IT CookBook, C 로 배우는 쉬운 자료구조, 개정 3 판 Self Test 해답

1장. Self Test

p.30 ① -39의 존 형식 : 1111 0000 1111 0011 1101 1001 (F0 F3 D9)
② -39의 팩 형식 : 0000 0000 0000 0011 1001 1101 (00 03 9D)

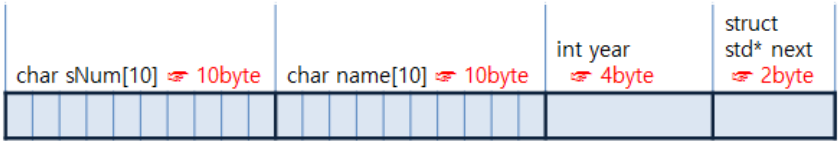
p.32 ① 부호와 절대값 형식 : 10000000 00000000 00100111
② 1의 보수 형식 : 11111111 11111111 11011000
③ 2의 보수 형식 : 11111111 11111111 11011001

p.54 시간복잡도 $O(n^2+n) = O(n^2)$

2장. Self Test

Self Test p.100

2바이트의 메모리 주소를 사용하는 시스템에서, 다음의 구조체 변수 s가 메모리에 할당 되었을 때, 내부구조와 할당되는 메모리 크기는 얼마인가?




```
struct   std {
        char sNum[10];
        char name[10];
        int year;
        struct   std* next ;
};
struct   std   s;
```

☞ **구조체변수 s의 메모리 크기 : 10byte + 10byte + 4byte + 2byte = 26byte**

3장. Self Test

Self Test

p.131

C 프로그램에서 다음과 같이 배열 a를 선언하였다. 배열 a가 할당된 시작주소를 10000이라고 가정했을 때, a[2][8]의 주소를 구하여라. 그리고 a[2][8]은 몇 번째 원소인가?

int a[4][10] ;

☞ **C 프로그램이므로 행우선 순서를 적용한다.**

① a[2][8]의 주소 : 10000 + (2*10 + 8)*4byte = 10000 + 28*4byte = 10112

② a[2][8]은 29번째 원소이다.

Self Test

p.134

C 프로그램에서 다음과 같이 배열 b를 선언하였다. 배열 b가 할당된 시작주소를 10000이라고 가정했을 때, b[1][2][8]의 주소를 구하여라. 그리고 b[1][2][8]은 몇 번째 원소인가?

int b[3][4][10] ;

☞ **C 프로그램이므로 면우선 순서를 적용한다.**

① b[1][2][8]의 주소 : 10000 + (1*4*10 + 2*10 + 8)*4byte = 10000 + 68*4byte = 10272

② b[1][2][8]은 69번째 원소이다.

Self Test

p.138

다음 다항식을 ①1차원 배열 과 ②2차원 배열 에 저장하는 경우를 설명하시오.

$A(x) = 12x^{101} + 8x^{99} - 5x^{98} + 72x^2$

① 1차원 배열

0	1	2	3	4	5	...	98	99	100	101
12	0	**8**	**-5**	0	0	...	0	**72**	0	0

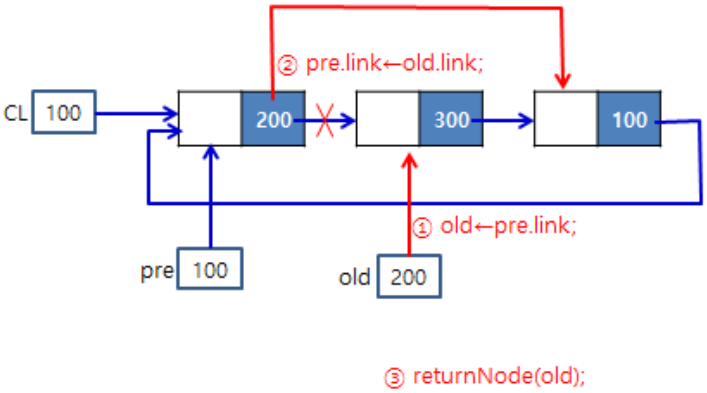
② 2차원 배열

	0	1
0	101	**12**
1	99	**8**
2	98	**-5**
3	2	**72**

4장. Self Test

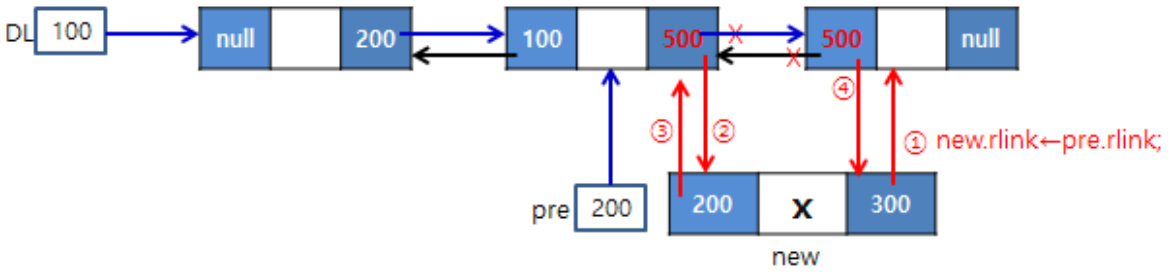
Self Test

p.193. 원형 연결 리스트 CL에서 pre=CL인 경우에 [알고리즘 4-8]을 적용하여 노드를 삭제하는 과정을 설명하여라.



