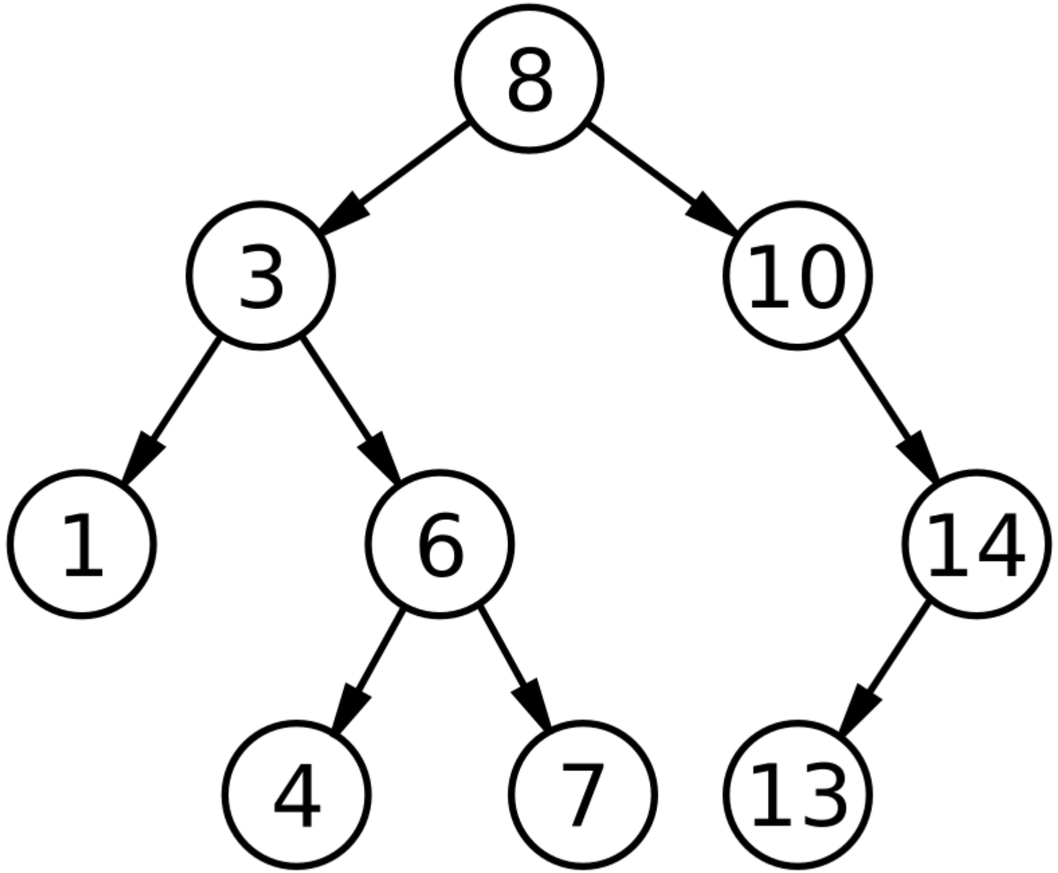
한쪽 자식만 계속 이어지는 트리 구조.

트리가 배열과 유사하게 동작한다.

왼쪽 편향 : 모든 노드가 왼쪽 자식만 가진다.

오른쪽 편향 : 모든 노드가 오른쪽 자식만 가진다.

이진 검색 트리란?



이진 탐색 트리(Binary Search Tree)란 이진 트리의 한 종류로, 효율적인 데이터 탐색, 삽입, 삭제를 가능하게 하는 구조를 일컫는다.

C언어 에서는 주로 포인터 연산으로 구현하게 되며 제일 쉽게 구현할 수 있는 트리 구조 중 하나이다.

노드 구조 : 각 노드에는 값, 왼쪽 자식, 오른쪽 자식이 있다.

노드 값의 조건
- 왼쪽 서브 트리의 모든 노드 값은 부모 노드의 값보다 작다.
- 오른쪽 서브 트리의 모든 노드 값은 부모 노드 값보다 크다.
- 각 서브 트리에도 조건이 재귀적으로 적용된다.

일반적으로 중복된 데이터를 허용하지 않는다.

이진 검색 트리의 삽입

문제 상황 : 값 8을 삽입하기

루트가 존재 하지 않으므로 **루트 자리에 8을 삽입**한다.

항상 트리구조의 시작은 **루트**부터 시작한다.

삽입 절차가 끝났으므로 복귀한다.

이진 검색 트리의 삽입

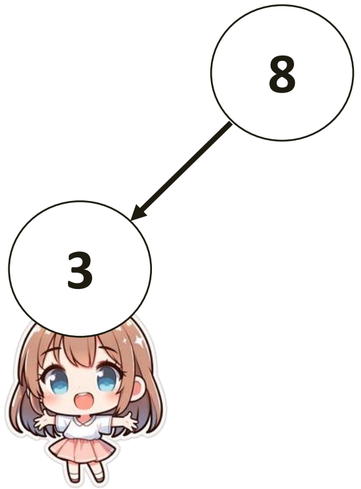
문제 상황 : 값 3을 삽입하기

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

8(현재 노드) > 3(삽입할 값)이므로 삽입 값이 더 작다.

고로 왼쪽으로 이동한다.

이진 검색 트리의 삽입



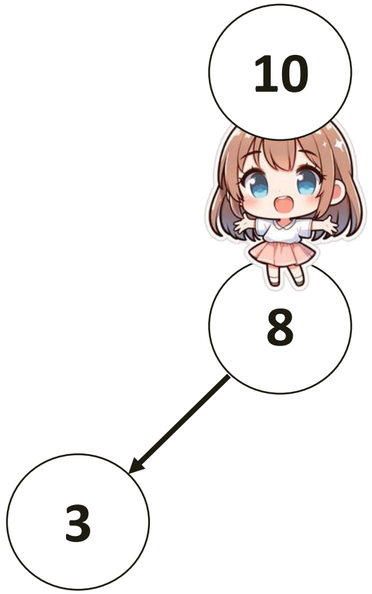
문제 상황 : 값 3을 삽입하기

왼쪽으로 이동했지만 아무것도 존재하지 않는다.

그렇다면 중복되는 값이 없다는 의미이므로 삽입을 해준다.

삽입 절차가 끝났으므로 복귀한다.

