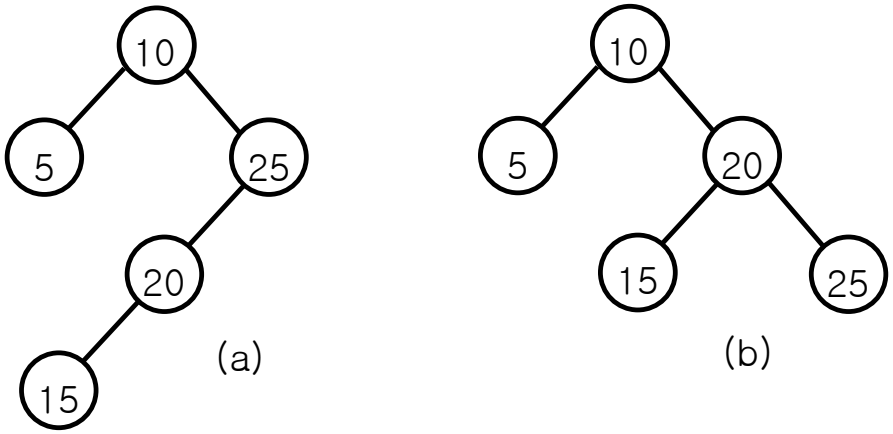
제 10 장 효율적 이원탐색트리

10.1 최적 이진탐색트리 (optimal binary search tree; OBST)

- 트리에 대한 삽입이나 삭제가 없으며, 오직 탐색만이 수행된다(정적 심볼테이블).
- 이진탐색트리의 비용구하기

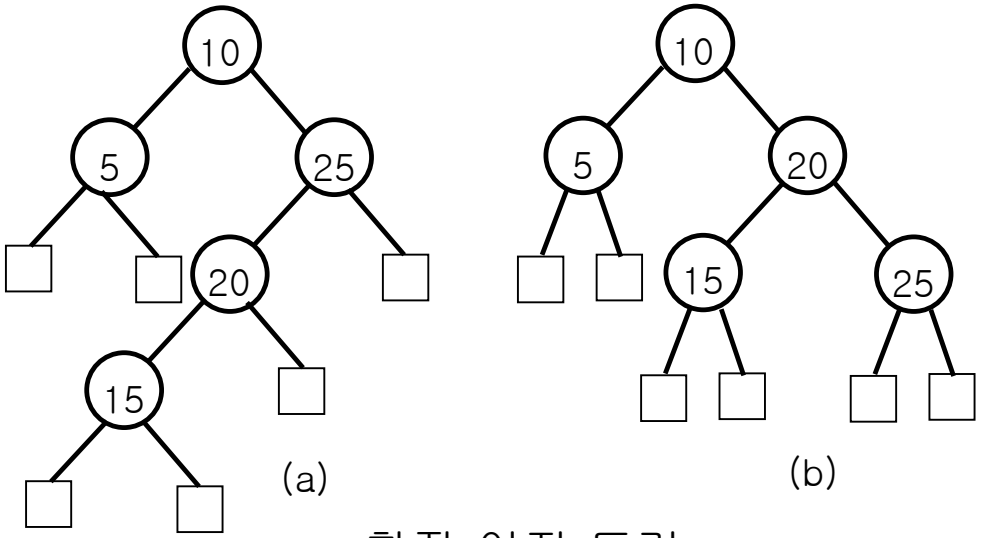


두 개의 이진 탐색 트리

- 각 식별자를 찾는 비용은 트리에서 식별자가 위치한 깊이다.
- 같은 확률로 검색할 때, 평균 비용은 (a) (1+2+2+3+4)/5 = 2.4
  (b) (1+2+2+3+3)/5 = 2.2 이다.

확장된 이진탐색트리

- 널 링크가 있는 곳에 특별한 사각형 노드를 붙인 트리이다.
- n개의 노드를 가진 트리는 n+1개의 널 링크를 가지므로 n+1개의 사각형 노드가 필요하다(사각형 노드를 실패노드(failure node)라 한다).



