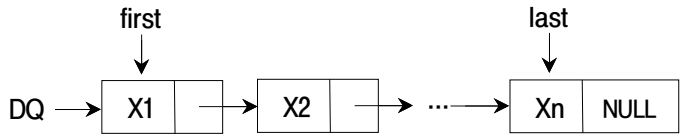
자료구조론	국가 전산 7급	2014년 7월 26일

♤ 합격선/최종합격인원(74.28점/32명) - 양성(71.42점) ♤

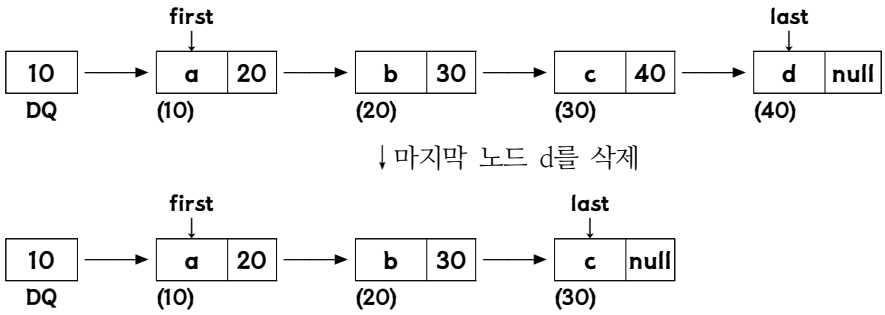
1. 데크(deque: double-ended queue)는 삽입과 삭제가 양끝에서 임의로 수행되는 자료구조이다. 다음 그림과 같이 단순연결리스트(singly linked list)로 데크를 구현한다고 할 때 O(1) 시간 내에 수행할 수 없는 연산은? (단, first와 last는 각각 데크의 첫 번째 원소와 마지막 원소를 가리키며, 연산이 수행된 후에도 데크의 원형이 유지되어야 한다) [2014년 국가 7급]



① insertFirst 연산 : 데크의 첫 번째 원소로 삽입
② insertLast 연산 : 데크의 마지막 원소로 삽입
③ deleteFirst 연산 : 데크의 첫 번째 원소를 삭제
④ deleteLast 연산 : 데크의 마지막 원소를 삭제

♤ 데크(deque) 삽입/삭제

① insertFirst 연산 : 데크의 첫 번째 원소로 삽입 → O(1)
② insertLast 연산 : 데크의 마지막 원소로 삽입 → O(1)
③ deleteFirst 연산 : 데크의 첫 번째 원소를 삭제 → O(1)
④ deleteLast 연산 : 데크의 마지막 원소를 삭제 → O(n)
→ 마지막 노드 삭제는 마지막 노드의 선행 노드를 찾아야 한다.
→ 따라서, 리스트 처음부터 끝 부분까지 추적해야 하며, 연산시간은 O(n)이다.



정답 : ④

2. 다음은 x^n을 구하는 알고리즘이다. 이 알고리즘의 시간복잡도는? (단, n>0이며, even(m)은 m이 짝수일 때 참을 반환하고 그렇지 않을 때 거짓을 반환하는 함수이다) [2014년 국가 7급]

```
-------------------------------------------------
int PowersRec(int x, int n)
{
    int Pow;
    if (n == 1)
        Pow = x;
    else {
        if (even(n))
            Pow = PowersRec(x*x, n/2);
        else
            Pow = x * PowersRec(x*x, (n-1)/2);
    }
    return (Pow);
}
-------------------------------------------------
```

① $\Theta(1)$ ② $\Theta(\log\ n)$
③ $\Theta(n)$ ④ $\Theta(n\log\ n)$

♤ 거듭제곱(x^n) 구하는 알고리즘

//반복 알고리즘	//재귀 알고리즘

double iter_pow(double x, int n)
{
  int i;
  double r = 1.0;
  for(i=0; i<n; i++)
  r = r * x;
  return r;
}
	double pow(double x, int n)
	{
	  if(n==0) return 1;
	  else if(n%2==0) //n이 짝수일 때
	  return pow(x*x, n/2);
	  else //n이 홀수일 때
	  return x*pow(x*x, (n-1)/2);
	}

• 연산시간 : O(n)	• 연산시간 : O(log n)

• 반복 알고리즘은 for를 사용하여, x를 n번 곱한다.
• 해서, 연산시간은 O(n)이다.

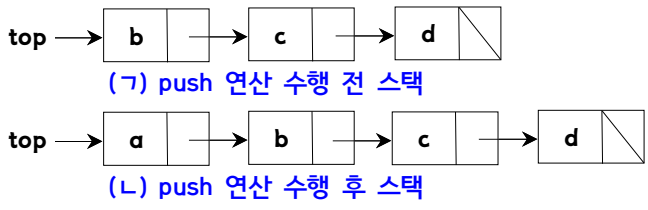
// 재귀 알고리즘

- 2의 4승은 2를 4번 곱하는 것보다는 4를 2번 곱하는 것이 더 효율적이다.
- 2의 8승은 4^4으로 계산하고
- 2의 9승은 2 * 4^4으로 계산하고
- 4의 5승은 4 * 16^2으로 계산하고

예	• x = 2, n = 10일 때, pow(x, n) = ? ↓다음처럼 호출된다. • pow(2,10) = pow(4,5) = 4*pow(16,2) = pow(256,1) = 256*pow(256*256, 0) → 호출을 10번하지 않고, 5번 호출하게 된다. 1/2씩 줄어든다. → 해서, 연산시간은 O(log n)이 된다. → n이 커지면, 호출은 엄청나게 줄어든다.