이진 검색 트리의 삽입



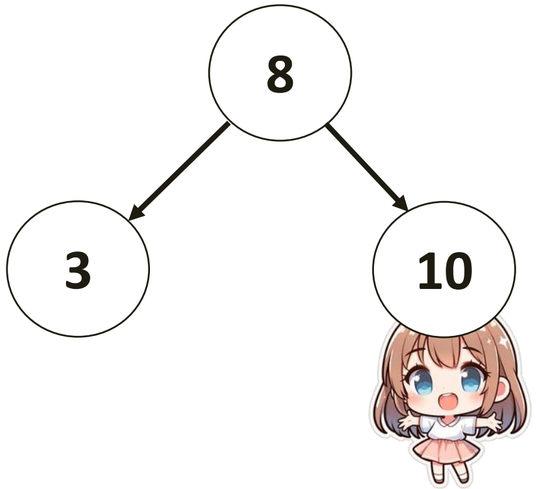
문제 상황 : 값 10을 삽입하기

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

8(현재 노드) < 10(삽입할 값)이므로 삽입 값이 더 크다.

고로 오른쪽으로 이동한다.

이진 검색 트리의 삽입



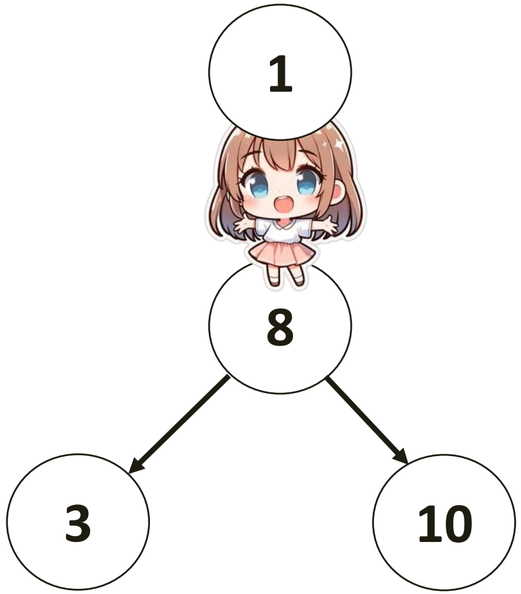
문제 상황 : 값 10을 삽입하기

오른쪽으로 이동했지만 아무것도 존재하지 않는다.

그렇다면 중복되는 값이 없다는 의미이므로 삽입을 해준다.

삽입 절차가 끝났으므로 복귀한다.

이진 검색 트리의 삽입



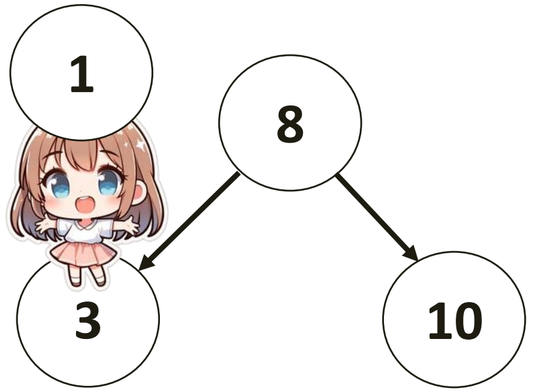
문제 상황 : 값 1을 삽입하기

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

8(현재 노드) > 1(삽입할 값)이므로 삽입 값이 더 작다.

고로 왼쪽으로 이동한다.

이진 검색 트리의 삽입



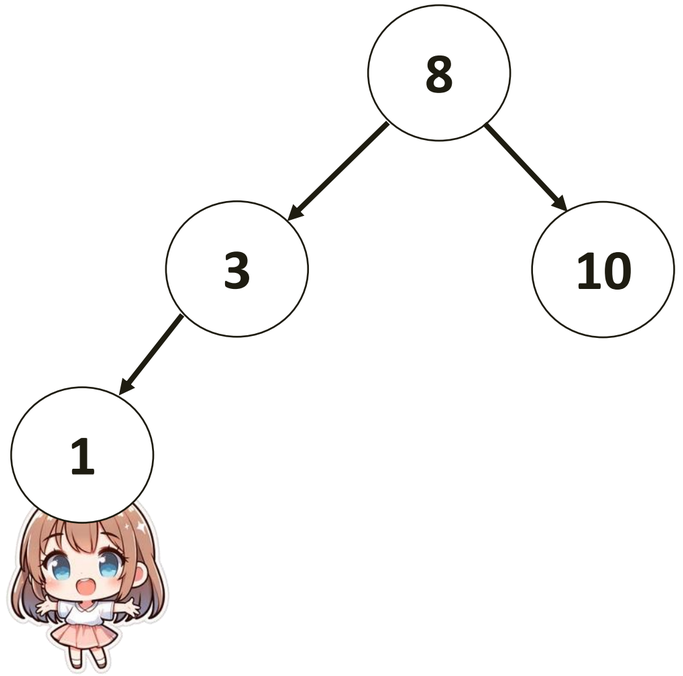
문제 상황 : 값 1을 삽입하기

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

3(현재 노드) > 1(삽입할 값)이므로 삽입 값이 더 작다.

고로 왼쪽으로 이동한다.

이진 검색 트리의 삽입



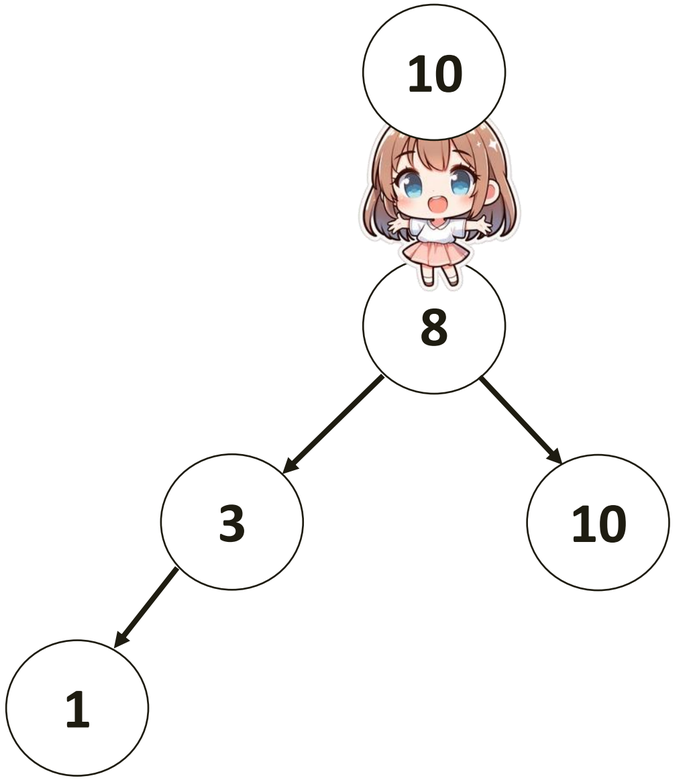
문제 상황 : 값 1을 삽입하기

왼쪽으로 이동했지만 아무것도 존재하지 않는다.

그렇다면 중복되는 값이 없다는 의미이므로 삽입을 해준다.

삽입 절차가 끝났으므로 복귀한다.