확장 이진 트리

- 외부 경로 길이(external path length):
  E = 루트에서부터 외부 노드들까지의 경로 길이들의 합
- 내부 경로 길이(internal path length):
  I = 루트에서부터 내부 노드들까지의 경로 길이들의 합
- 예: 그림 10.3(a)에서
  E = 2 + 2 + 4 + 4 + 3 + 2 = 17
  I = 0 + 1 + 1 + 2 + 3 = 7
- 이때, E = I + 2n 의 관계가 성립된다(n: 내부 노드의 수).
- 최소 내부 경로 길이(I)를 갖는 것은 완전 이진 트리(complete binary tree)이다.

최적 이진 탐색 트리의 정의

- 정의:

  (i) $a_1 < a_2 < \ldots < a_n$ 은 n 개의 식별자

  (ii) a_i 를 검색할 확률은 p_i (successful search)

  (iii) $a_i < x < a_{i+1}$ 인 식별자를 검색할 확률 q_i (unsuccessful search)

이때,

이진 탐색 트리의 총비용:

$$C = \sum_{1 \le i \le n} p_i \cdot \text{레벨}(a_i) + \sum_{0 \le i \le n} q_i \cdot (\text{레벨}(\text{실패노드 } i) - 1)$$

최적 이진 탐색 트리는 n 개의 식별자를 갖는 모든 가능한 이진 탐색 트리들중 총비용 C를 최소화하는 이진 탐색 트리이다.

- 예제 10.1: 식별자 집합 (a1, a2, a3) = (do, if, while)
  모든 가능한 이진 탐색 트리:

