한라대학교
정보통신소프트웨어
교수 이경호

알고리즘 개요

(자료구조와 알고리즘)

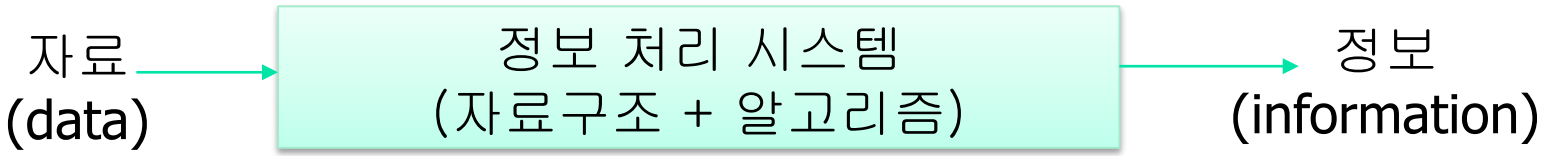
이 단원에서 배울 내용

- 정보 처리를 위한 프로그램과 자료구조, 알고리즘의 이해

컴퓨터 프로그램을 작성하는 이유

- 주어진 문제를 해결하여 어떤 효과를 얻고자 함.
  - 프로그램 : 문제상태, 해결상태를 정의하고, 효과(이익)를 고려하여, 효율성을 고려한 해결방법 명령어들의 단계적 나열이 프로그램이다.
    - 문제 : 해결이 되면 이익이 발생하는 '해답을 요구하는 물음'.
    - 해결 : 주어진 상태에서 원하는 상태로 되게 하는 것
    - 해결 방법 : 해답을 만들어 내는 과정
    - 이익 : 물질적으로나 정신적으로 보탬이 되는 것.
    - 효율성 : (결과 이익 / 해답을 만들어 내는 과정의 들인 노력) 비율이 높은 특성

- 문제 해결 프로그램을 전쟁에 비유하면 문제 상태를 파악하고 병력과 보급을 잘 배치한 후 효율적인 보급과 효율적인 전략으로 해결.
  - 효율적인 보급 : 자료구조
  - 효율적인 전략 : 알고리즘



프로그램에 의한 문제의 해결

[Q] 기둥을 오르내리는 달팽이 로봇이 있다.
낮에는 3m 올라가고, 밤에는 2m 내려온다. 30m 기둥 꼭대기에 올라가는데 몇 일 걸리나?

- 문제 이해 및 문제 세계 컴퓨터 묘사
  - 문제의 잘 보고 애매모호성 제거. //아침 or 저녁 출발?
  - 문제에서 프로그램에 묘사(표현)할 내용 파악. //등장 요소 및 상태
  - 표현해야 할 내용으로 초기 상태와 종료 상태를 결정.

    표현 사항? 달팽이 로봇 / 로봇의 기둥 상 위치 / 기둥 상태 / 출발 상태 / 목표 상태 / 기간
- 프로그램에 의한 문제 해결
  - 초기 상태에서 종료 상태로 가게 하기 위하여 변경해야 하는 자료는. //달팽이 위치
  - 자료 관리(삽입, 삭제, 조회, 변경)하며, <= 자료구조

    달팽이 로봇의 기둥 상 위치 변경, 기간
  - 결과를 생성할 수 있도록 명령어 절차를 구성. <= 알고리즘

```
while( true ) {
    if (현 상태 == 낮) {
        달팽이 기둥상 위치 =+ 3;    현 상태 = 밤;
        위치 변화 후 목표 상태 도달(프로그램 종료) 또는 예외 처리 상태 확인;
    }
    if (현 상태 == 밤) {
        달팽이 기둥상 위치 =- 2;   현 상태 = 낮;
        위치 변화 후 목표 상태 도달(프로그램 종료) 또는 예외 처리 상태 확인;
    }
}
```

자료구조와 알고리즘의 이해

- 웹상에서 경매로 물건을 판매하는 프로그램을 구성하라.