이진 검색 트리의 삽입



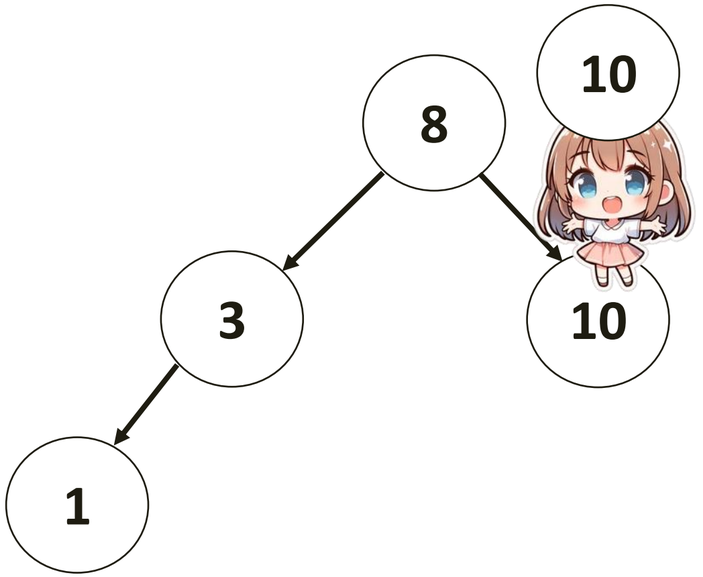
문제 상황 : 값 10을 삽입하기

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

3(현재 노드) > 10(삽입할 값)이므로 삽입 값이 더 크다.

고로 오른쪽으로 이동한다.

이진 검색 트리의 삽입



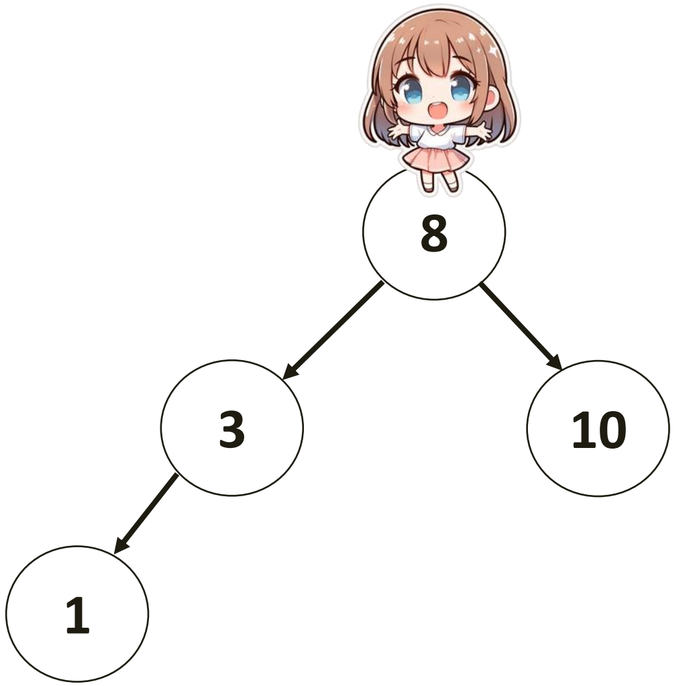
문제 상황 : 값 10을 삽입하기

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

10(현재 노드) == 10(삽입할 값)이므로 삽입 값과 현재 값이 **같다**.

이진 검색 트리는 일반적으로 중복을 허용하지 않으므로 **아무것도 하지 않고 복귀**한다.

이진 검색 트리의 검색



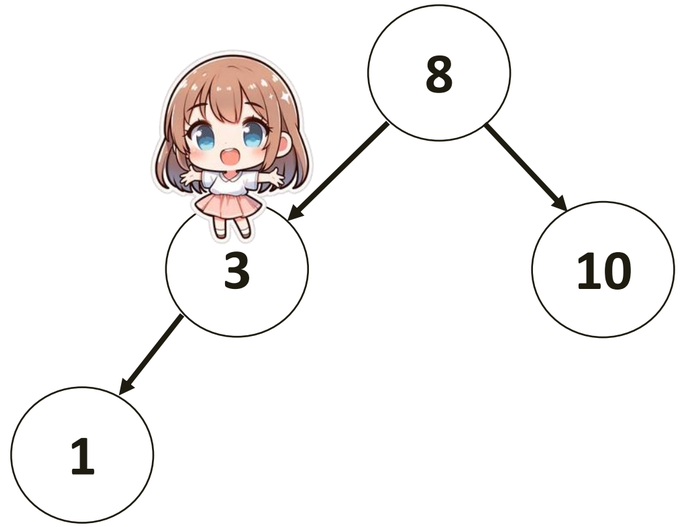
문제 상황 : 값 7을 찾기

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

8(현재 노드) > 7(찾을 값)이므로 찾을 값이 더 작다.

고로 왼쪽으로 이동한다.

이진 검색 트리의 검색



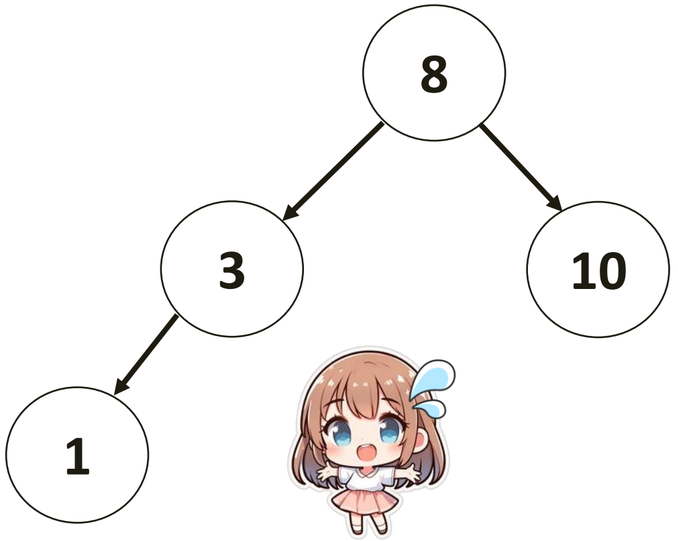
문제 상황 : 값 7을 찾기

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

3(현재 노드) < 7(찾을 값)이므로 찾을 값이 더 크다.

고로 오른쪽으로 이동한다.

