5.1 자료구조 개념

(1) 자료구조의 개요

1) 자료구조

① 일련의 자료들을 조직하고 구조화하는 것

② 자료의 표현과 그것과 관련된 연산

2) 자료구조에 따라 저장공간의 효율성과 프로그램의 실행시간이 달라짐

3) 어떠한 자료구조에서도 필요한 모든 연산들을 처리하는 것이 가능함

(2) 자료구조의 분류


자료 구조
(Data Structure)
선형 구조
(Linear Structure)
비선형 구조
(Non-Linear Structure)
리스트 (List)
스택 (Stack)
큐 (Queue)
덱크 (Deque)
선형 리스트 (Linear List)
연결 리스트 (Linked List)
트리 (Tree)
그래프 (Graph)

5.2 선형 구조

(1) 리스트 (List)

1) 리스트의 종류

① 선형 리스트: "연속적" ⇨ Array(배열)

② 연결 리스트: "비연속적" ⇨ 포인터

2) 선형 리스트(Linear List)

① 연속적인 기억장소에 저장된 리스트

(순차리스트 또는 연접리스트(Dense List) 라고 함)

② 형태: 임의의 노드에 접근을 할 때는 인덱스(Index)를 사용하므로 포인터가 없음

1	심은하
2	고현정
3	전지현
4	김태희
5	문근영
6	이다해

③ 장점

- 간단한 자료구조
- 저장 효율이 뛰어남 (기록밀도: 1)
- 접근 속도(Access Time)가 빠름

④ 단점

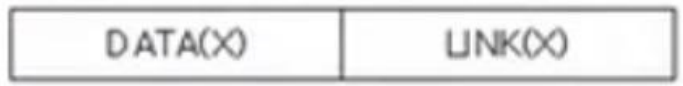
- 삽입과 삭제가 어려움
  (삽입 및 삭제 시 삽입하거나 삭제할 위치 이후의 모든 자료의 이동이 필요)

3) 연결리스트(linked list)

① 자료들을 임의의 기억공간에 기억시키고, 자료 항목의 순서에 따라 노드의 포인터 부분을 이용하여 서로 연결시킨 자료구조

② 형태

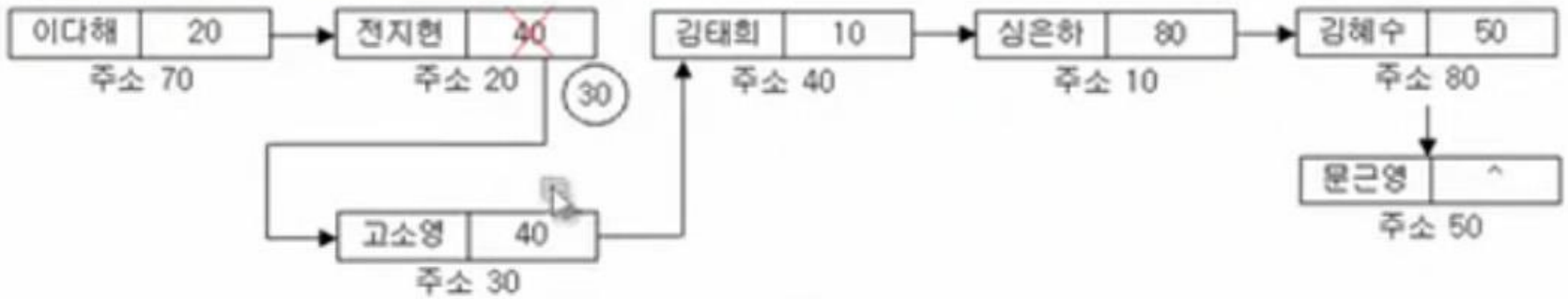
- 각 노드는 다음 노드를 가리키는 링크(Link , Pointer) 정보를 가짐
- 마지막 노드는 포인터 정보를 Null 값을 가짐



주소	데이터	링크
10	심은하	80
20	전지현	40
30		
40	김태희	10
50	문근영	^ (Null)
60		
70	이다해	20
80	김혜수	50