- 조건
  - 판매하는 물건은 한 번에 최대 5개까지 있을 수 있다.(단 1개 물품 단위로 경매하고 높은 가격 신청 순으로 팔린다.)
  - 경매는 회원으로 등록된 회원이면 누구나 참여할 수 있다. //회원으로 전제
  - 한 번에 한 개를 신청할 수 있어 한 사람이 여러 개를 사고 싶으면 여러 번 신규 입찰 신청하면 된다.
  - 낙찰 결정은 주어진 시간까지 써낸 가격 중 높은 가격 신청 순으로 설정된 물건 수만큼 팔린다.
  - 입찰 종료 시간 24시간 전이면 입찰 신청을 취소할 수 있다.
  - 입찰에 참여한 회원은 입찰 기록 가격을 변경할 수 있다.

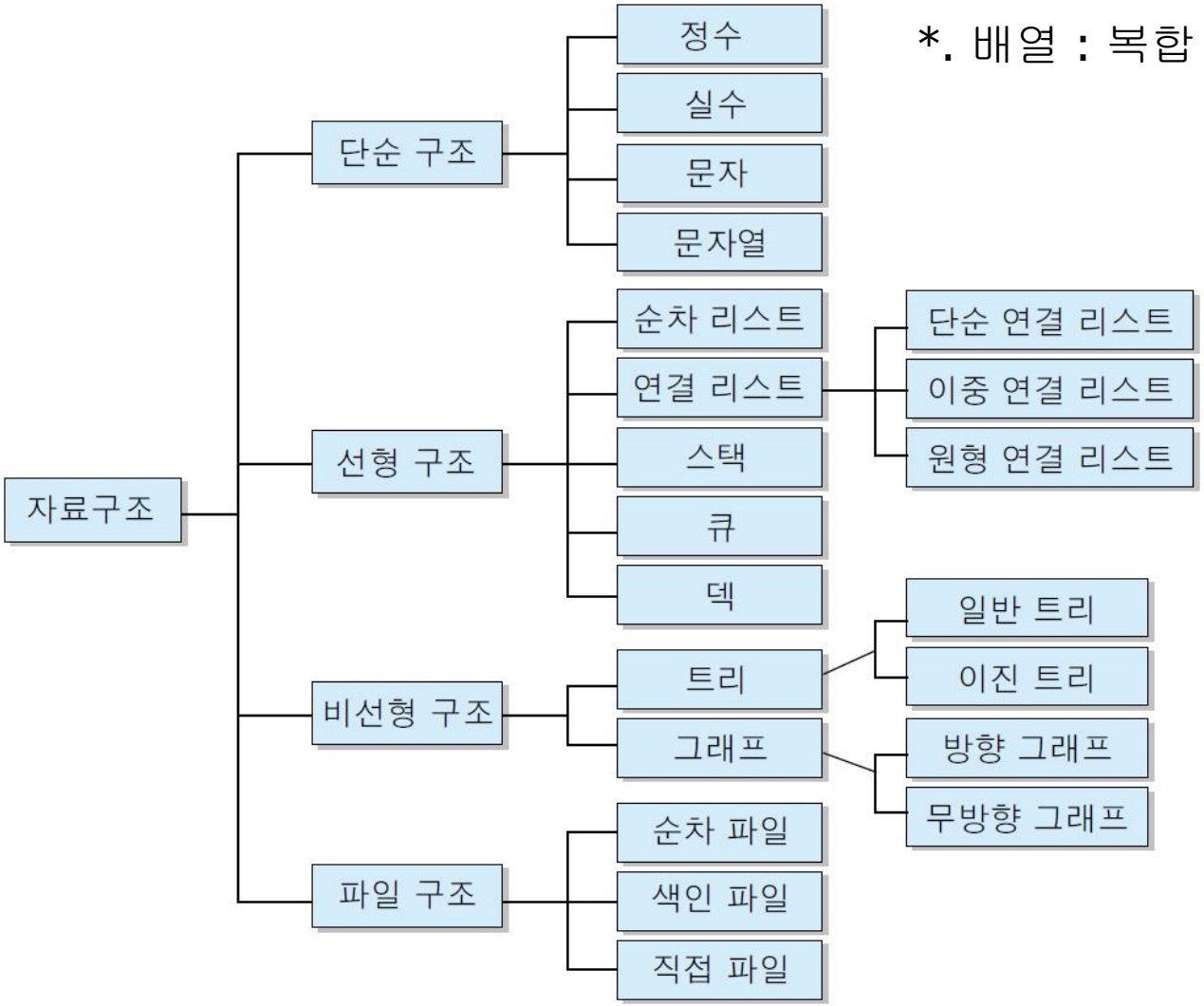
  - 문제의 잘 보고 애매모호성 제거.
  - 문제에서 프로그램에 묘사(표현)할 내용 파악. //등장 요소 및 상태
  - 표현해야 할 내용으로 초기 상태와 종료 상태를 결정.