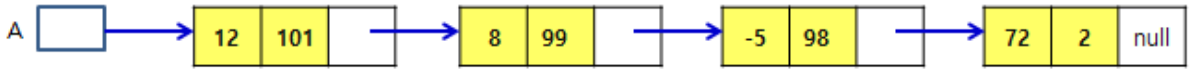
Self Test

p.206. 이중 연결 리스트 DL에서 pre=DL**.rlink**인 경우에 [알고리즘 4-9]를 적용하여 노드를 삽입하는 과정을 설명하여라.



Self Test

p.219. $A(x) = 12x^{101} + 8x^{99} - 5x^{98} + 72x^{2}$



5장. Self Test

p.237.

입력 순서가 A,B,C,D로 정해진 자료를 가지고 스택에 push연산과 pop연산을 수행하면서 pop 연산 결과로 반환되는 자료를 출력한다. 출력 가능한 결과를 만들어 보시오.

☞예) push(A), push(B), pop(), pop(), push(C), push(D), pop(), pop() ⇒ BADC

예) push(A), push(B), push(C), push(D), pop(), pop(), pop(), pop() ⇒ DCBA

예) push(A), pop(), push(B), pop(), push(C), pop(), push(D), pop() ⇒ ABCD

p.257.

다음 중위 표기식을 [알고리즘 5-4]에 따라서 후위 표기식으로 변환하여라.

(9-(4/2+1))*(5*2-2)

☞ (9-((4/2)+1))*((5*2)-2) : 우선순위에 따른 괄호 추가

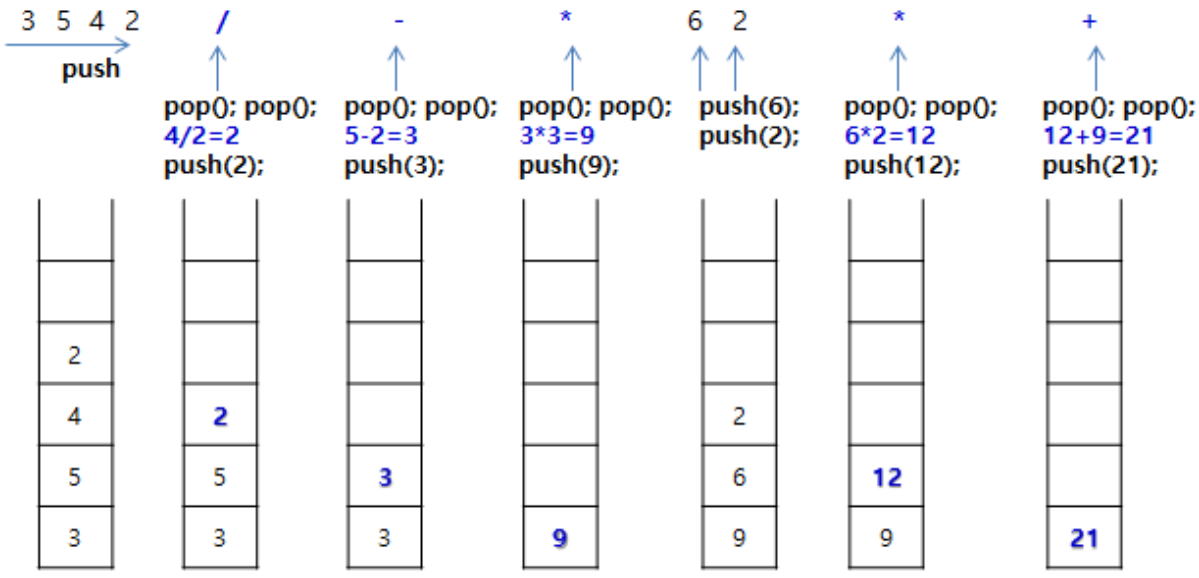
후위표기식 : **9 4 2 / 1 + - 5 2 * 2 - ***

p.259.

다음 후위 표기식을 [알고리즘 5-5]에 따라서 연산하여라.

3 5 4 2 / - * 6 2 * +

☞



6장. Self Test

Self Test

p.280.