③ 장점

• 자료의 삽입 및 삭제가 용이



[30번지에 고소영 데이터를 추가]

• 비연속적 (한 리스트를 여러 개의 리스트로 분리하기 쉬움)

• 희소 행렬(행렬의 요소들 중에서 많은 부분이 0으로 되어 있는 행렬)을 연결리스트로 표현 시 기억장소 이용효율이 좋음

④ 단점

• Access Time이 느림

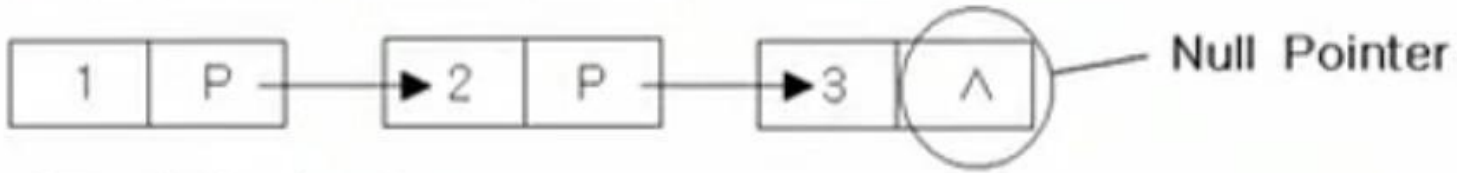
• 기억장소 이용 효율이 나쁨 (저장되지 않은 빈 공간 발생)

• 포인터를 위한 추가 공간이 필요

• 중간 노드 연결이 끊어지면 그 다음 노드를 찾기 힘듦

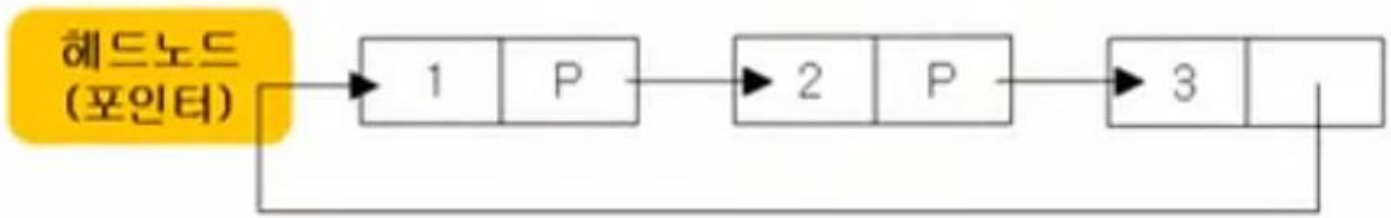
⑤ 종류

- 단순 연결 리스트 (단순 링크드 리스트)



- 이중 연결 리스트

- 단순 환형 연결 리스트

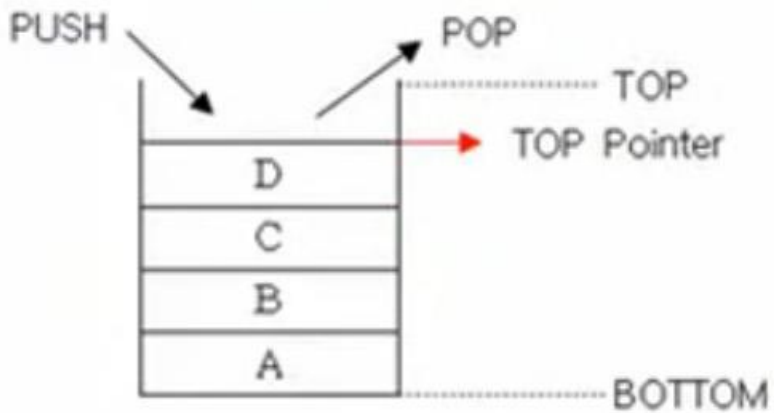


(2) 스택 (Stack)