이진 검색 트리의 검색

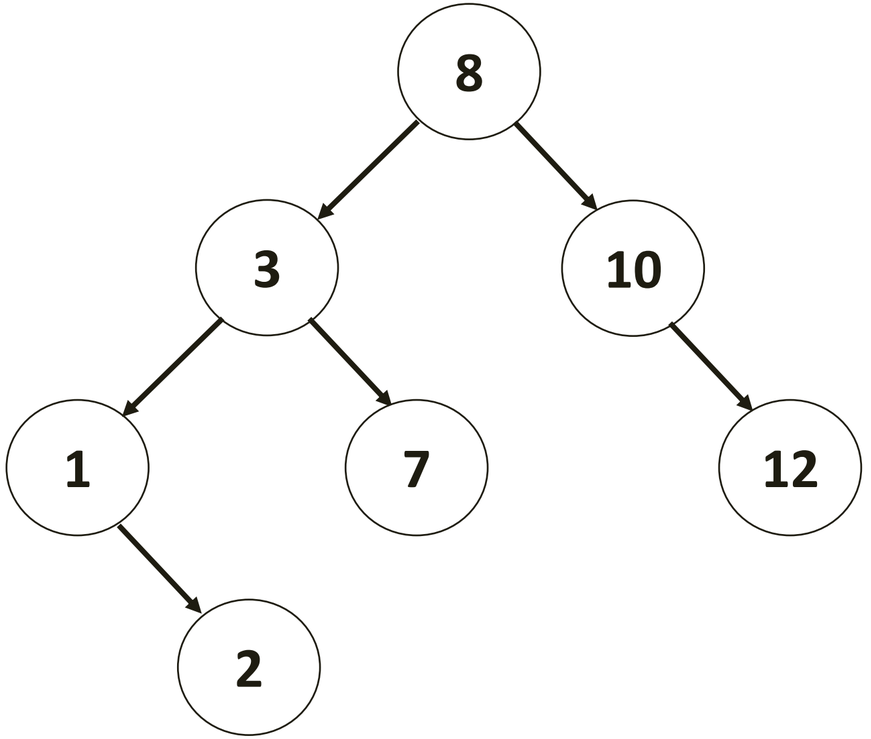


문제 상황 : 값 7을 찾기

노드가 존재 하지 않는다.

탐색 과정에서 7을 찾지 못했으므로 실패 했다는 메시지를 들고 복귀한다.

이진 검색 트리의 삭제



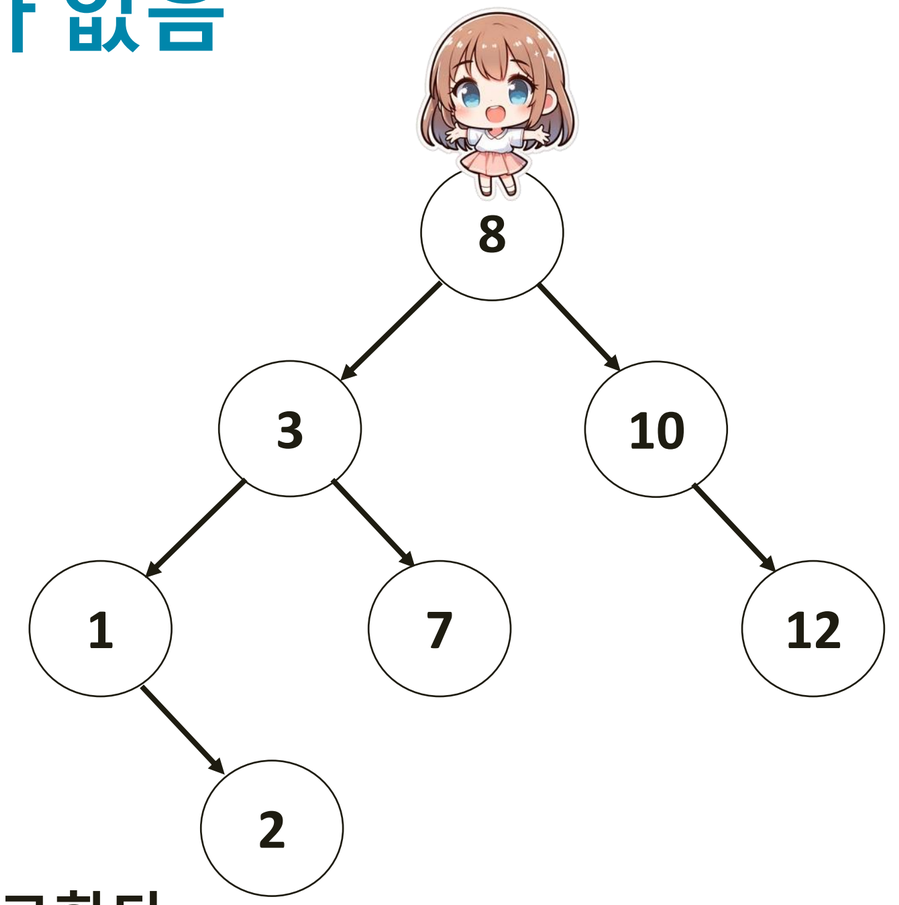
이진 검색 트리 삭제의 유형

1 – 자식 노드가 없는 경우

2 – 자식 노드가 1개인 경우

3 – 자식 노드가 2개인 경우

이진 검색 트리의 삭제 – 자식 노드가 없음



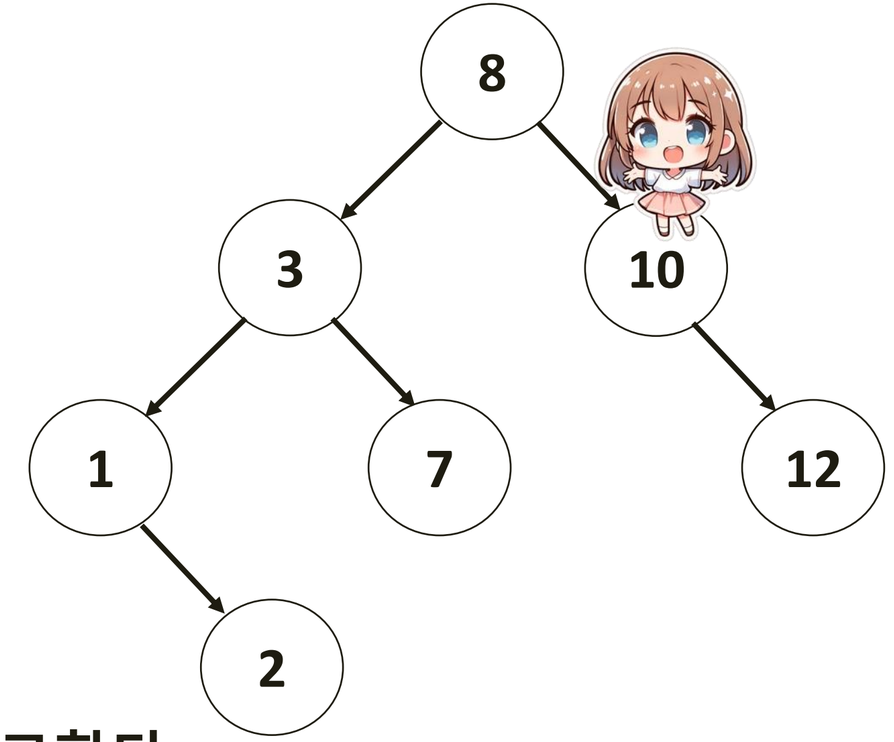
문제 상황 : 값 12를 삭제

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

8(현재 노드) < 12(삭제할 값)이므로 삭제 값이 더 크다.