공백 순차 큐에서 다음 연산의 수행 결과 상태를 설명하여라.

enQueue(Q1, A), enQueue(Q1, B), enQueue(Q1, C), deQueue(Q1), enQueue(Q1, D), deQueue(Q1), deQueue(Q1), enQueue(Q1, E)

☞

0	1	2	3	4	5	6	7	8	9
			D	E					

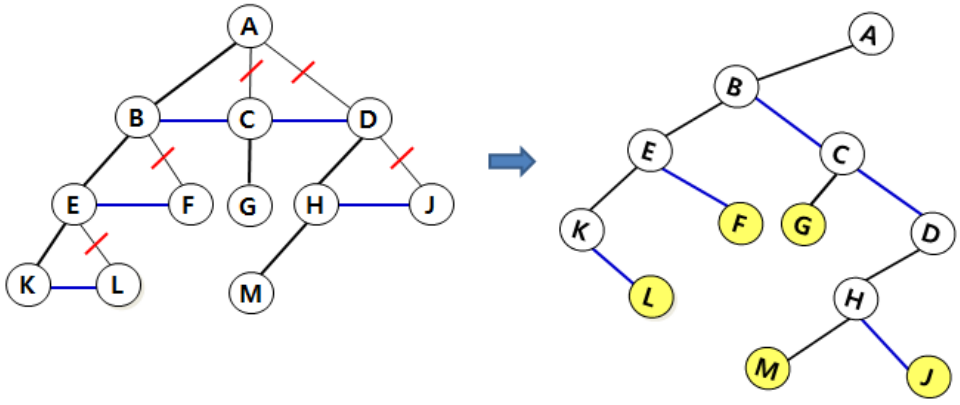
rear =4 front=2

7장. Self Test

Self Test

p.324

다음의 일반 트리를 [그림7-6]에 따라 이진 트리로 변환하고, 이진 트리의 높이와단말 노드를 구하시오.



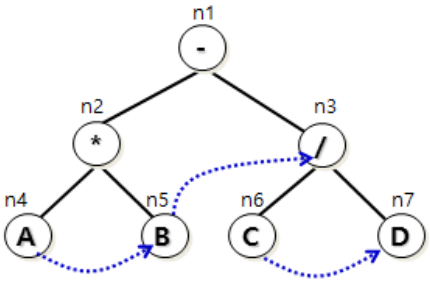
단말노드 : **F,G,J,L,M** 높이 : **5**

Self Test

p.345

[예제 7-3]의 44행~54행에서 중위 순회를 하기 위해 후속자 스레드만 갖는 스레드 이진 트리를 구성하였다. ①전위 순회를 하기 위한 스레드 이진 트리, ②후위 순회를 하기 위한 스레드 이진 트리가 되도록 44행~47행과 52행~55행을 각각 수정하시오.

☞ ①전위 순회 : **-*AB/CD**



44행~47행 : [예제 7-3]과 같음.

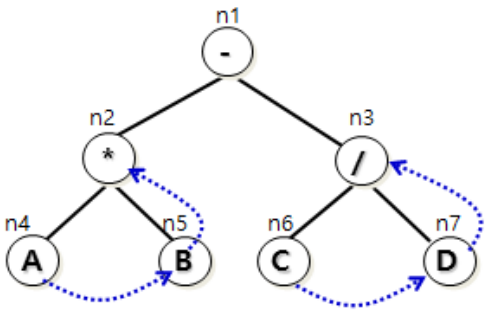
…

52행 n4 -> right = **n5**;

53행 n5 -> right = **n3**;

54행 n6 -> right = **n7**;

☞ ②후위 순회 : **AB*CD/-**



44행 treeNode* n7 = makeRootNode('D', NULL, NULL, **1**);

45행~47행 : [예제 7-3]과 같음.

…

52행 n4 -> right = **n5**;

53행 n5 -> right = **n2**;

54행 n6 -> right = **n7**;

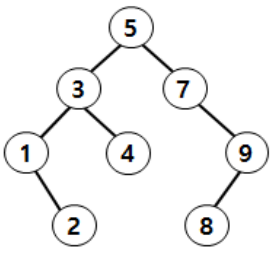
55행 n7 -> right = **n3**;

p.355

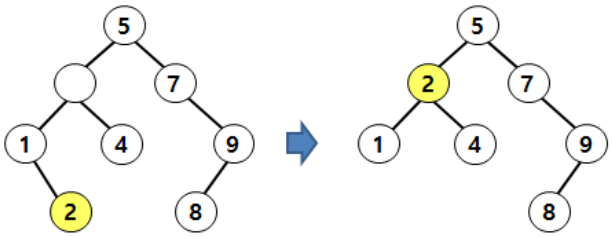
①공백 이진 탐색 트리에 5, 3, 1, 7, 4, 9, 8, 2 를 삽입하는 과정을 설명하여라.

②3을 삭제하고 이진 탐색 트리를 재구성하는 과정을 설명하여라.