1) 스택의 정의

① An ordered list in which insertions and deletions are made at one end called the top. (top이라고 하는 리스트의 한쪽 끝에서만 삽입과 삭제가 일어나는 자료구조)

② 자료의 후입선출(last-in-first-out) 방법



Top : 자료의 삽입과 삭제가 이루어지는 스택공간의 위치를 표시하는 포인터
Push : 스택에서 자료의 삽입
Pop : 스택에서 자료의 삭제

- 자료의 삽입: TOP = TOP + 1
- 자료의 삭제: TOP = TOP - 1
- Overflow 발생: 스택의 크기가 M 일 때 , TOP > M 이면 Overflow 발생

2) 스택의 용도

① 인터럽트의 처리

② 수식의 계산 (산술식 표현)

③ 서브루틴의 복귀번지 저장 (함수 호출의 순서 제어)

3) 순환적 프로그램을 처리하기 위한 요소

① 스택

② 복귀주소

③ 순환에서 탈출하는 조건

4) 삽입 알고리즘

```
Top = Top + 1
If(Top > M) Then
        Stack_overflow
Else
        Stack(top) ← data
```

5) 삭제 알고리즘

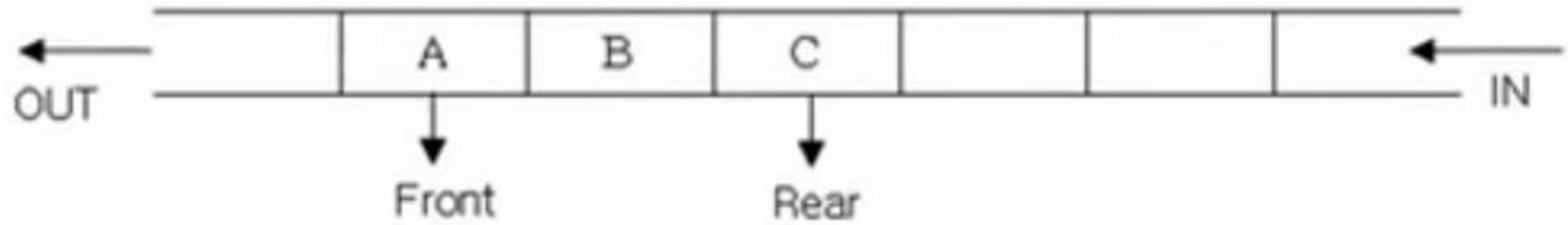
```
If(Top = 0) Then
        Stack_underflow
Else
        data ← Stack(top)
        top = top - 1
```

(3) 큐 (Queue)

1) 큐의 정의

An ordered list in which all insertions take place at one end and all deletions take place at the opposite end. (rear라고 하는 리스트의 한쪽 끝에서 삽입이 일어나고 front라 부르는 반대쪽 끝에서 삭제가 일어나는 자료구조(FIFO 구조))

2) 형태



3) 운영체제의 작업 스케줄링 등에 응용되는 것으로 가장 적합한 자료구조

(4) 데크 (Deque, Double Ended Queue)

1) 서로 다른 방향에서 입·출력이 가능한 구조 (삽입과 삭제가 양쪽 끝에서 일어남)

2) 입력이 한쪽에서만 발생하고 출력은 양쪽에서 일어날 수 있는 입력제한과 입력은 양쪽에서 일어나고 출력은 한곳에서만 이루어지는 출력제한이 있음

3) 스택과 큐를 복합한 형태

- 입력제한데크 : 입력이 한 쪽 끝으로 제한 (Scroll)
- 출력제한데크 : 출력이 한 쪽 끝으로 제한 (Shelf)

4) A generalization of a queue because it allows insertions and deletions at both end. (양 끝에서 삽입과 삭제가 가능하게 하도록 큐를 일반화한 것)