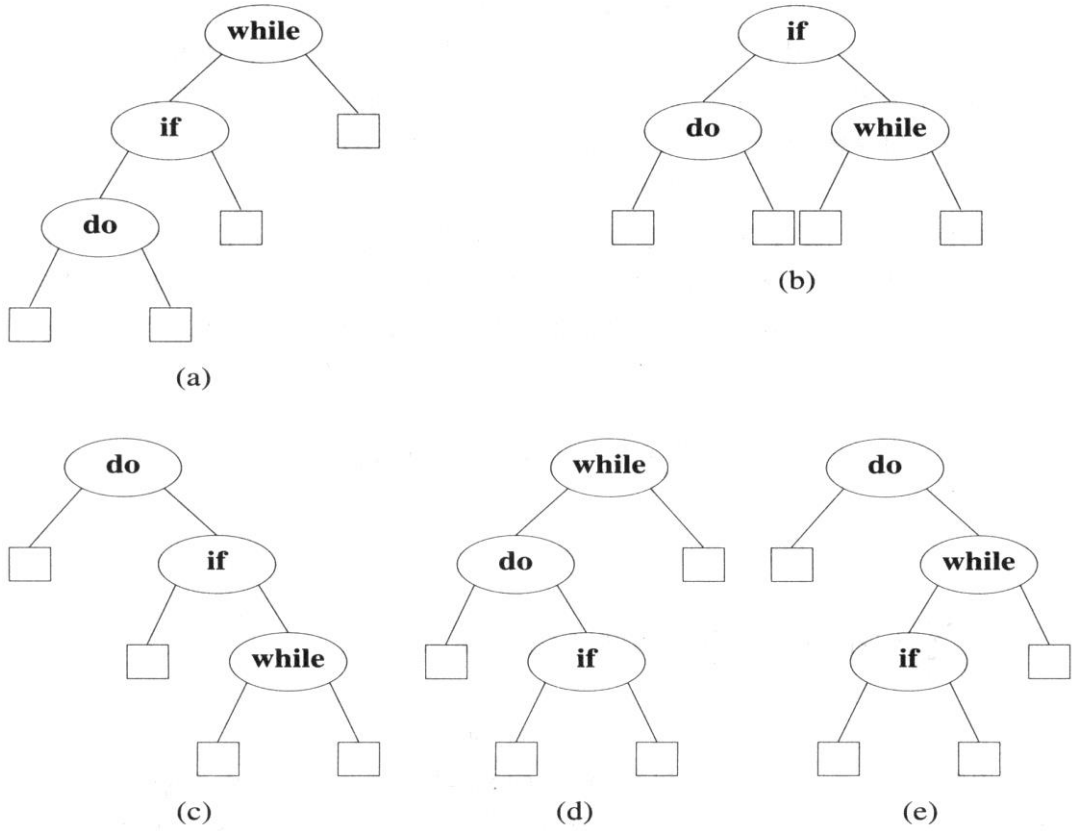


그림 10.4 : 세 개의 식별자를 가진 이진 탐색 트리

- 경우 1: 모든 노드에 같은 확률일 때,
  즉, $p_i = q_i = 1/7$
  cost(tree a) = 15/ 7, cost(tree b) = 13/ 7
  cost(tree c) = 15/ 7, cost(tree d) = 15/7
  cost(tree e) = 15/7
  트리 b가 최적 이진 탐색 트리이다.
- 경우 2: $p_1 = 0.5$, $p_2 = 0.1$, $p_3 = 0.05$,
  $q_0 = 0.15$, $q_1 = 0.1$, $q_2 = 0.05$, $q_3 = 0.05$
  cost(tree a) = 2.65, cost(tree b) = 1.9
  cost(tree c) = 1.5, cost(tree d) = 2.15
  cost(tree e) = 1.6
  트리 c가 최적 이진 탐색 트리이다.

최적 이진탐색트리의 생성

- 방법 1: 앞의 예제와 같이 모든 가능한 이진탐색트리들을 만든 후, 각각의 비용 C를 구하여 그 중 가장 작은 비용을 갖는 트리를 얻는다.

  ➔ n이 커질수록 이진탐색트리의 수가 급격히 증가하므로 이 방법은 비실용적이다.

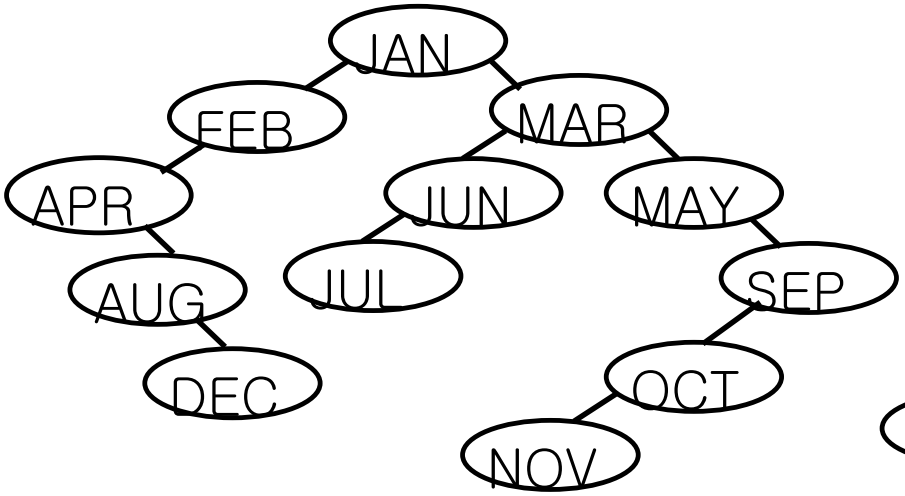
- 방법 2: $a_1 < a_2 < \ldots < a_n$ 은 n 개의 식별자에 대해 동적 프로그래밍 (dynamic programming) 기법을 적용함으로써 $O(n^3)$ 시간에 생성할 수 있다.