- 문제 해결 요소들의 표현
  1. 물품 표현 : 변수를 통한 물품명 기록 string ProductName;
  2. 물품 개수 : 변수를 통한 개수 기록 int ProductNum;
  3. 입찰 종료 시간 : 변수를 통한 기록 date EndTime; (변수 종류 경계)
  4. 입찰 기록 : (입찰자, 입찰 금액) 형태로 여러 개를 기록해야 함. 구조체 배열을 이용한 기록. 리스트이나 입찰 금액에 의한 내림 차순 정렬
     - 삽입 – (입찰자, 입찰 금액)을 제공하면 삽입된 입찰 기록 형성. 내림차순 정렬
     - 삭제 – 위치를 제공하면 삭제된 입찰 기록 형성
     - 조회 – 회원 번호(입찰자)를 제공하면 위치와 값 반환
     - 변경 – 위치와 (입찰자, 입찰 금액)으로 새로운 값 제공하면 값이 변경된 내림차순 입찰 기록 형성.
       - 위 변수들보다 삽입, 삭제, 변경, 조회가 더 복잡함. => 자료구조 학문이 발생.
       - 리스트는 단순히 작업에 필요한 값을 주면 작업을 수행할 뿐, 높은 값이 앞으로 오게 하는 작업은 하지 않음. //리스트는 자료구조의 요소
       - 삽입, 삭제, 변경 등의 작업에서 위치를 결정하는 명령의 구성 필요 => 알고리즘.

자료구조 복습

자료구조

*. 배열 : 복합 구조


자료구조
단순 구조
정수
실수
문자
문자열
선형 구조
순차 리스트
연결 리스트
단순 연결 리스트
이중 연결 리스트
원형 연결 리스트
스택
큐
덱
비선형 구조
트리
일반 트리
이진 트리
그래프
방향 그래프
무방향 그래프
파일 구조
순차 파일
색인 파일
직접 파일

자료 관리 전제

- 자료 관리 시 수행하는 연산의 종류
  - 삽입, 삭제, 조회, 변경

- 자료 관리 전제
  - 관리자(프로그램)는
  - 관리하는 자료에 연산(삽입, 삭제, 조회, 변경)을 하기 위해서는
  - 각 자료의 위치와 자료의 범위(시작과 끝 그리고 자료 간 다음 자료의 위치와 전 자료의 위치)는 알고 있어야 한다.(프로그램 구성 시 반드시 반영되어야 한다.)

*. 컴퓨터 상의 모든 메모리는 입력한 적이 없어도 항상 어떤 값을 표현하고 있음을 기억하라.