5.3 비선형 구조

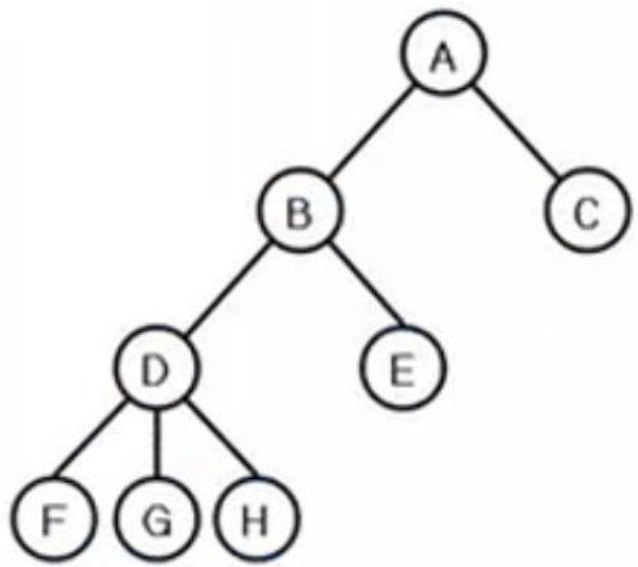
(1) 트리 (Tree)

1) 트리의 정의

① Linked List로 표현할 때 가장 효율적임

② 계층형 구조(hierarchical structure)를 나타내기 편리함

2) 트리 용어정리



① Node
- Tree 의 기본 구성요소
- A , B , C , D , E , F , G , H

② 근노드(Root Node)
- 가장 상위에 위치한 노드
- A

③ 레벨(Level)
- 근노드를 기준으로 특정 노드까지의 경로길이
- E 의 레벨은 3

④ 조상노드(Ancestors Node)
- 특정 노드에서 루트에 이르는 경로상의 모든 노드
- D 의 조상노드는 B , A

⑤ 자식노드(Son Node)
- 특정 노드에 연결된 다음 레벨의 노드
- B 의 자식노드는 D , E

⑦ 형제노드(Sibling)

- 같은 부모를 가진 노드
- F 의 형제노드는 G , H

⑧ 깊이(Depth, Height)

- 트리의 최대 레벨
- 위 트리의 깊이는 4

⑨ 차수(Degree)

- 특정 노드에 연결된 자식노드의 수
- D 의 차수는 3

⑩ 단말노드(Terminal Node , Leaf Node)

- 트리의 제일 마지막에 위치한 노드
- F , G , H , E , C

⑪ 트리의 차수

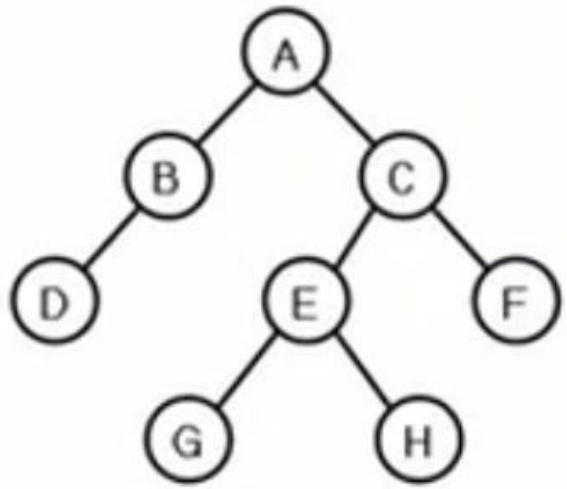
- 트리의 노드 중 가장 큰 차수
- 위 트리의 차수는 3 (D의 차수가 가장 크므로 D의 차수가 트리의 차수)

3) 트리의 운행법(Traversal)

① 운행법 정의

- 컴퓨터 기억 공간에 표현된 트리의 각 노드를 중복되지 않게 정해진 순서대로 전부 검색하여 트리의 정보에 관한 사항을 알고자 함
- 이진 트리의 운행법은 산술식의 표기법과 관련됨