고로 오른쪽으로 이동한다.

이진 검색 트리의 삭제 – 자식 노드가 없음



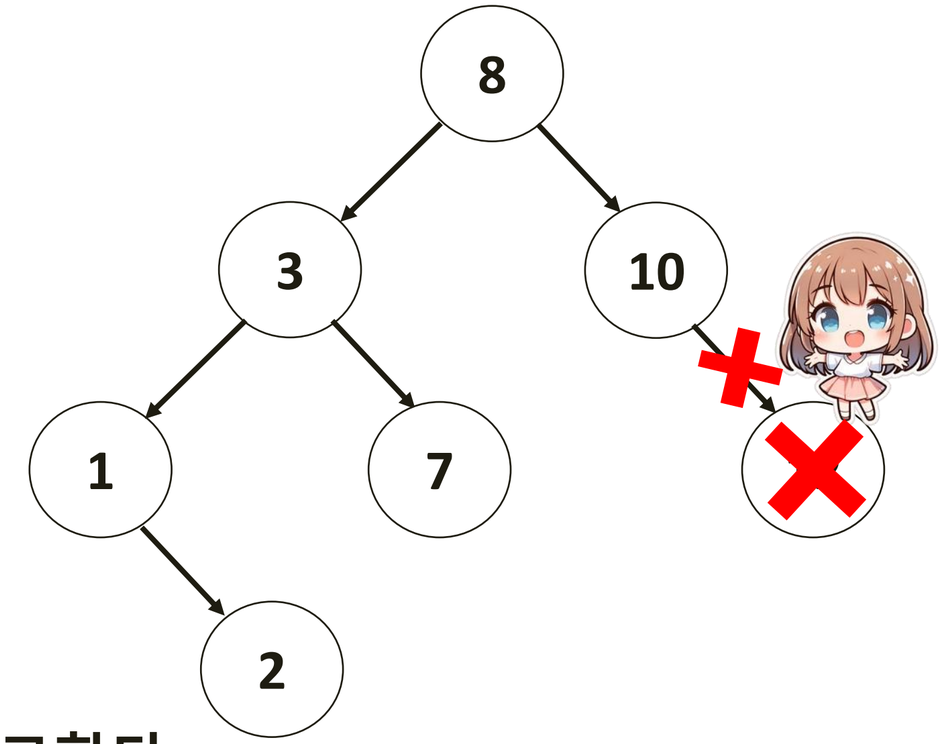
문제 상황 : 값 12를 삭제

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

10(현재 노드) < 12(삭제할 값)이므로 삭제 값이 더 크다.

고로 오른쪽으로 이동한다.

이진 검색 트리의 삭제 – 자식 노드가 없음



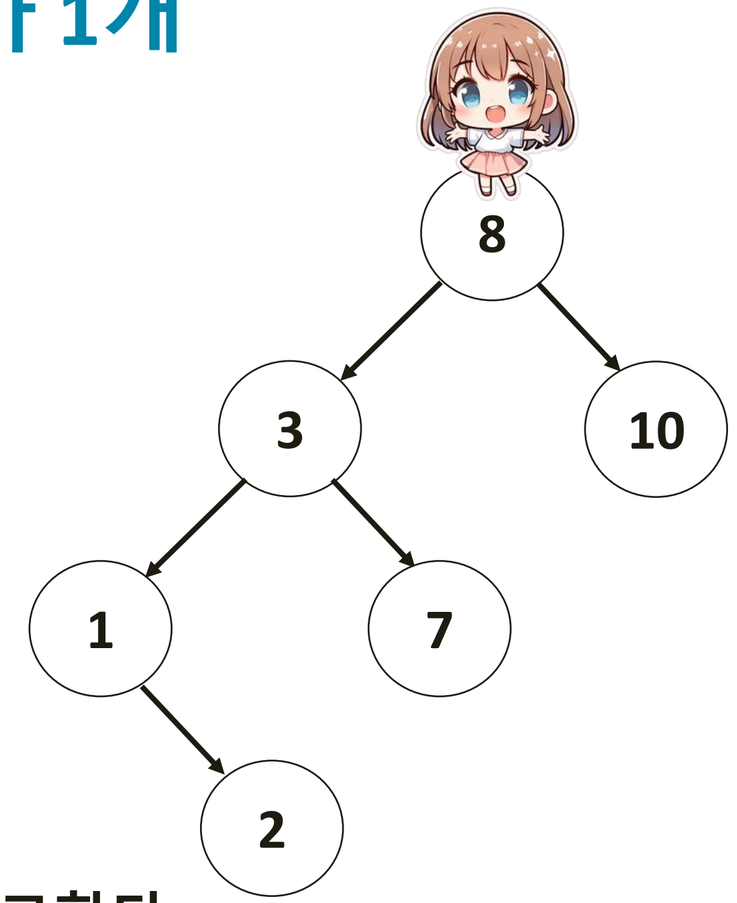
문제 상황 : 값 12를 삭제

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

12(현재 노드) == 12(삭제할 값)이므로 현재 노드의 값과 삭제 노드의 값이 같다!

현재 노드를 삭제하고 복귀한다.

이진 검색 트리의 삭제 – 자식 노드가 1개



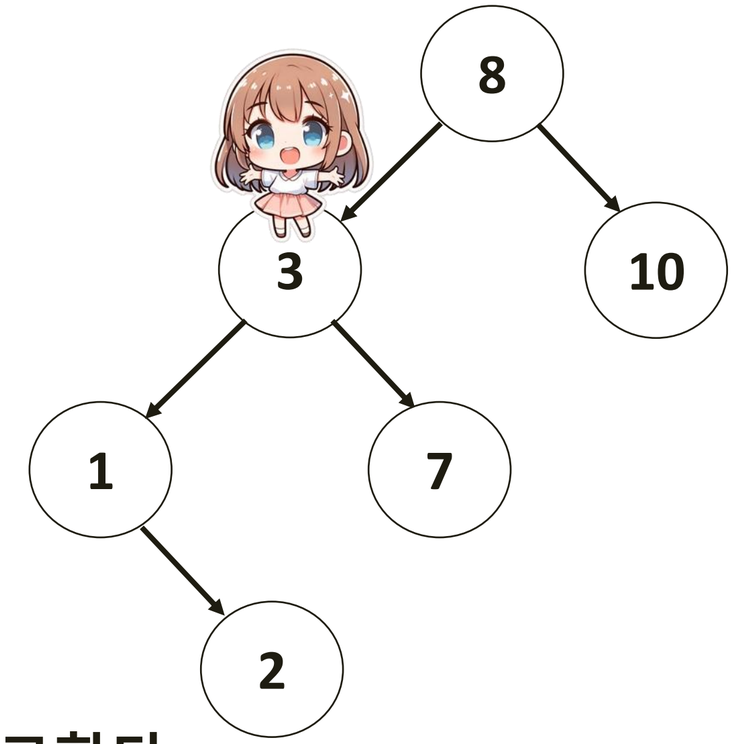
문제 상황 : 값 1를 삭제

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

8(현재 노드) < 1(삭제할 값)이므로 삭제 값이 더 작다.

