

IT@COOKBOOK, C 로 배우는 쉬운 자료구조 4 판

SELF TEST 해답

1장

p.30

-39를 3바이트의 ① 존 형식과 ② 팩 형식으로 나타내시오.

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

-39를 3바이트의 ① 부호와 절댓값 형식, ② 1의 보수 형식, ③ 2의 보수 형식으로 나타내시오.

① 부호와 절댓값 형식 : **1**0000000 00000000 00**100111**

② 1의 보수 형식 : 11111111 11111111 11**011000**

③ 2의 보수 형식 : 11111111 11111111 11**011001**

p.55

다음 알고리즘의 시간 복잡도를 구하시오.

```
A(n)
  for (m←1,i←1; i<n; i←i+1) do {
    while (m < n) do {
      x ← x * i;
      m ← m + 1;
    }
  }
  for (j←0; j<n; j←j+1) do
    x ← x + j;
  return x;
end
```

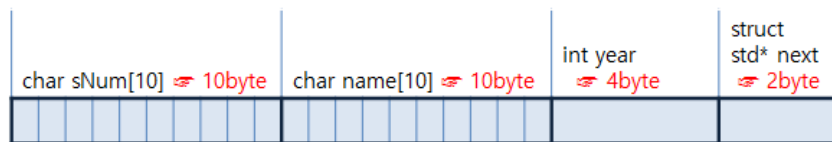
시간 복잡도 $O(n^2+n) = O(n^2)$

2장

p.98

2바이트의 메모리 주소를 사용하는 시스템에서, 다음의 구조체 변수 `s`가 메모리에 할당되었을 때, 내부 구조와 할당되는 메모리 크기는?

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



☞ 구조체 변수 `s`의 메모리 크기 : 10byte + 10byte + 4byte + 2byte = 26byte

3장

p.126

C 프로그램에서 다음과 같이 배열 a를 선언하였다. 배열 a가 할당된 시작 주소를 10000이라고 가정했을 때, ① a[2][8] 주소와 ② a[2][8]이 몇 번째 원소인지 구하시오.

```
int a[4][10];
```

☞ C 프로그램이므로 행 우선 순서를 적용한다.

- ① a[2][8]의 주소 : $10000 + (2 \times 10 + 8) \times 4\text{byte} = 10000 + 28 \times 4\text{byte} = 10112$
- ② a[2][8]은 29번째 원소이다.

p.129

C 프로그램에서 다음과 같이 배열 b를 선언하였다. 배열 b가 할당된 시작 주소를 10000이라고 가정했을 때, ① b[1][2][8] 주소와 ② b[1][2][8]이 몇 번째 원소인지 구하시오.

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

☞ C 프로그램이므로 면 우선 순서를 적용한다.

- ① b[1][2][8]의 주소 : $10000 + (1 \times 4 \times 10 + 2 \times 10 + 8) \times 4\text{byte} = 10000 + 68 \times 4\text{byte} = 10272$
- ② b[1][2][8]은 69번째 원소이다.

p.144

다음 다항식을 ① 1차원 배열과 ② 2차원 배열에 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