☞ ①



☞ ②



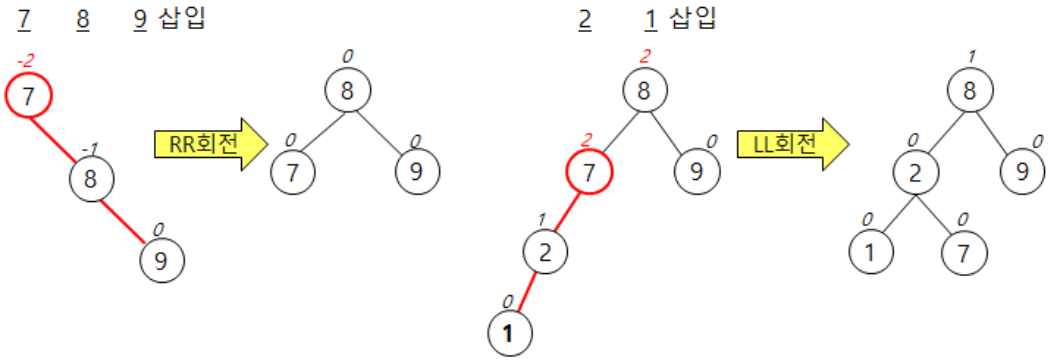
이진탐색트리에서 3을 탐색하여 삭제하고, 3의 왼쪽서브트리에서 최대값 2를 후계자로 선택하여 삭제한 3의 자리를 물려준다.

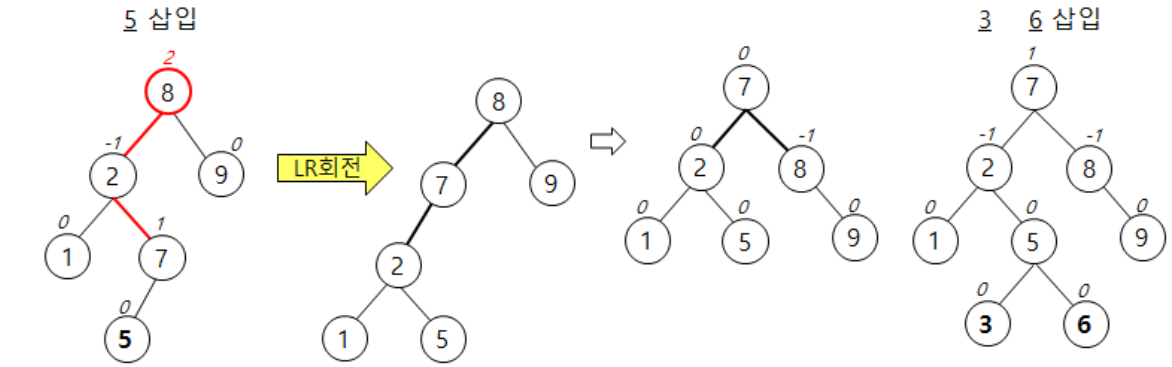
p.372

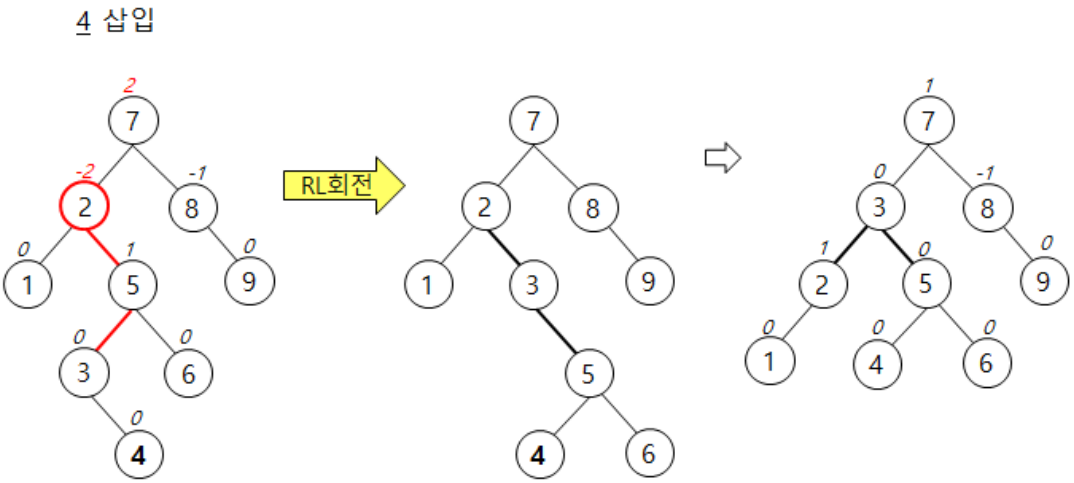
다음 노드를 순서대로 AVL 트리에 삽입하는 과정을 설명하시오.