고로 왼쪽으로 이동한다.

이진 검색 트리의 삭제 – 자식 노드가 1개



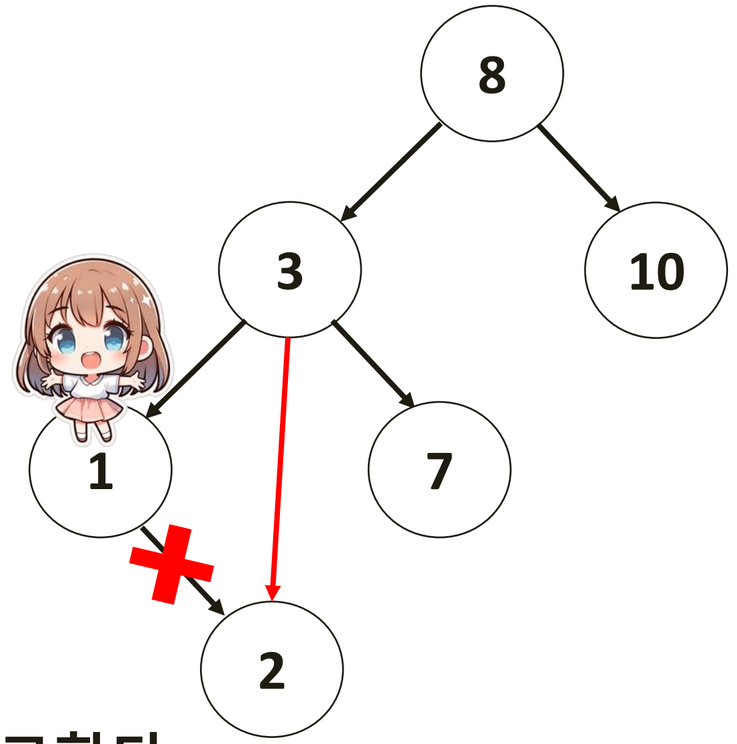
문제 상황 : 값 1를 삭제

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

3(현재 노드) < 1(삭제할 값)이므로 삭제 값이 더 작다.

고로 왼쪽으로 이동한다.

이진 검색 트리의 삭제 – 자식 노드가 1개



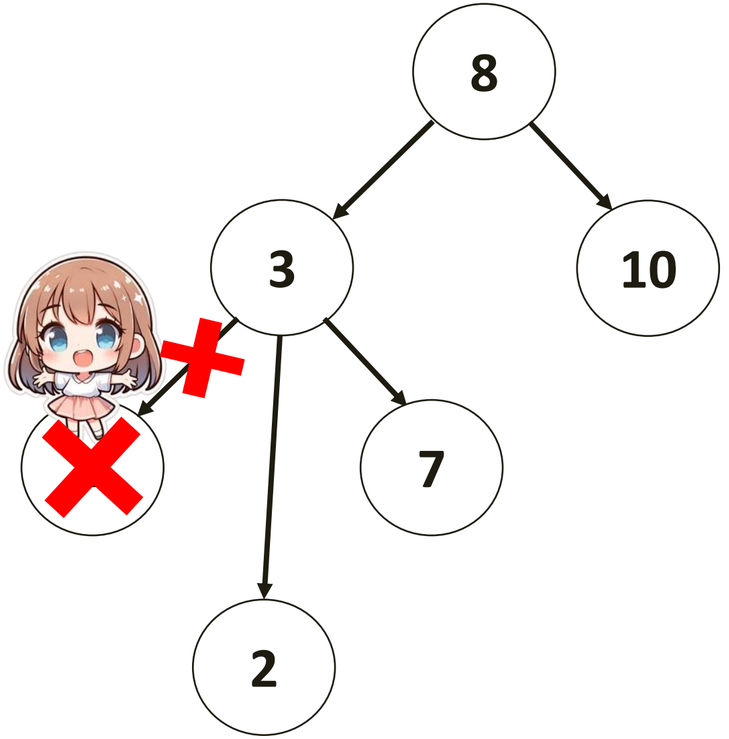
문제 상황 : 값 1를 삭제

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

1 (현재 노드) == 1 (삭제할 값)이므로 현재 노드의 값과 삭제 노드의 값이 같다!

현재 노드에겐 자식이 1개 있으므로 그 자식(2)을 삭제할 노드의 부모(3)와 연결 시켜준다.

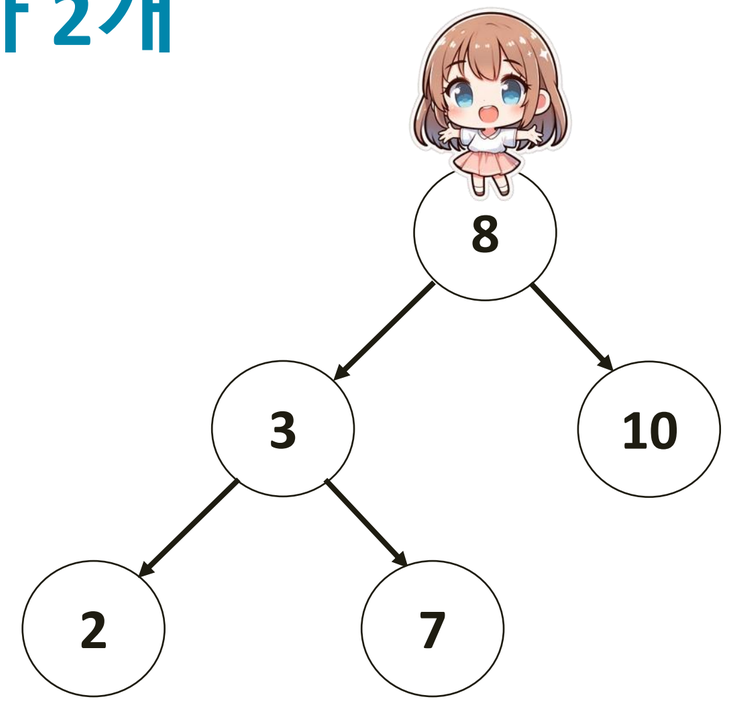
이진 검색 트리의 삭제 – 자식 노드가 1개



문제 상황 : 값 1를 삭제

이후, 현재 노드(1)을 삭제 시켜준 후 복귀한다.