② 운행법 종류



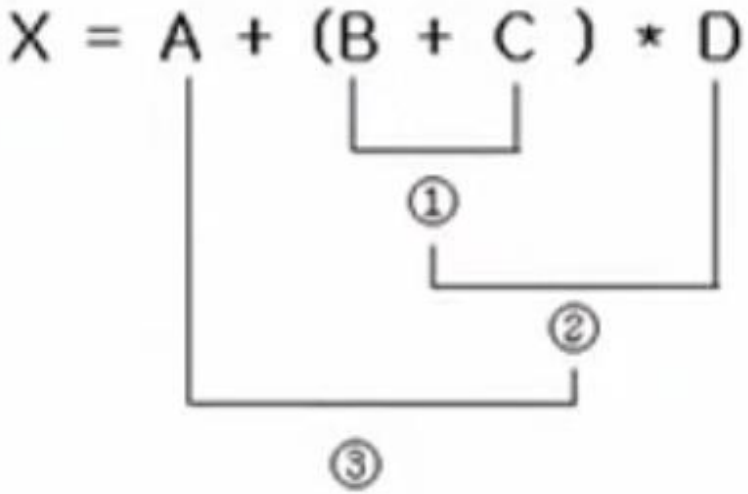
- preorder
  - root → left → right
  - A B D C E G H F

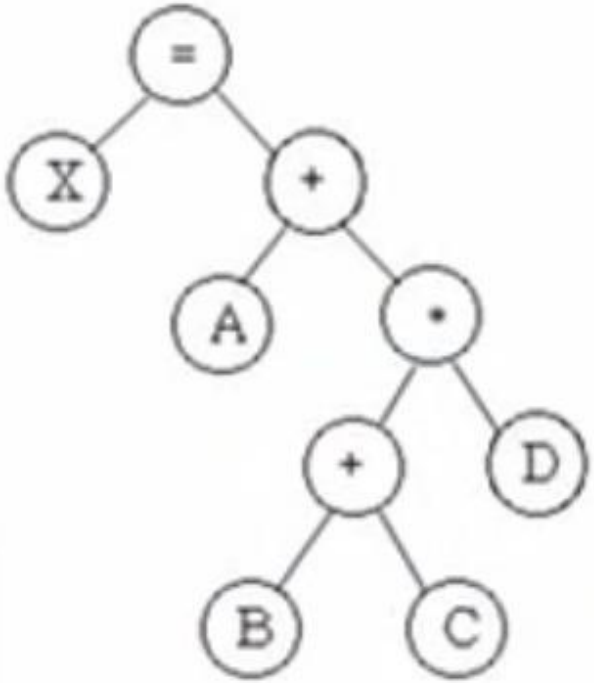
- inorder
  - left → root → right
  - D B A G E H C F
- postorder
  - left → right → root
  - D B G H E F C A

③ 수식의 표기법

- 산술식을 계산하기 위해 기억공간에 기억시키는 방법으로 이진트리를 사용함
- 표기법의 종류
  - 중위 표기법 (Infix notation)
    연산자가 피연산자 사이에 있는 표기법
  - 후위 표기법 (Postfix notation, reverse Polish notation)
    피연산자 뒤에 연산자가 표기되는 표기법
  - 전위 표기법 (Prefix notation)

  피연산자들 앞에 연산자를 표시하는 표기법





▷ 중위 표기법

A + B + C * D

▷ 후위 표기법

A B C + D * +

▷ 전위 표기법

+ A * + B C D

④ 표기법 변환

- 현재의 표기법에서 우선순위가 가장 높은 두 개의 피연산자와 한 개의 연산자를 바꾸고자 하는 표기법으로 바꿈. 바뀐 연산은 하나의 피연산자로 생각하고 위의 과정 반복
- 중위 표기법 → 후위 표기법

  예) A/B-(C*D)/E

  AB/ CD*

  CD*E/

  AB/CD*E/-

- 중위 표기법 → 전위 표기법

  예) A/B-(C*D)/E

  /AB *CD

  /*CDE

  -/AB/*CDE

※ 후위나 전위 표기법에서 중위 표기법으로 바꾸려면 산술식의 앞쪽에서부터 연속된 2개의 피연산자와 하나의 연산자를 찾아 중위 표기법으로 바꾸는 과정을 반복함