자료구조의 종류

- 자료구조 종류 : 자료의 나열 시 선형 유무에 의한 분류
  - 선형 자료 구조(자료간 관계가 단순하며 자료 관리가 쉽다) / 리스트, 스택, 큐
  - 비선형 자료구조(자료간 관계 및 관리가 조금 더 복잡하다) / 트리, 그래프

- 각 자료 구조의 특성
  - 일렬로 관리하되, 임의 위치에서 자료의 삽입, 삭제와 조회, 변경을 하고 싶다. => 리스트
  - 일렬로 관리하되, '삽입 ,삭제'에 관심이 있고, 반드시 먼저 들어온 자료가 먼저 나가게 하고 싶다. => 큐
  - 일렬로 관리하되, '삽입 ,삭제'에 관심이 있고, 반드시 나중(최근)에 들어온 자료가 먼저 나가게 하고 싶다. => 스택
  - 자료들 간의 관계성을 표현하고 싶다 => 그래프.
    - 자료는 노드(정점)로 표현하고 관계는 아크(링크, 연결선)로 표현한다. 주로 탐색 문제 해결에 이용.
  - 자료들 간에 관계성을 표현하되 근원(root)으로부터 레벨이 있게 구성하고 싶다. => 트리.

- 리스트 : 임의로 삽입, 삭제, 조회 변경하고자 함.(선형적,
  - 배열로 구성. 필요 시 구조체 배열로 구성 //필요 시 연결리스트로 구성
  - 연산 : 삽입, 삭제, 조회, 변경. 연산 후 자료들은 앞부터 채워져 있음.
    *. 최근 언어에서는 언어에 리스트 자료구조가 구성되어 있음.

- 스택 : 삽입 삭제에 관심이 있으며, 나중 들어온 자료가 먼저 나가게 하고 싶다.
  - 배열로 구성. 필요 시 구조체 배열로 구성 //필요 시 연결리스트로 구성
  - 연산 : 삽입, 삭제. //조회, 변경은 관심이 없음. 필요 시 구성.

- 큐 : 삽입 삭제에 관심이 있으며, 먼저 들어온 자료가 먼저 나가게 하고 싶다.
  - 배열로 구성. 필요 시 구조체 배열로 구성 //필요 시 연결리스트로 구성
  - 연산 : 삽입, 삭제. //조회, 변경은 관심이 없음. 필요 시 구성.

- 그래프 : 자료 간 관계에 의한 상태 공간 형성.
  - 배열로 구성 //필요 시 연결리스트로 구성
  - 삽입, 삭제, 변경 등의 연산을 할 수도 있지만 주로 조회(탐색)에 깊은 관심.
  -
- 트리 : 자료간 레벨 관계에 의한 상태 공간 형성.
  - 배열로 구성 //필요 시 연결리스트로 구성
  - 삽입, 삭제, 조회, 변경

순차리스트

A[0]	A[1]	A[2]	A[3]		...		A[9]	A[10]	A[11]
0	10	20	30		...		90		

(a) 12 개의 원소를 갖는 배열

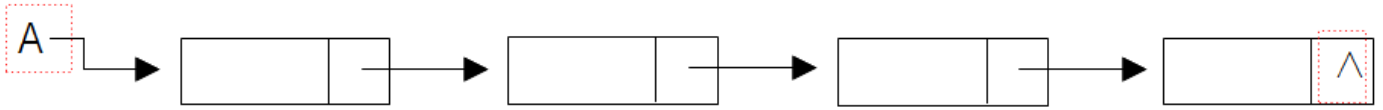
A[0]	A[1]	A[2]	A[3]		...		A[9]	A[10]	A[11]
0	10	15	20		...		80	90	