이진 검색 트리의 삭제 – 자식 노드가 2개



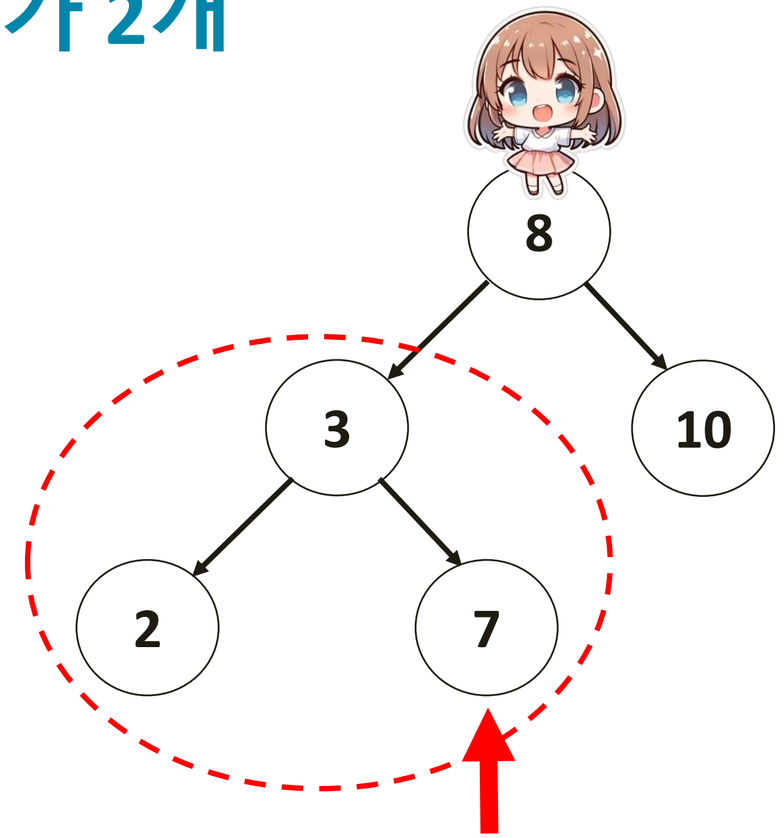
문제 상황 : 값 8를 삭제

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

8(현재 노드) == 8(삭제할 값)이므로 현재 노드의 값과 삭제 노드의 값이 **같다**!

현재 노드에겐 자식이 2개 있다. 이번 삭제 방법은 조금 더 **특이**하다.

이진 검색 트리의 삭제 – 자식 노드가 2개



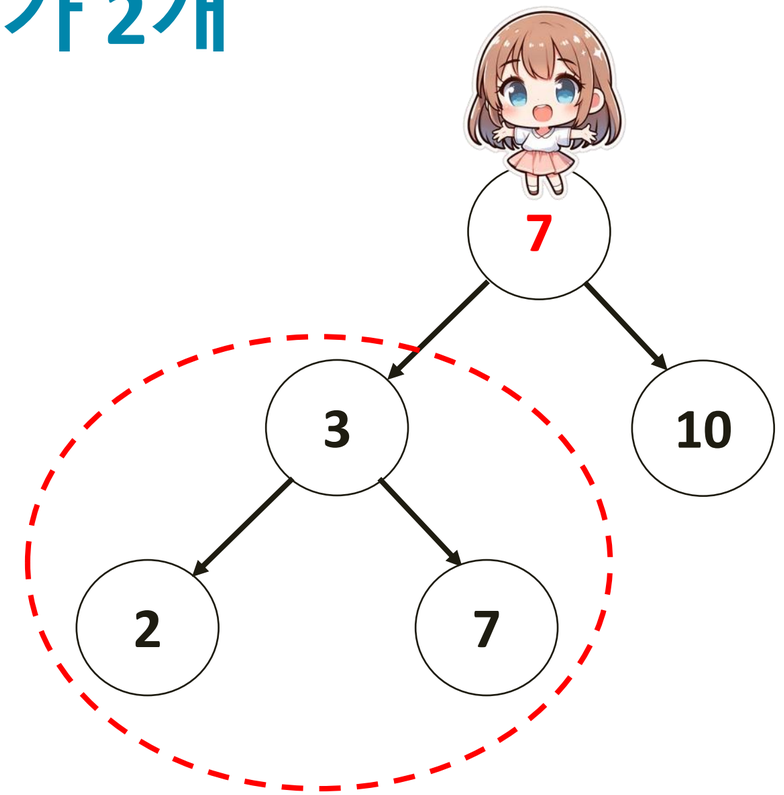
문제 상황 : 값 8를 삭제

첫번째로, 왼쪽 서브 트리에서 제일 큰 값을 찾는다.

자명하게, 왼쪽 서브 트리에서 제일 큰 값은 제일 오른쪽에 있는 값일 것이다.
(BST 특성상 현재 노드 보다 큰 값이면 오른쪽에 배치하기 때문)

7을 찾았다!

이진 검색 트리의 삭제 – 자식 노드가 2개

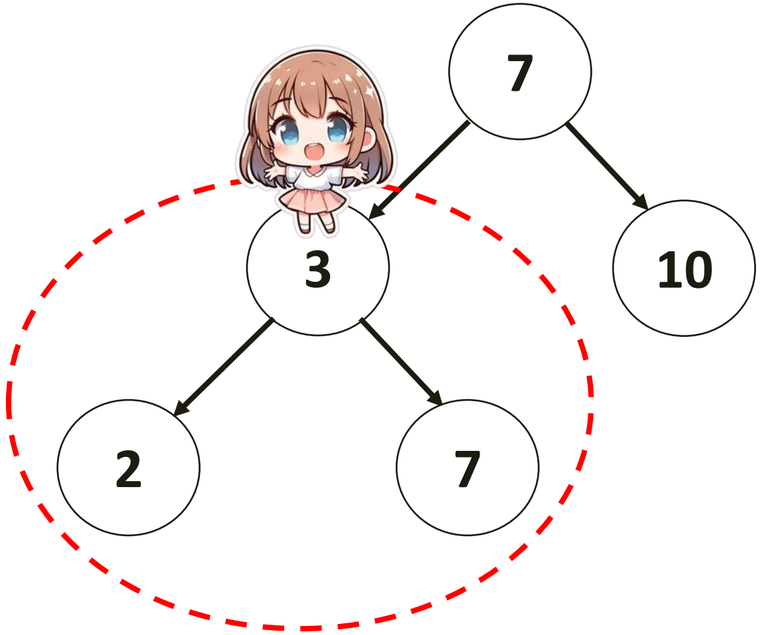


문제 상황 : 값 8를 삭제

찾은 7을 삭제할 노드에 대입 한다.