예) -/AB/*CDE

A/B C*D

C*D/E

A/B-C*D/E

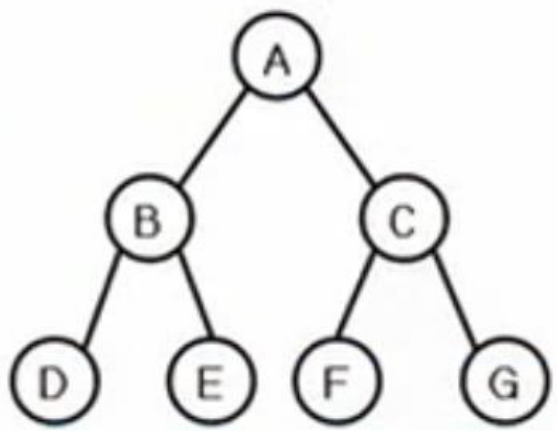
4) 트리의 종류

① 이진 트리

- 정의
  - Degree가 2이하로 구성된 트리
- 특성
  - 깊이가 k인 이진 트리의 최대노드의 수 : 2^k-1
  - 이진 트리의 레벨 i 에서 최대노드의 수 : $2^{(i-1)}$
  - $n_0 = n_2 + 1$

    (n_0: 이진 트리의 터미널노드, n_2: 차수가 2인 노드 수)

예) 레벨3에서 최대노드 수

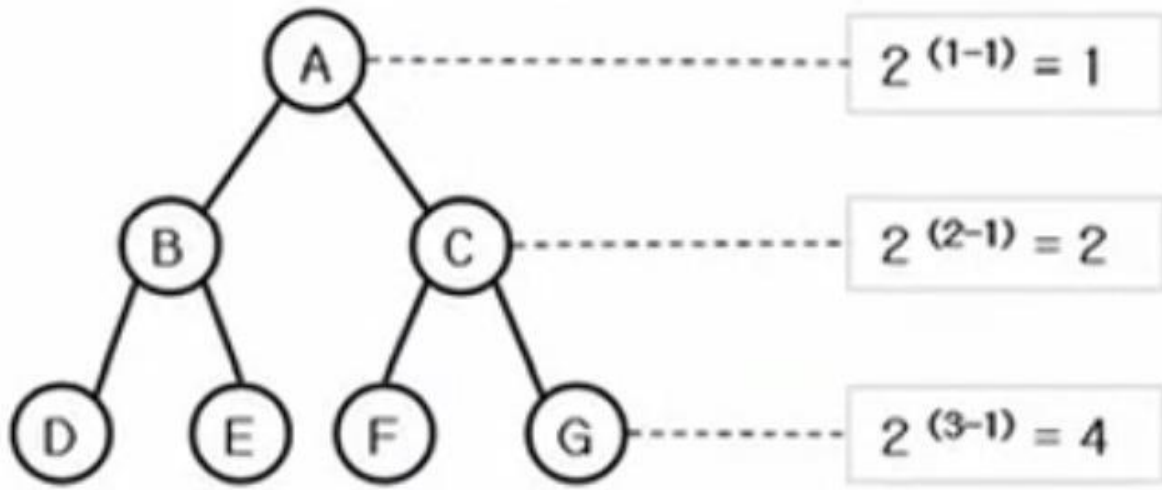


$2^{(i-1)} \Rightarrow 2^{(3-1)} = 4$

$n_0 = n_2 + 1$

$(4 = 3 + 1)$

- 종류
  - 정이진 트리(FBT, Full Binary Tree)
    * depth(k), 전체노드 수 : 2^k-1
    * 레벨마다 $2^{(i-1)}$개 노드 가득