10.2 AVL 트리

- 삽입과 삭제가 일어나는 동적 테이블을 위해 이진탐색트리를 사용할 경우, 입력되는 데이터의 순서에 따라 생성되는 이진탐색트리의 모양이 달라지므로 평균비교횟수가 달라진다 (다음 슬라이드 참조).
- 이진탐색트리가 완전이진트리가 되었을 때, 평균 탐색시간과 최대 탐색시간을 최소화할 수 있으나 새로운 노드가 삽입될 때 재구성에 많은 시간이 걸린다.

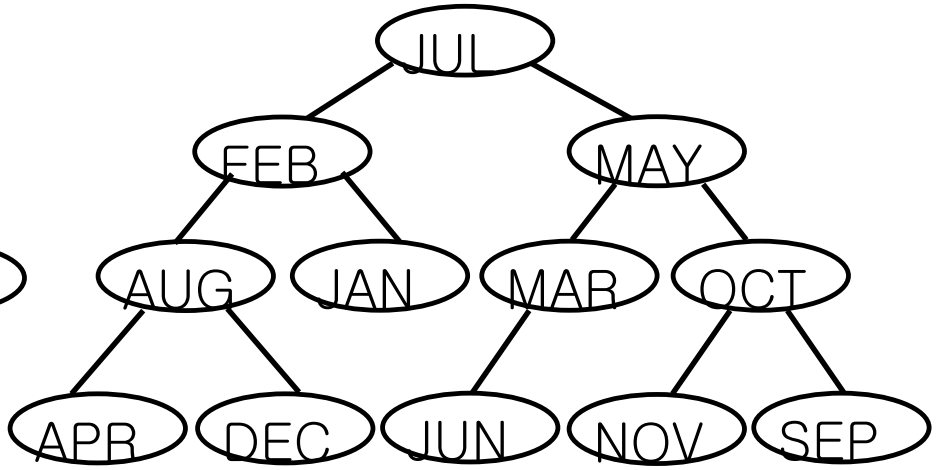
  → 삽입, 삭제, 탐색이 모두 O(log n) 시간에 이루어지는 높이 균형 이진트리(일명 AVL 트리)
- Adelson-Velskii와 Landis가 제안