노드 : 7 8 9 2 1 5 3 6 4

☞





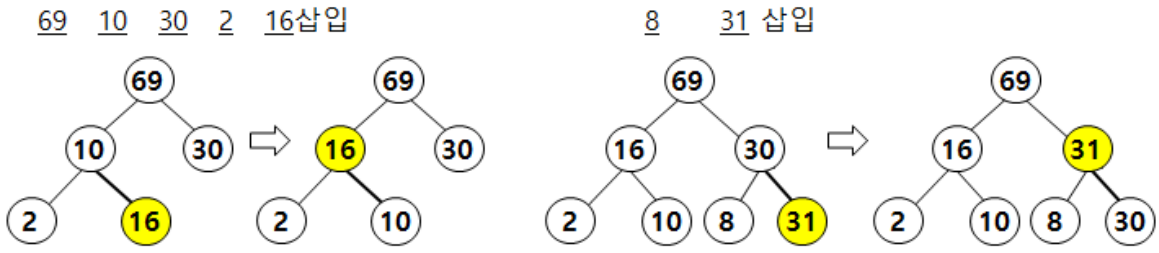


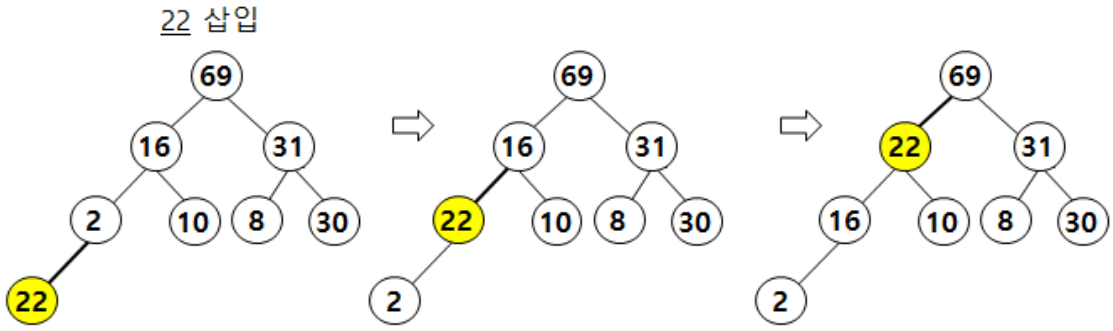
Self Test

p.381

[알고리즘 7-11]에 따라서 공백 히프에 {69, 10, 30, 2, 16, 8, 31, 22}를 삽입하는 과정을 설명하시오.

☞





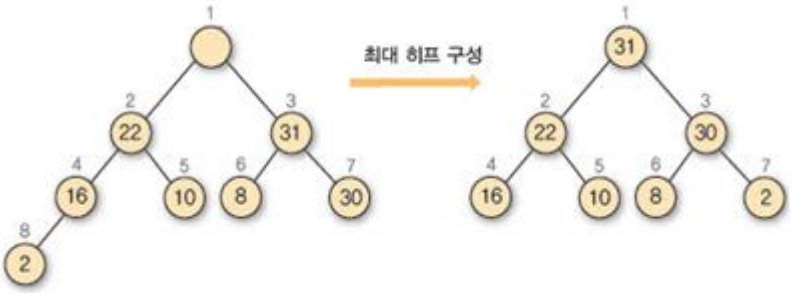
Self Test

p.383

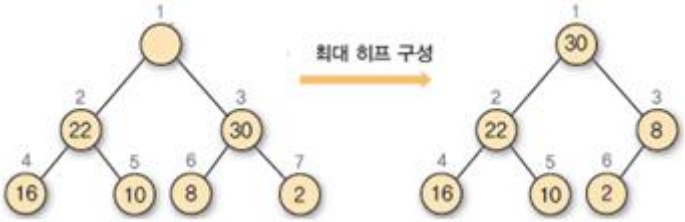
381쪽 Self Test에서 완성한 히프를 [알고리즘 7-12]에 따라 삭제 연산을 여덟 번 수행하는 과정을 설명하시오.

☞ (p.539 히프 정렬 참고)