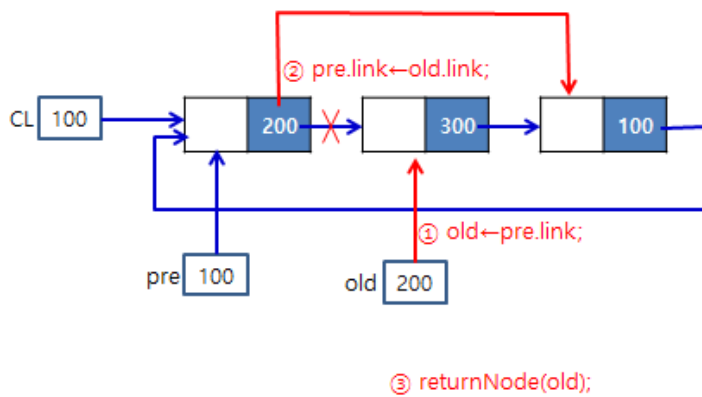
② 2차원 배열

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

4장

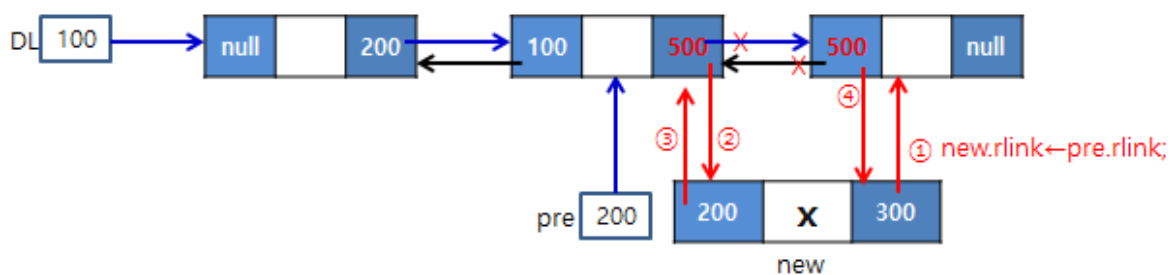
p.199

원형 연결 리스트 CL에서 $pre = CL$ 인 경우에 [알고리즘 4-8]을 적용하여 노드를 삭제하는 과정을 설명하시오.



p.212

이중 연결 리스트 DL에서 $pre = DL.rlink$ 인 경우에 [알고리즘 4-9]를 적용하여 노드를 삽입하는 과정을 설명하시오.



5장

p.245

입력 순서가 A, B, C, D로 정해진 자료를 스택에 push 연산과 pop 연산을 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() $\Rightarrow$ BADC

예) push(A), push(B), push(C), push(D), pop(), pop(), pop(), pop() $\Rightarrow$ DCBA

예) push(A), pop(), push(B), pop(), push(C), pop(), push(D), pop() $\Rightarrow$ ABCD

p.266

다음 중위 표기식을 [알고리즘 5-4]에 따라 후위 표기식으로 변환하시오.

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

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

후위 표기식 : $9\ 4\ 2\ /\ 1\ +\ -\ 5\ 2\ *\ 2\ -\ *$

6장

p.294

공백 순차 큐에서 다음 연산의 수행 결과 상태를 설명하시오.

enqueue(Q1, A), enqueue(Q1, B), enqueue(Q1, C), dequeue(Q1),
enqueue(Q1, D), dequeue(Q1), dequeue(Q1), enqueue(Q1, E)



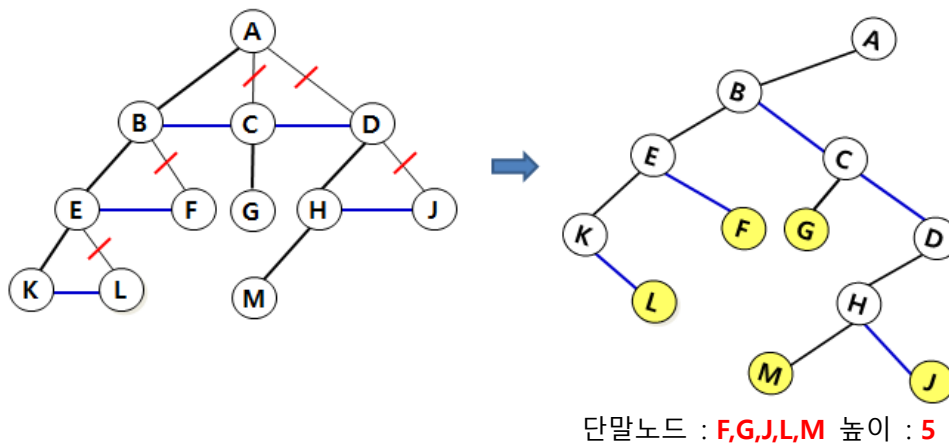
0	1	2	3	4	5	6	7	8	9
			D	E					

rear =4 front=2

7장

p.341

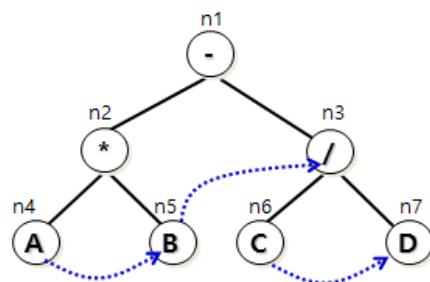
다음 일반 트리를 [그림 7-6]에 따라 이진 트리로 변환하고, 이진 트리의 높이와 단말 노드를 구하시오.