이러면 삭제 노드의 데이터는 자연적으로 소멸 되며 (덮어 쓰기),
트리 내에 2개의 7이 존재한다.

이진 검색 트리의 삭제 – 자식 노드가 2개

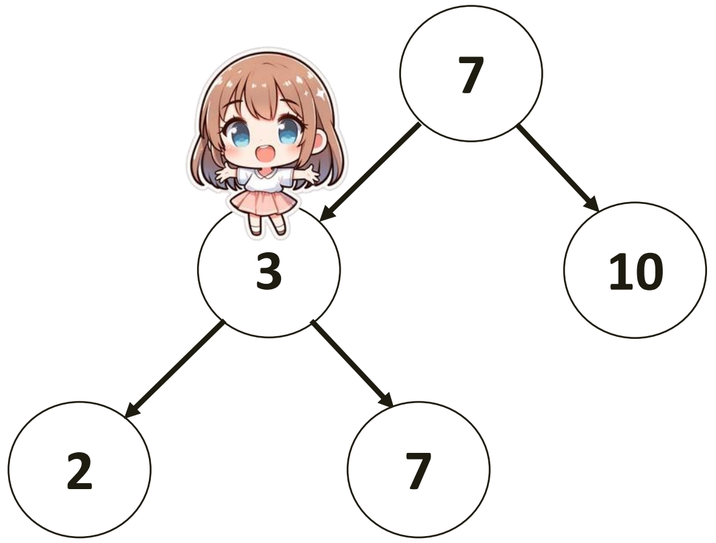


문제 상황 : 값 8를 삭제

이진 검색 트리는 중복된 값을 허용 하지 않으므로 원본인 7을 삭제 해야한다.

고로, 왼쪽 서브 트리로 이동한다.

이진 검색 트리의 삭제 – 자식 노드가 2개



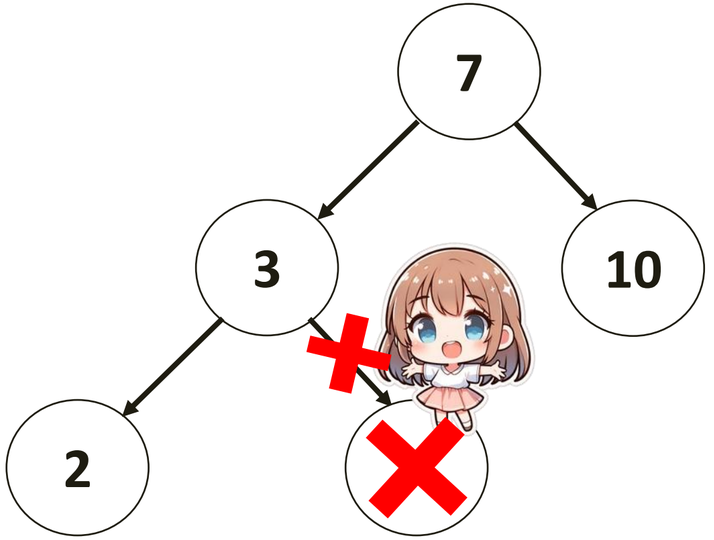
문제 상황 : 값 8를 삭제

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

3(현재 노드) < 7(삭제할 값)이므로 삭제 값이 더 크다.

고로 오른쪽으로 이동한다.

이진 검색 트리의 삭제 – 자식 노드가 2개



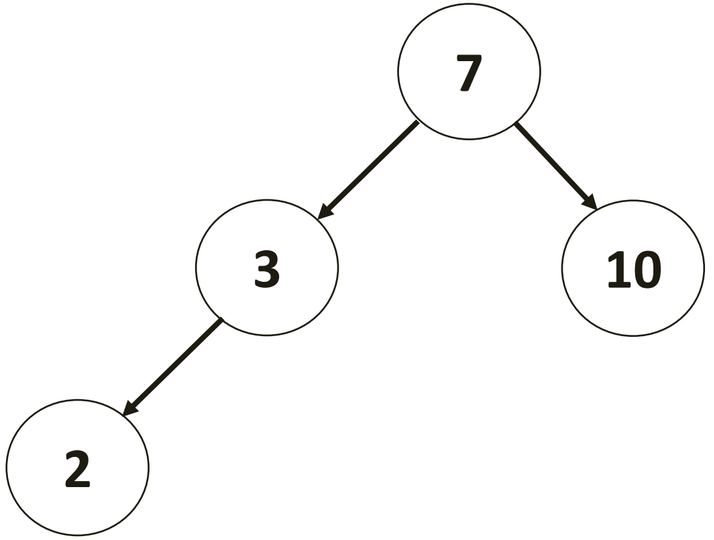
문제 상황 : 값 8를 삭제

현재 노드가 존재하므로 삽입할 값과 현재 노드의 값을 비교한다.

7(현재 노드) == 7(삭제할 값)이므로 현재 노드의 값과 삭제 노드의 값이 같다!

노드를 삭제 시켜주고 복귀한다.

이진 검색 트리의 삭제 – 자식 노드가 2개



본 예제 에서는 현재 노드의 왼쪽 서브 트리의 제일 큰 값으로 교체하는 방법을 택했다.

그러나, 오른쪽 서브 트리의 제일 작은 값으로 교체해도 무방하다.

왜일까??

이진 검색 트리의 삭제 – 자식 노드가 2개

이유

이진 탐색 트리는 다음 조건을 만족해야 한다.
- 왼쪽 서브 트리의 모든 값은 부모 노드보다 작아야 한다.
- 오른쪽 서브 트리의 모든 값은 부모 노드보다 커야 한다.

왼쪽 서브 트리의 최댓값.
- 왼쪽 서브 트리의 가장 큰 값은 부모 노드보다 작고, 왼쪽 서브 트리 내에서는 가장 크다.
- 따라서 이 값을 삭제할 노드의 위치로 대체하면, BST의 조건이 유지 된다.

오른쪽 서브 트리의 최소값.
- 오른쪽 서브 트리의 가장 작은 값은 부모 노드보다 크고, 오른쪽 서브 트리 내에서는 가장 작다.
- 따라서 이 값을 삭제할 노드의 위치로 대체하면, BST의 조건이 유지 된다.

왼쪽 서브 트리의 최댓값 또는 오른쪽 서브 트리의 최솟값은 항상 리프 노드 이거나, 자식이 하나이다.
- 즉, 대체 노드를 삭제하는 경우 단순히 해당 노드를 부모와 끊는 것으로 간단하게 처리 가능하다.
- 덕분에 삭제 과정이 단순화되고, 추가적인 트리 구조 조정이 필요 없게 된다.