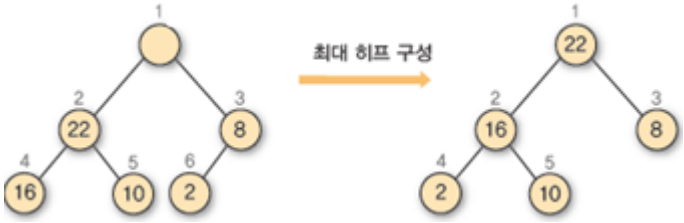
1)삭제 : 69



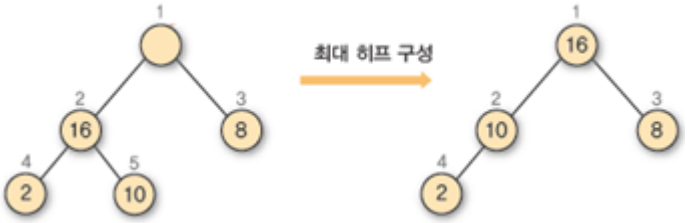
2) 삭제 : 31



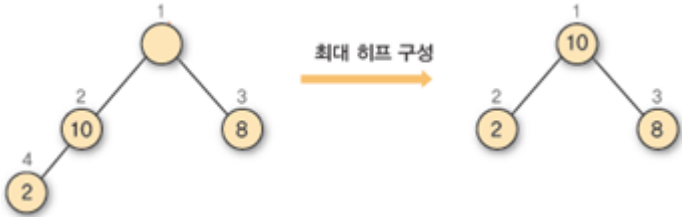
3) 삭제 : 30



4) 삭제 : 22



5) 삭제 : 16



6) 삭제 : 10

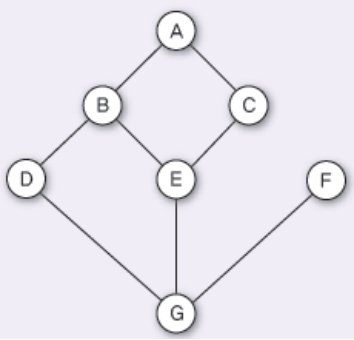
7) 삭제 : 8

8) 삭제 : 2 (삭제 후 공백히프)

8장. Self Test

p.418

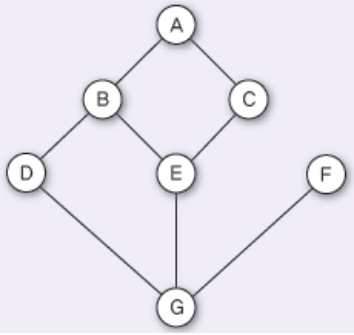
인접행렬

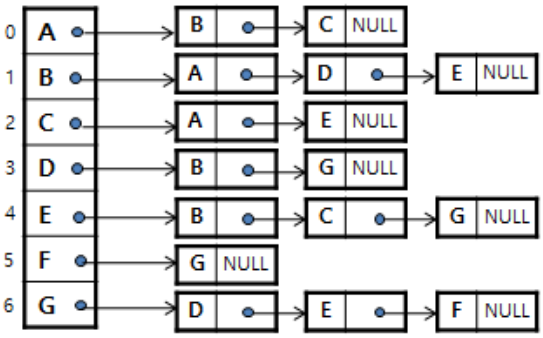


	A	B	C	D	E	F	G
	0	1	2	3	4	5	6
A 0	0	1	1	0	0	0	0
B 1	1	0	0	1	1	0	0
C 2	1	0	0	0	1	0	0
D 3	0	1	0	0	0	0	1
E 4	0	1	1	0	0	0	1
F 5	0	0	0	0	0	0	1
G 6	0	0	0	1	1	1	0

p.424

인접리스트

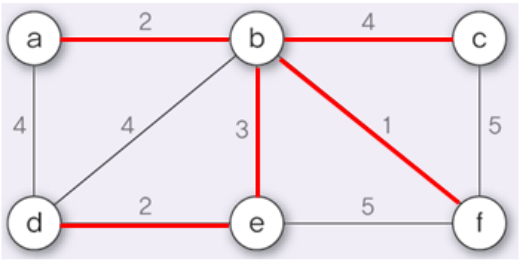




p.466

크루스칼 알고리즘 II

☞ 정점 : 6개, 간선 : 9개. ☞ 최소비용신장트리 : 간선 5개 (시작정점 a는 의미 없음)



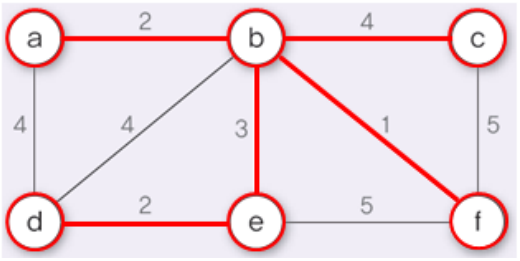
가중치	간선
1	(b, f)
2	(a , b)
2	(d, e)
3	(b, e)
4	(a, d)
4	(b, c)
4	(b, d)
5	(c, f)
5	(e, f)

Self Test

p.469

프림 알고리즘

☞ 정점 : 6개, 간선 : 9개. ☞ 최소비용신장트리 : 간선 5개



정점 a에서 시작 → 정점 b와 (a, b) 삽입 →
정점 f와 (b, f) 삽입 → 정점 e와 (b, e) 삽입 →
정점 d와 (d, e) 삽입 → 정점 c와 (b, c) 삽입

Self Test

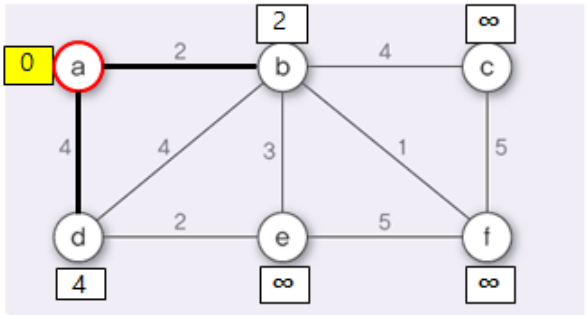
p.474

다익스트라 알고리즘

☞ 1) 가중치인접행렬에서 시작정점 a에 대한 행을 뽑아서 distance를 만들고 시작.