p.364

[예제 7-3]의 [ex7_3.c]의 06~12행에서 중위 순회를 하기 위해 후속자 스레드만 갖는 스레드 이진 트리를 구성하였다. ① 전위 순회를 하기 위한 스레드 이진 트리, ② 후위 순회를 하기 위한 스레드 이진 트리가 되도록 06~12행을 각각 수정하시오.

① 전위 순회 : **-*AB/CD**



06~09행 : [ex7_3.c]과 같음

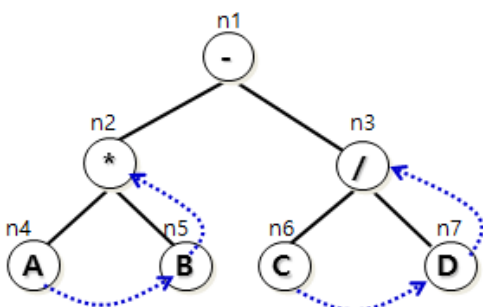
...

14행 : n4 -> right = **n5**;

15행 : n5 -> right = **n3**;

16행 : n6 -> right = **n7**;

② 후위 순회 : **AB*CD/-**



06행 : treeNode* n7 = makeRootNode('D', NULL, NULL, **1**);

07~09행 : [ex7_3.c]과 같음

...

14행 : n4 -> right = n5;

15행 : n5 -> right = n2;

16행 : n6 -> right = n7;

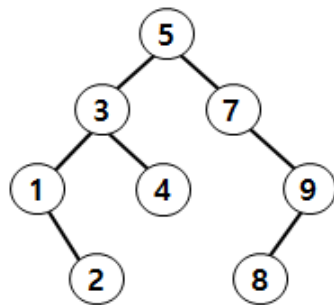
17행 : n7 -> right = n3;

p.374

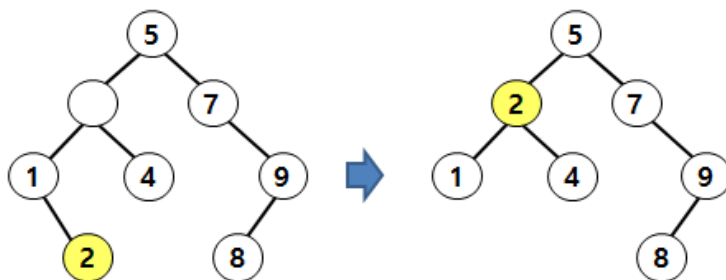
① 공백 이진 탐색 트리에 5, 3, 1, 7, 4, 9, 8, 2를 삽입하는 과정을 설명하시오.

② 3을 삭제하고 이진 탐색 트리를 재구성하는 과정을 설명하시오.

①



②



이진 탐색 트리에서 3을 탐색하여 삭제하고, 3의 왼쪽 서브 트리에서 최대값 2를 후계자로 선택하여 삭제한 3의 자리를 물려준다.

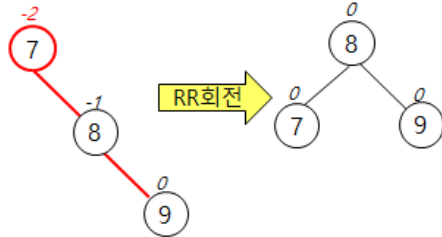
p.391

다음 노드를 순서대로 AVL 트리에 삽입하는 과정을 설명하시오.