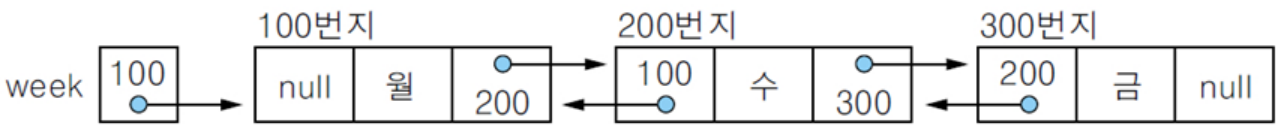
(b) 15가 삽입된 후의 형태

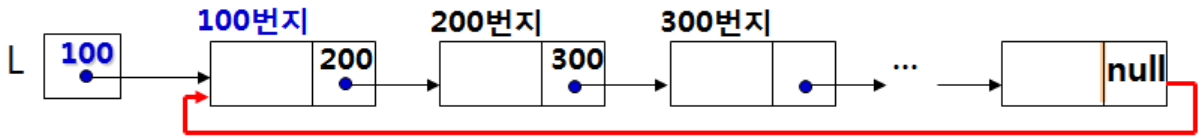
A[0]	A[1]	A[2]	A[3]		...		A[9]	A[10]	A[11]
0	10	20	30		...		90		