- 어떤 원소를 찾기 위한 최대 비교 횟수



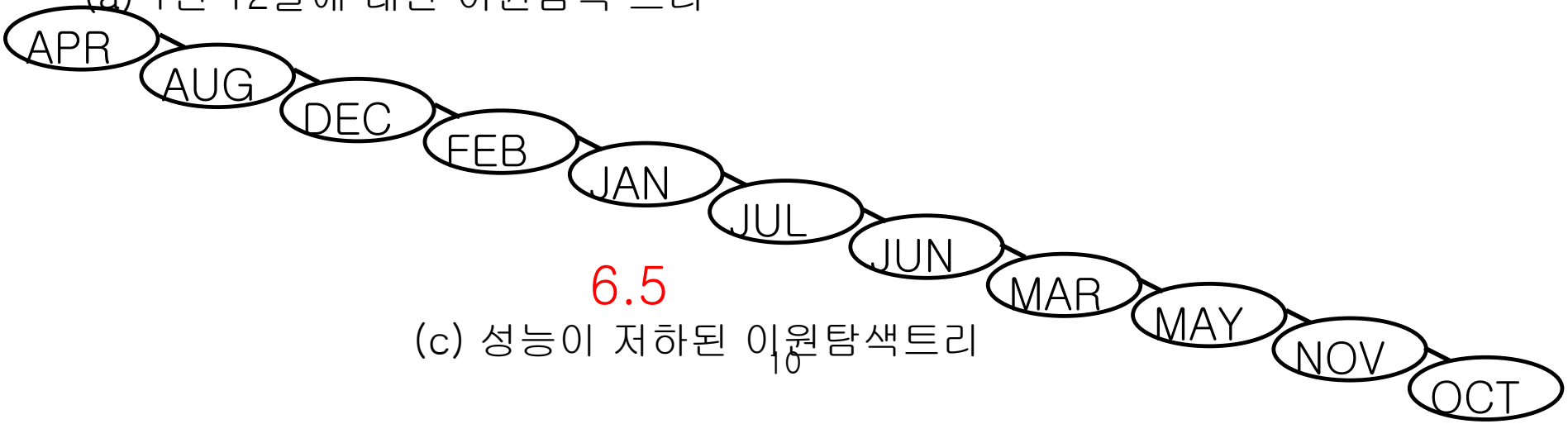
3.5

(a) 1년 12달에 대한 이원탐색 트리



3.1

(b) 1년 12달에 대한 균형트리



6.5

(c) 성능이 저하된 이원탐색트리

AVL 트리의 정의

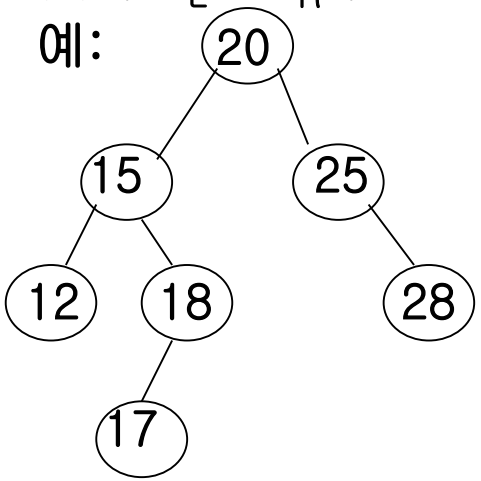
- 정의:

  공백트리는 높이 균형을 이룬다. 트리 T가 왼쪽과 오른쪽 서브트리인 T_L과 T_R 을 가진 이진트리라 할때, 다음과 같은 높이 균형 조건을 만족하면 이진탐색트리를 AVL 트리라 한다.

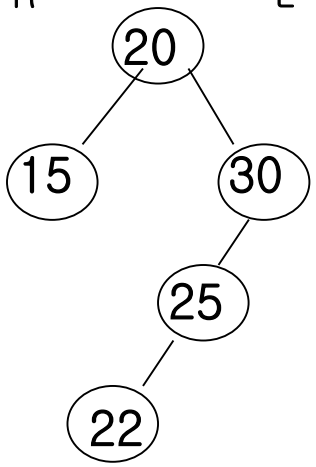
  (i) T_L과 T_R 이 높이 균형을 이루며,

  (ii) $| h_L - h_R | \leq 1$ (h_L 과 h_R 은 각각 T_L과 T_R 의 높이)

- 예:



AVL 트리



AVL 트리 아님

AVL 트리의 탐색, 삽입, 삭제

- 정의:

  이진트리에서 노드 T의 균형 인수 BF(T)는 $h_L - h_R$ 로 정의한다.

  T가 AVL 트리의 노드일 때, BF(T)는 -1, 0, 또는 1이다.
- 탐색: AVL 트리는 일종의 이진탐색트리이므로 이진탐색트리의 탐색과 같다
- 삽입, 삭제: 이진탐색트리의 삽입, 삭제 방법과 같으며, 삽입과 삭제 후에 불균형이 발생하면, 4가지 형태 중 하나의 방식으로 회전을 실시하여 균형을 맞춰준다.
- 네가지 방식(삽입):

  LL: 새로운 노드 Y는 A의 왼쪽 서브트리의 왼쪽 서브트리에 삽입된다.

  RR: “ 오른쪽 “ 오른쪽 “

  LR: 새로운 노드 Y는 A의 왼쪽 서브트리의 오른쪽 서브트리에 삽입된다.

  RL: “ 오른쪽 “ 왼쪽 “

(Y: 새로운 노드, A: Y에 가장 근접한 조상으로 균형 인수가 2혹은 -2인 노드)