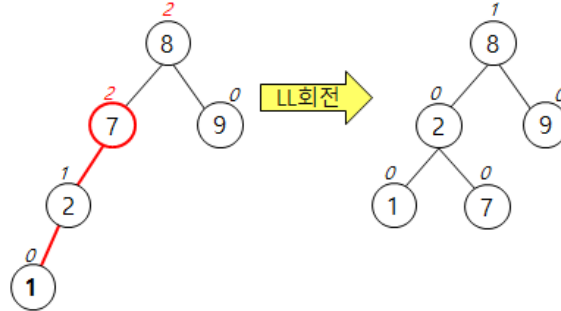
노드 : 7 8 9 2 1 5 3 6 4



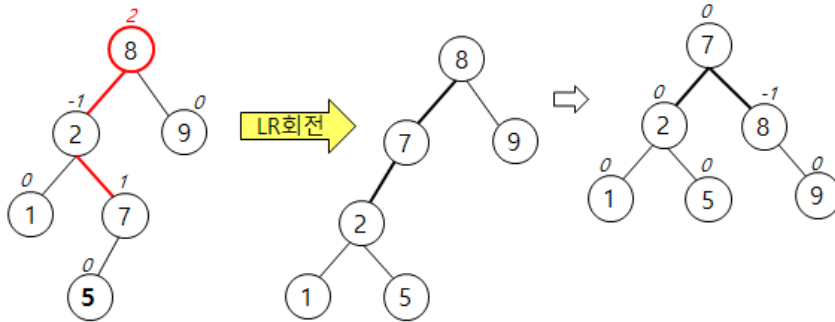
7 8 9 삽입



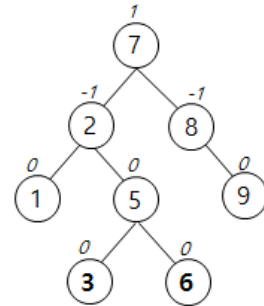
2 1 삽입



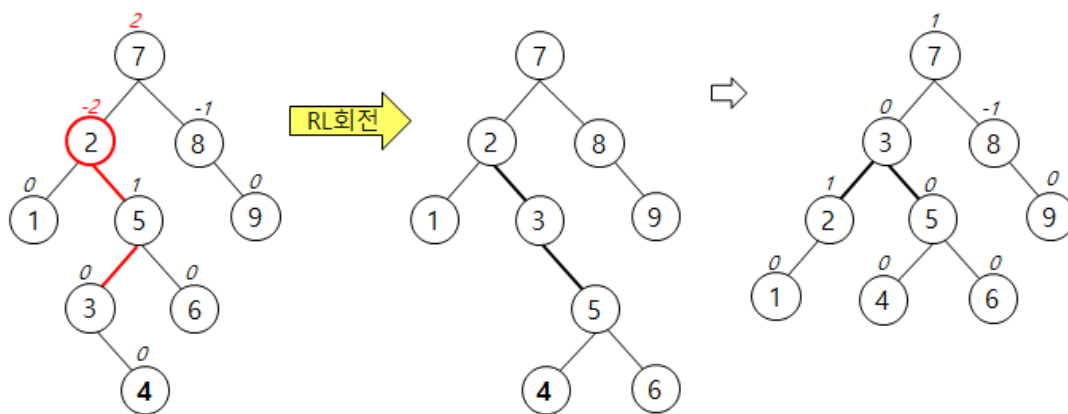
5 삽입



3 6 삽입

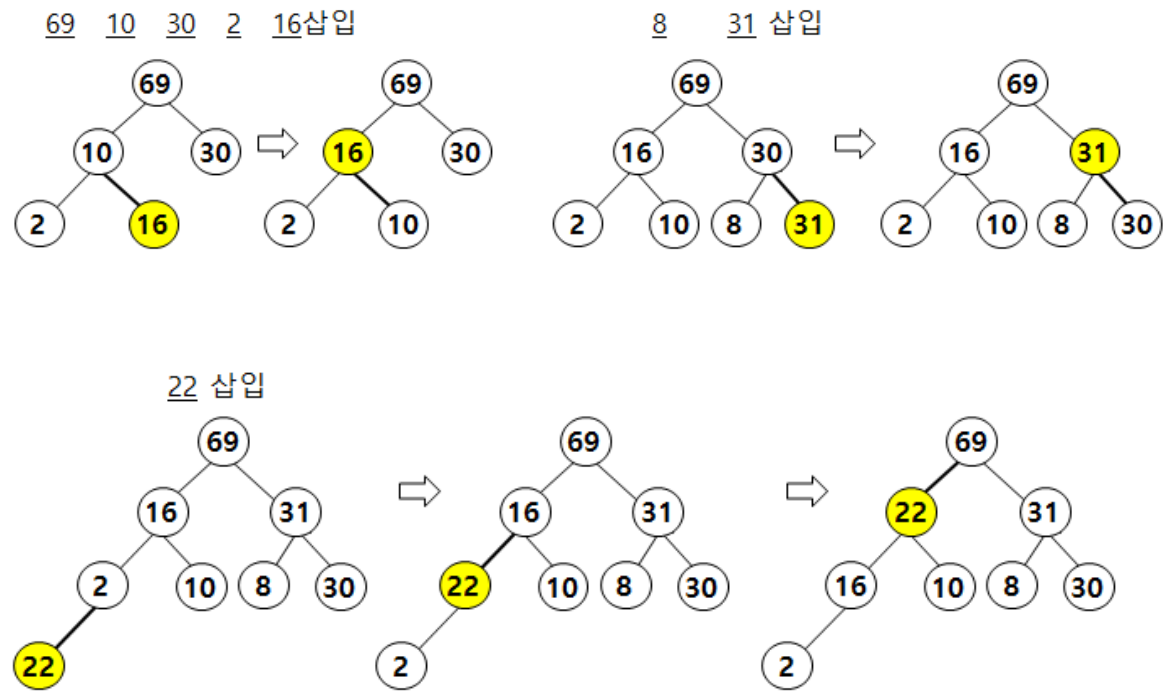