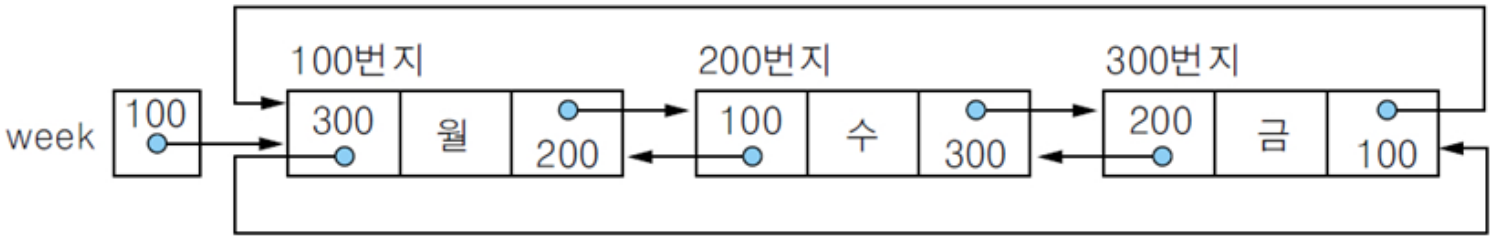
(c) 이어 15가 삭제된 후의 형태

연결리스트

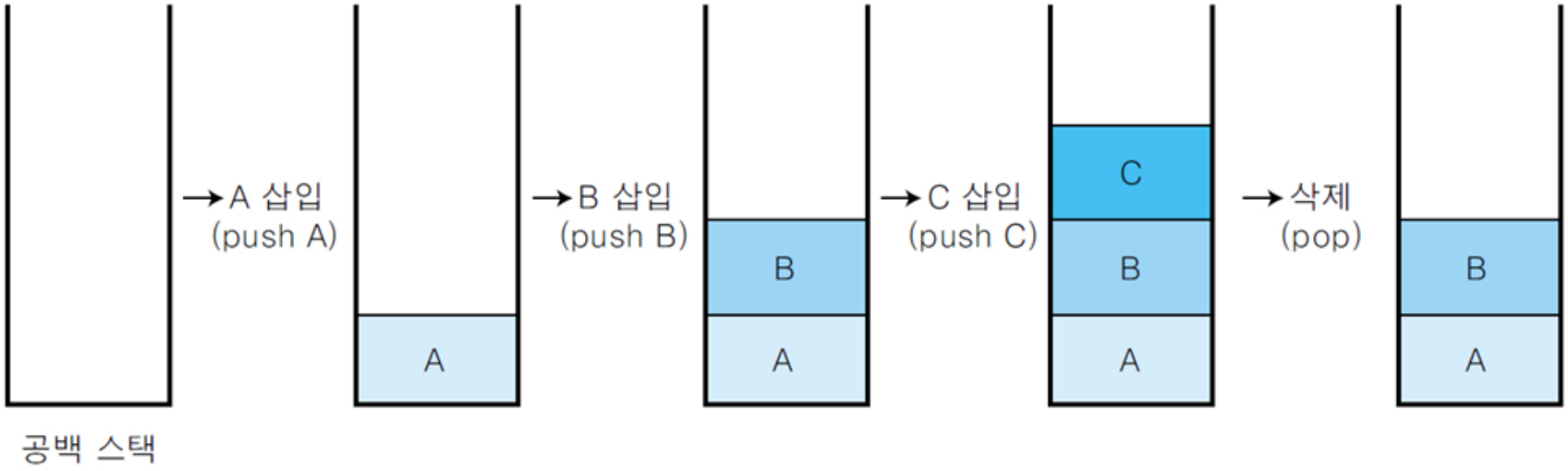

A
^


week
100
100번지
null
월
200
200번지
100
수
300
300번지
200
금
null


L
100
100번지
200
200번지
300
300번지
...
null

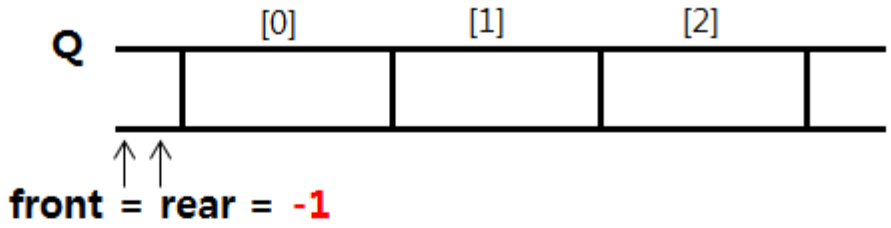

week
100
100번지
300
월
200
200번지
100
수
300
300번지
200
금
100

스택

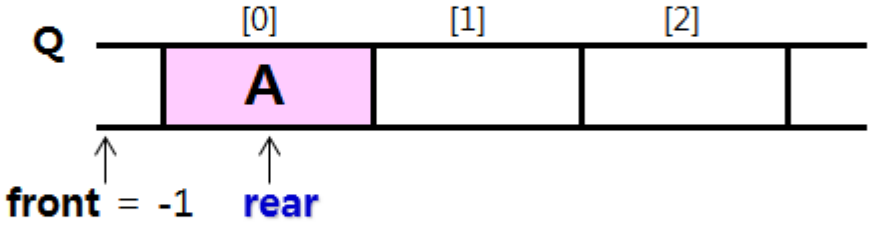


큐

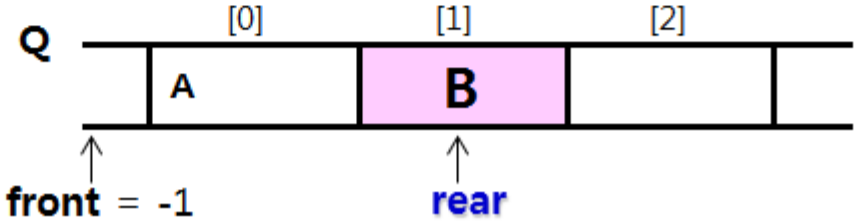
① 공백 큐 생성 : createQueue();



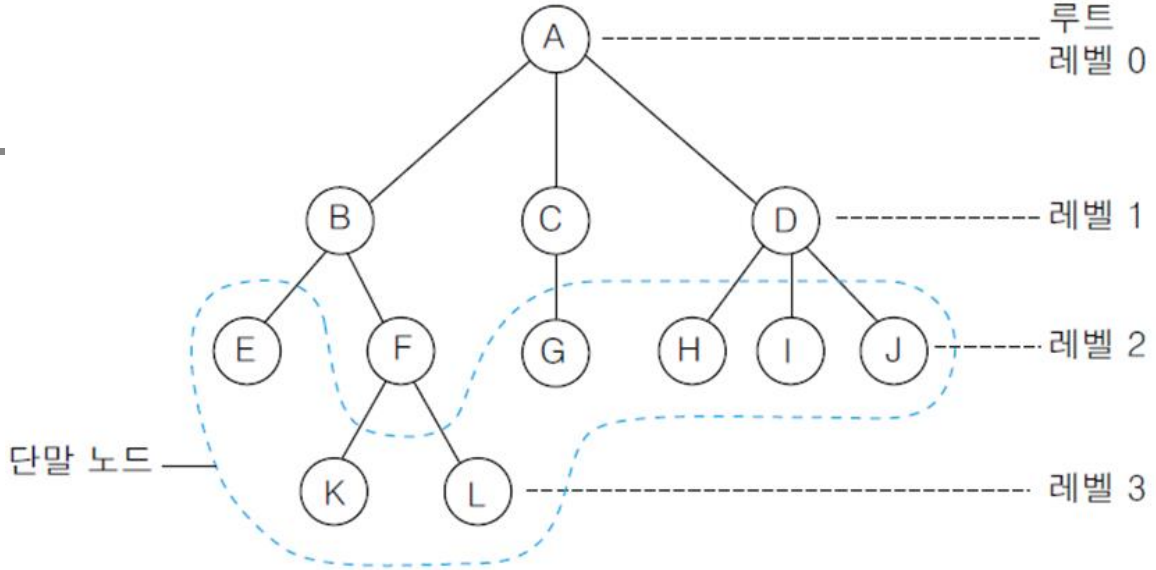
② 원소 A 삽입 : enQueue(Q, A);