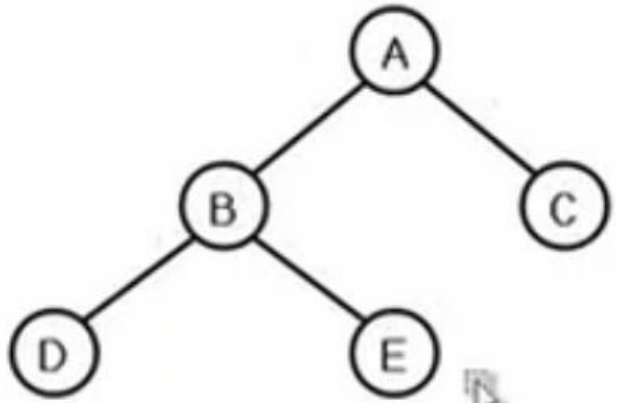


depth(3)

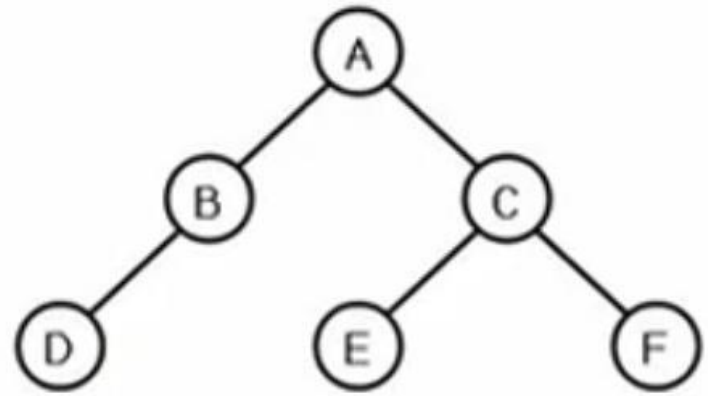
$2^3 - 1 = 7$

  - 전이진 트리(CBT, Complete Binary Tree)
    * 정이진 트리의 각 노드에 붙인 1~n의 일련번호와 1:1로 대응되는 tree
    * 중간에 비워 있으면 안 됨

* 정이진 트리의 각 노드에 붙인 1~n의 일련번호와 1:1로 대응되는 tree
* 중간에 비워 있으면 안 됨

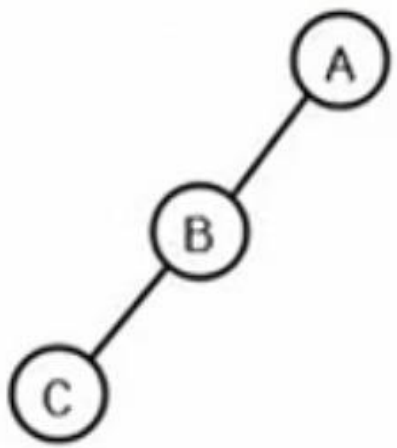


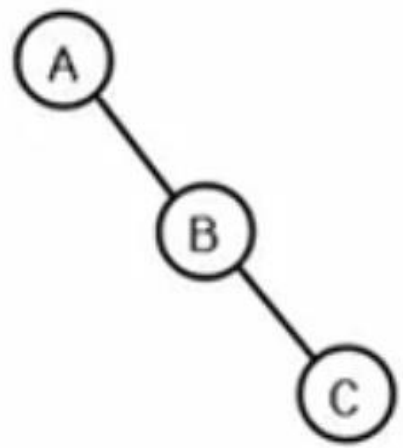
<전이진 트리>



<전이진 트리 아님>

- 사향 이진 트리(SBT, Skewed Binary Tree)
  ⇨ 이진트리가 한쪽방향





② 스레드(thread) 이진 트리

- 비순환적인 이진트리 운행 알고리즘에서 스택을 사용하지 않고 낭비되는 널(NULL) 연결 필드를 활용하여 트리 운행에 필요한 다른 노드의 포인터로 사용하도록 고안된 이진트리