4 삽입



p.400

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

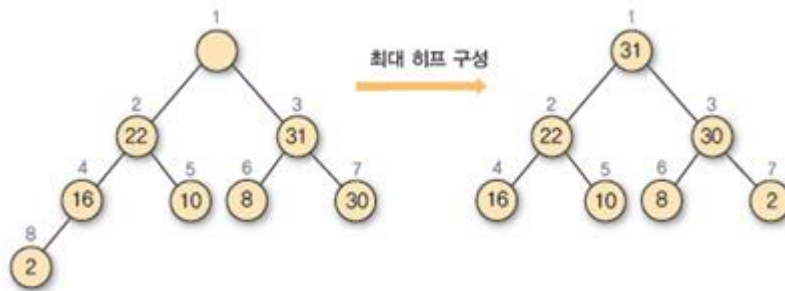


p.402

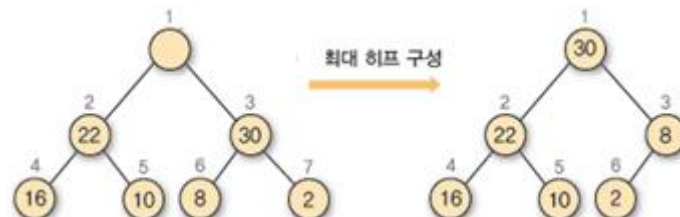
400쪽 Self Test에서 완성한 히프를 [알고리즘 7-12]에 따라 삭제 연산을 여덟 번 수행하는 과정을 설명하시오.

(p.557 히프 정렬 참고)