가중치 인접행렬

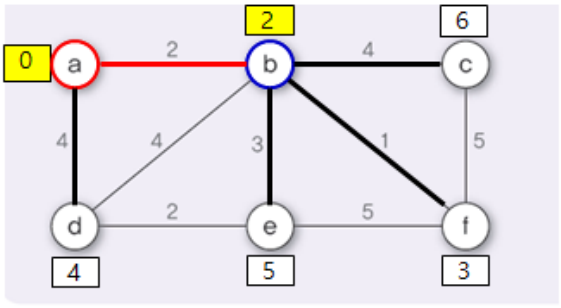
	a	b	c	d	e	f
a	0	2	∞	4	∞	∞
b	2	0	4	4	3	1
c	∞	4	0	∞	∞	5
d	4	4	∞	0	2	∞
e	∞	3	∞	2	0	5
f	∞	1	5	∞	5	0



S = {a}

	a	b	c	d	e	f
distance	0	2	∞	4	∞	∞

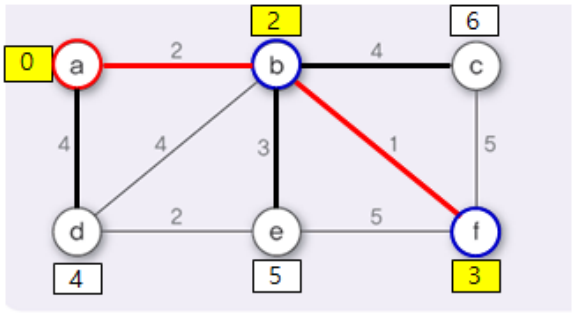
2) 최소 경로값의 **정점 b 선택**. 현재 b의 경로값을 최소값으로 확정하고, b를 거쳐서 가는 경로에 대해 최소 경로 수정여부 확인하여 c, e, f 수정.



S = {a, **b**}

	a	b	c	d	e	f
distance	0	2	6	4	5	3

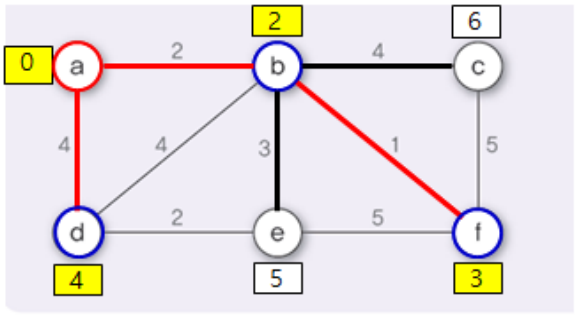
3) 최소 경로값의 **정점 f 선택**. 현재 f의 경로값을 최소값으로 확정하고, f를 거쳐서 가는 경로에 대해 최소 경로 수정여부 확인. 수정 사항 없음.



S = {a, b, **f**}

	a	b	c	d	e	f
distance	0	2	6	4	5	3

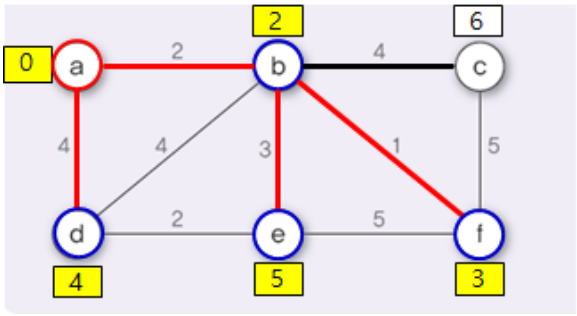
4) 최소 경로값의 **정점 d 선택**. 현재 d의 경로값을 최소값으로 확정하고, d를 거쳐서 가는 경로에 대해 최소 경로 수정여부 확인. 수정 사항 없음.