정답 : ②

3. 다음은 연결리스트를 이용하여 스택을 표현한 것이다. 이에 대한 설명으로 옳지 않은 것은? (단, push는 스택에 자료를 삽입하는 연산이고, pop은 스택에서 자료를 삭제하는 연산이다) [2014년 국가 7급]



① 스택에 가장 최근에 입력된 자료는 top이 지시한다.
② 스택에 입력된 자료 중 d가 가장 오래된 자료이다.
③ (ㄴ)에서 자료 c를 가져오려면 pop 연산이 2회 필요하다.
④ (ㄱ)에서 자료의 입력된 순서는 d, c, b이다.

♤ 연결리스트를 이용한 스택

- (ㄴ)에서 자료 c를 가져오려면 pop 연산이 2회 필요하다.(×)
  → (ㄴ)에서 자료 c를 가져오려면 pop 연산이 3회 필요하다.

정답 : ③

4. 다음은 배열 A에 저장된 n개의 정수를 오름차순으로 정렬하는 삽입정렬(insertion sort) 알고리즘이다. ㉠과 ㉡에 순서대로 들어갈 내용으로 옳은 것은? [2014년 국가 7급]

```
-----------------------------------------
void sort(int A[ ], int n) {
  int i, j, key;
  for (i = 1; i < n; i++) {
     key = A[i];
     for (j = i - 1; [    ㉠    ]; j--)
        [    ㉡    ]
     A[j+1] = key;
  }
}
-----------------------------------------
```

	㉠	㉡
①	j >= 0 && key > A[j]	A[j+1] = A[j];
②	j > 0 && key >= A[j]	A[j−1] = A[j];
③	j > 0 && key < A[j]	A[j] = A[j+1];
④	j >= 0 && key < A[j]	A[j+1] = A[j];

♤ **삽입정렬**

[예] 3 5 2 9 8을 오름차순정렬

j	[0]	[1]	[2]	[3]	[4]
초기상태	3	5	2	9	8
2	3	5	2	9	8
3	2	3	5	9	8
4	2	3	5	9	8
5	2	3	5	8	9

- 삽입정렬은 단계가 지나면서 앞 부분이 정렬되어 가는 형태이다.
- 삽입정렬에서 삽입은 이전보다 크기가 작은 자료인 경우에 발생된다. (오름차순정렬에서)

```
void sort(int A[ ], int n) {
  int i, j, key;
  for (i = 1; i < n; i++) {
     key = A[i];
     for (j = i - 1; j >= 0 && key < A[j]; j--)
        A[j+1] = A[j];          //큰 값은 뒤로 이동된다,
     A[j+1] = key;
  }
}
```

정답 : ④

5. 순환(recursive) 함수 f()에 대한 첫 번째 호출이 f(7)일 때, f(7)을 포함하여 함수 f()가 호출되는 총 횟수는? [2014년 국가 7급]

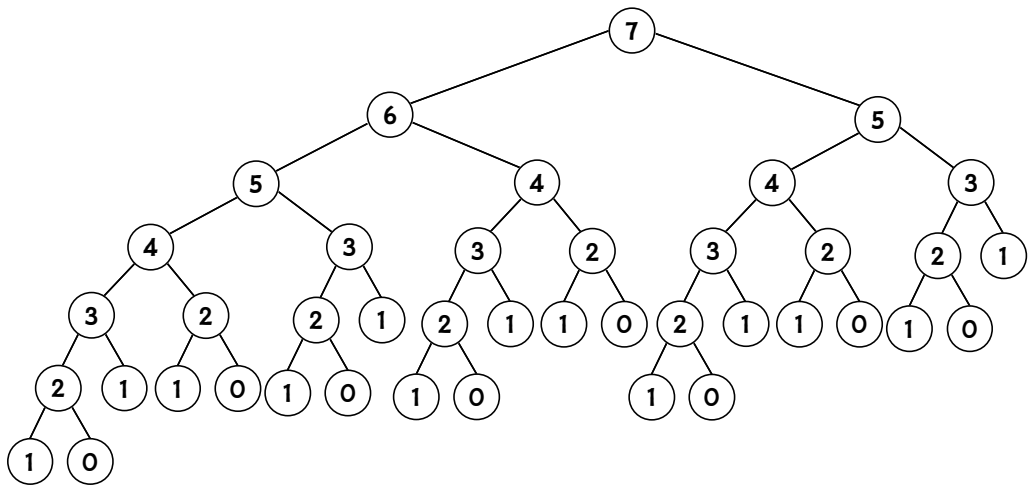
```
-----------------------------
int f(int n)
{
    if(n == 0 || n == 1)
        return n;
    return f(n-1) + f(n-2);
}
-----------------------------
```

① 25　　② 35
③ 41　　④ 51

♤ **피보나치수열과 재귀트리**

// 다음은 피보나치수열에서 f(7)을 구하는 재귀 알고리즘에 대한 재귀트리이다.

- f(7) = f(6) + f(5) → f(7)을 구하기 위해서는 f(6)과 f(5)가 필요하고,
- f(6) = f(5) + f(4) → f(6)을 구하기 위해서는 f(5)와 f(4)가 필요로 한다.
- f(5) = f(4) + f(3)
- f(4) = f(3) + f(2)
- f(3) = f(2) + f(1)
- f(2) = f(1) + f(0)
- f(1) = 1 → 완료조건
- f(0) = 0 → 완료조건



- 동그라미 수가 호출 되는 수이다.**(41번)**

정답 : ③