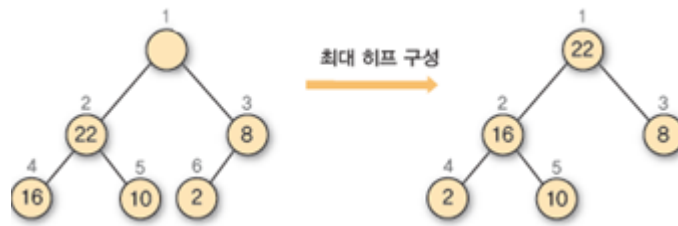
1) 삭제 : 69



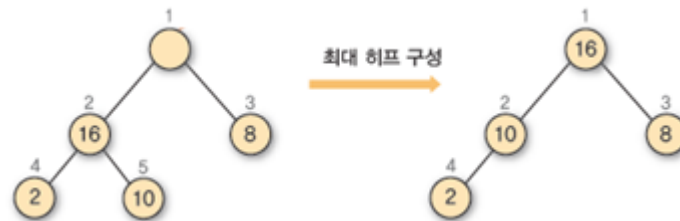
2) 삭제 : 31



3) 삭제 : 30



4) 삭제 : 22



5) 삭제 : 16



6) 삭제 : 10



7) 삭제 : 8

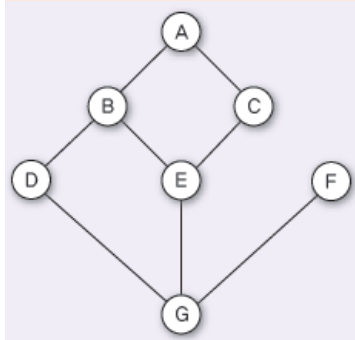


8) 삭제 : 2(삭제 후 공백 힙)

8장

p.434

그래프를 인접 행렬로 표현하시오.

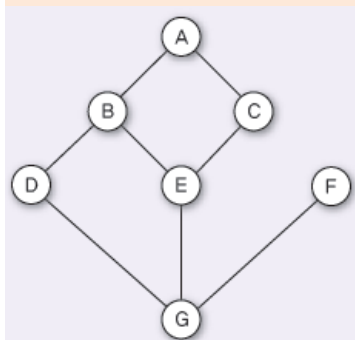


인접 행렬

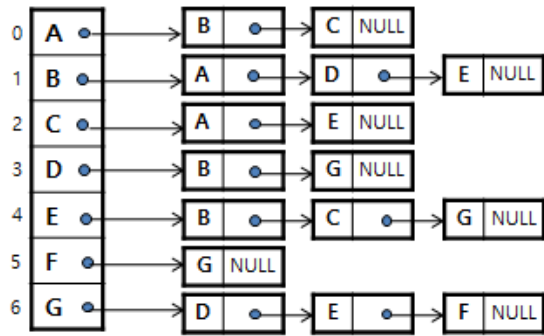
	A	B	C	D	E	F	G
	0	1	2	3	4	5	6
A 0	0	1	1	0	0	0	0
B 1	1	0	0	1	1	0	0
C 2	1	0	0	0	1	0	0
D 3	0	1	0	0	0	0	1
E 4	0	1	1	0	0	0	1
F 5	0	0	0	0	0	0	1
G 6	0	0	0	1	1	1	0

p.440

그래프를 인접 리스트로 표현하시오.



인접 리스트



p.479

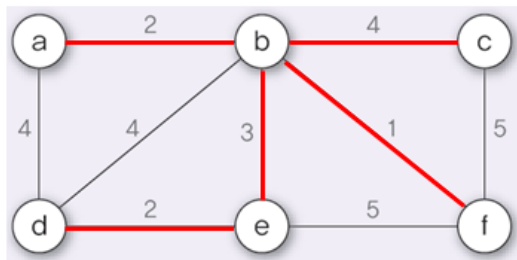
다음 가중 그래프에 대해서 정점 a를 시작으로 크루스칼 알고리즘 Π 를 수행하여 최소 비용 신장 트리를 구하시오.



크루스칼 알고리즘 Π

정점 : 6개, 간선 : 9개

최소 비용 신장 트리 : 간선 5개(시작 정점 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)

p.482

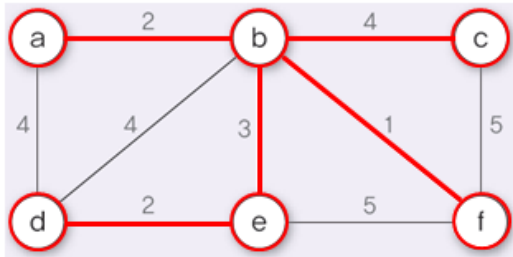
다음 가중 그래프에 대해 정점 a를 시작으로 프림 알고리즘을 수행하여 최소 비용 신장 트리를 구하시오.