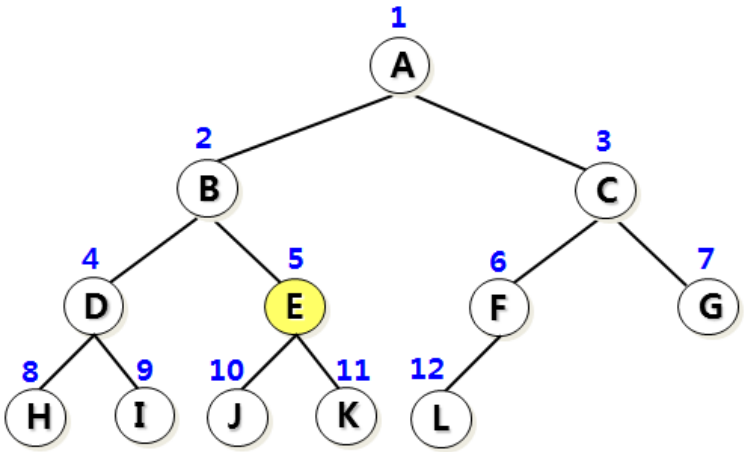


③ 원소 B 삽입 : enQueue(Q, B);



트리

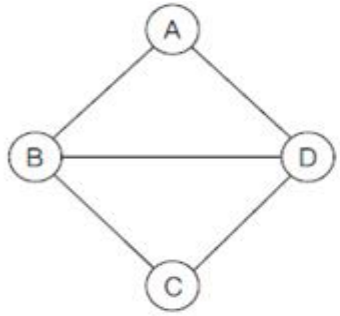




[0]	
[1]	A
[2]	B
[3]	C
[4]	D
[5]	E
[6]	F
[7]	G
[8]	H
[9]	I
[10]	J
[11]	K
[12]	L

그래프

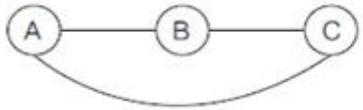
G1



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

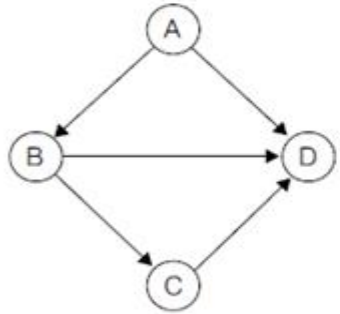
1+0+1+1=3 ⇨ 정점 B의 차수

G2



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

G3

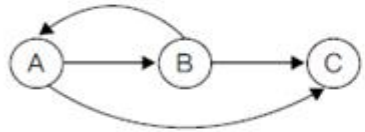


	A	B	C	D
A	0	1	0	1
B	0	0	1	1
C	0	0	0	1
D	0	0	0	0

0+0+1+1=2 ⇨ 정점 B의 진출 차수

1+0+0+0=1 ⇨ 정점 B의 진입 차수

G4



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