S = {a, b, **d**, f}

	a	b	c	d	e	f
distance	0	2	6	4	5	3

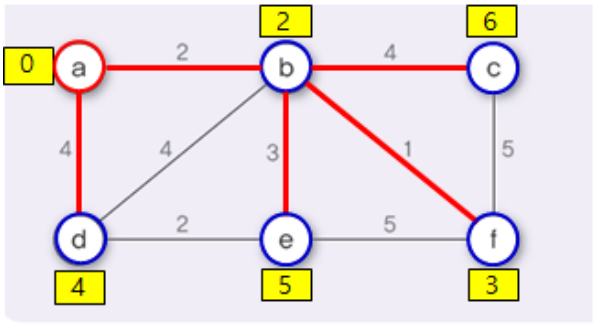
5) 최소 경로값의 **정점 e 선택**. 현재 e의 경로값을 최소값으로 확정하고, e를 거쳐서 가는 경로에 대해 최소 경로 수정여부 확인. 수정 사항 없음.



S = {a, b, d, e, f}

	a	b	c	d	e	f
distance	0	2	6	4	5	3

6) 최소 경로값의 **정점 c 선택**. 현재 c의 경로값을 최소값으로 확정하고, c를 거쳐서 가는 경로에 대해 최소 경로 수정여부 확인. 수정 사항 없음.



S = {a, b, c, d, e, f}

	a	b	c	d	e	f
distance	0	2	6	4	5	3

Self Test

p.480

플로이드 알고리즘

☞ 1) 초기 상태 A^{-1} : 인접 행렬 weight를 배열에 복사하여 초기화한다.

(무방향 그래프의 인접행렬은 대각선을 기준으로 대칭이 되므로 우삼각형만 처리하고 변경사항을 좌삼각형에 반영한다.)

A^{-1}	a	b	c	d	e	f
a	0	2	∞	4	∞	∞
b	2	0	4	4	3	1
c	∞	4	0	∞	∞	5
d	4	4	∞	0	2	∞
e	∞	3	∞	2	0	5
f	∞	1	5	∞	5	0

2) A^0 : 두 정점 사이의 최단 경로에서 정점 a를 거쳐서 가는 경로를 고려하여 최단 경로를 수정한다. ☞ 변경사항 없음. $A^0 = A^{-1}$

3) A^1 : A^0에서 정점 b를 추가로 거쳐서 가는 경로를 고려하여 최단 경로를 수정한다.

A^1	a	b	c	d	e	f
a	**0**	**2**	**6**	**4**	**5**	**3**
b	2	**0**	**4**	**4**	**3**	**1**
c	**6**	4	**0**	**8**	**7**	5
d	4	4	8	**0**	**2**	**5**
e	**5**	3	7	2	**0**	**4**
f	**3**	1	5	**5**	**4**	**0**

4) A^2 : 두 정점 사이의 최단 경로에서 정점 c를 거쳐서 가는 경로를 고려하여 최단 경로를 수정한다. ☞ 변경사항 없음. $A^2 = A^1$

5) A^3 : 두 정점 사이의 최단 경로에서 정점 d를 거쳐서 가는 경로를 고려하여 최단 경로를 수정한다. ☞ 변경사항 없음. $A^3 = A^2$

6) A^4 : 두 정점 사이의 최단 경로에서 정점 e를 거쳐서 가는 경로를 고려하여 최단 경로를 수정한다. ☞ 변경사항 없음. $A^4 = A^3$

7) A^5 : 두 정점 사이의 최단 경로에서 정점 f를 거쳐서 가는 경로를 고려하여 최단 경로를 수정한다. ☞ 변경사항 없음. $A^5 = A^4$

9장. Self Test

p.516

피봇의 위치를 정렬할 원소의 마지막 원소로 정하기.

☞ [예제 9-3]의 11행 수정

```
pivot = end;
```

p.5144

AVL트리를 트리 정렬에 이용할 수 있는가?

☞**AVL트리도 정렬에 사용할 수 있다.** 정렬할 원소들을 AVL 트리로 구성 한후에 중위 순회를 수행하면 오름차순 정렬된 결과를 구할 수 있다.

하지만, AVL트리는 이진탐색트리 연산에 균형을 유지하기 위한 회전 연산이 추가로 필요하기 때문에 이진탐색트리 보다 재구성 연산이 복잡해진다. (AVL트리는 정렬에도 이용 가능하지만, 탐색에 최적화된 트리이다.)

10장. Self Test

Self Test

p.584

슬롯이 두 개인 경우의 선행 개방 주소법

키값은 {45, 9, 10, 96, 25}

	슬롯0	슬롯1
버킷0	①45	③10
1	④96	⑤25
2		
3		
4	②9	

h(45) = 45 mod 5 = 0 ☞ 버킷0의 슬롯0에 저장

h(9) = 9 mod 5 = 4 ☞ 버킷4의 슬롯0에 저장

h(10) = 10 mod 5 = 0 ☞ 버킷0의 슬롯1에 저장

h(96) = 96 mod 5 = 1 ☞ 버킷1의 슬롯0에 저장

h(25) = 25 mod 5 = 0 ☞ 버킷0에 빈 슬롯이 없으므로, 그 다음 슬롯을 순차적으로 검색한다. 버킷1의 슬롯1이 비어있으므로 저장한다.