③ AVL 트리

- 각 노드의 왼쪽 서브트리의 높이와 오른쪽 서브트리의 높이 차이가 1이하인 이진 탐색 트리
- AVL 트리의 장점: 탐색시간이 빠름

④ 균형 트리

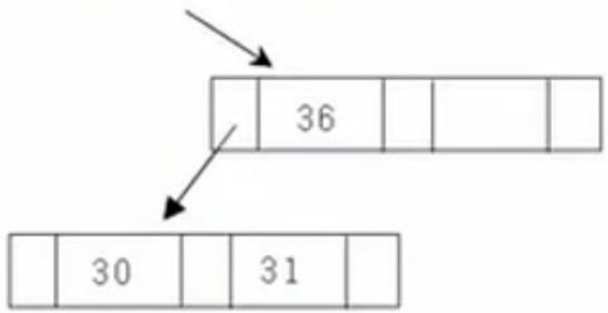
- 트리의 단말노드의 레벨이 모두 같게 만든 트리
- 실제 레코드까지의 탐색 길이가 동일하게 색인부를 완전 균형트리로 구성

⑤ B-트리

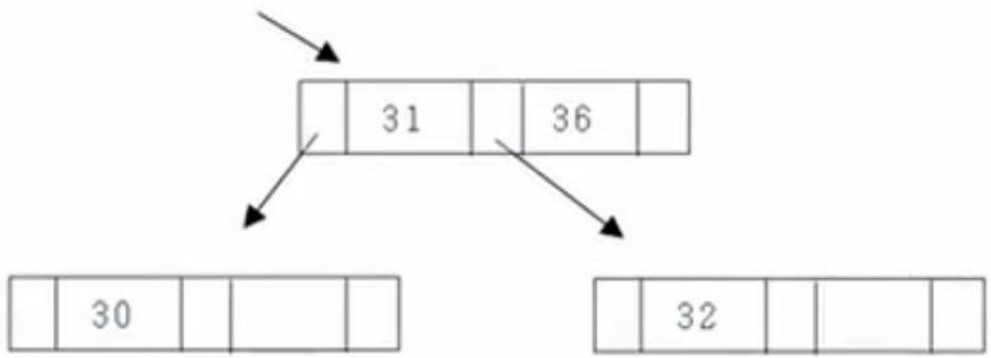
- 한 노드 안에 있는 키 값은 오름차순을 유지함
- 루트와 리프(leaf)를 제외한 모든 노드는 최소 (m/2)개, 최대 m개의 서브트리를 가짐

- 루트(root) 노드는 리프가 아닌 이상 적어도 두 개의 서브트리를 가짐
- 탐색, 추가, 삭제는 루트로부터 시작함
- 삽입과 삭제를 하여도 데이터 구조의 균형을 유지해야 함
- 순차 탐색은 각 노드를 중위 순회함으로써 좋은 성능을 발휘하지 못함
- B-트리는 인덱스 파일에서 인덱스를 구성하는 방법 중의 하나임

예) 다음과 같은 B-트리에서 노드 32가 추가된 후 다시 노드 35가 추가되었을 때의 결과는?



※ 32추가 시

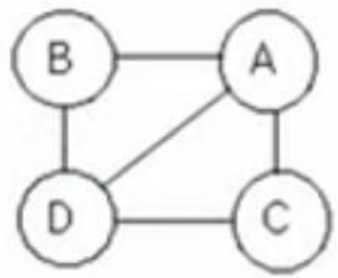


6) 그래프(Graph)

① 그래프의 정의

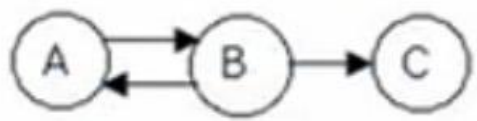
- 각각의 단위 정보를 링크로 연결하여 구조화시킨 자료 구조
- 정점(vertex) : 노드들의 집합
- 간선(edge) : 정점들 사이의 상호 연결의 집합, 임의의 점들의 쌍을 연결

② 인접행렬 (Adjacency Matrix)



	A	B	C	D
A	0	1	1	1
B	1	0	0	1
C	1	0	0	1
D	1	1	1	0

③ 인접 리스트



	A	B	C
A	0	1	0
B	1	0	1
C	0	0	0