프림 알고리즘

정점 : 6개, 간선 : 9개

최소 비용 신장 트리 : 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

p.488

다음 가중 그래프에 대해서 정점 a에서 모든 정점에 이르는 최단 경로를 다익스트라 알고리즘을 적용하여 구하시오.

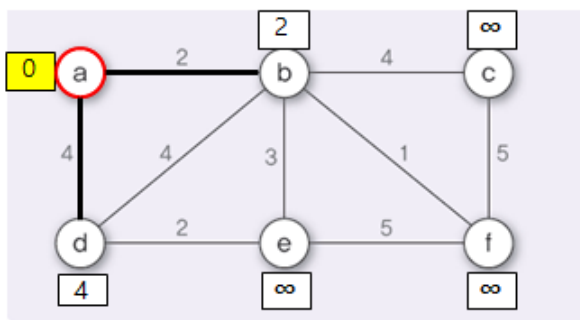


다익스트라 알고리즘

1) 가중치 인접 행렬에서 시작 정점 a에 대한 행을 뽑아서 distance를 만들고 시작.

가중치 인접행렬

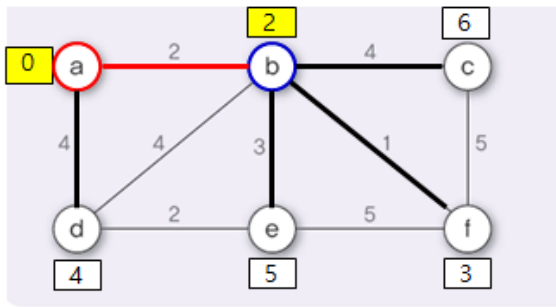
	a	b	c	d	e	f
a	0	2	∞	4	∞	∞
b	2	0	4	4	3	1
c	∞	4	0	∞	∞	5
d	4	4	∞	0	2	∞
e	∞	3	∞	2	0	5
f	∞	1	5	∞	5	0



$S = \{a\}$

	a	b	c	d	e	f
distance	0	2	∞	4	∞	∞

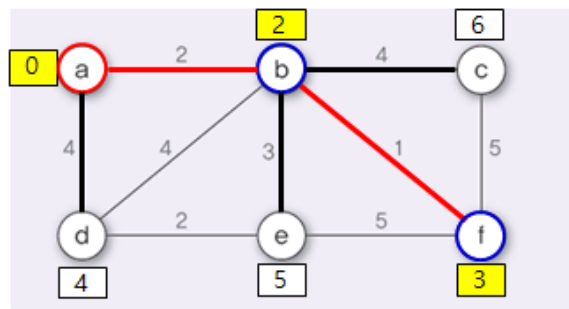
2) 최소 경로값의 정점 b 선택. 현재 b의 경로값을 최소값으로 확정하고, b를 거쳐서 가는 경로에 대해 최소 경로 수정 여부 확인하여 c, e, f 수정.



$$S = \{a, b\}$$

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

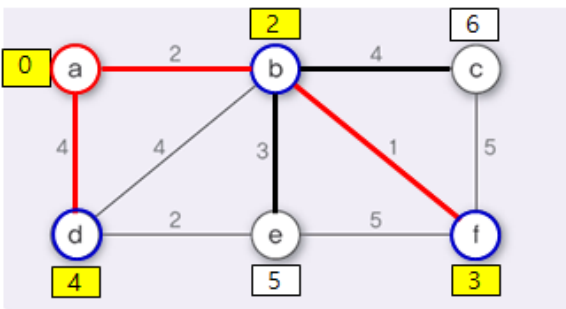
3) 최소 경로값의 정점 **f** 선택. 현재 f의 경로값을 최소값으로 확정하고, f를 거쳐서 가는 경로에 대해 최소 경로 수정 여부 확인. 수정 사항 없음.



$$S = \{a, b, f\}$$

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

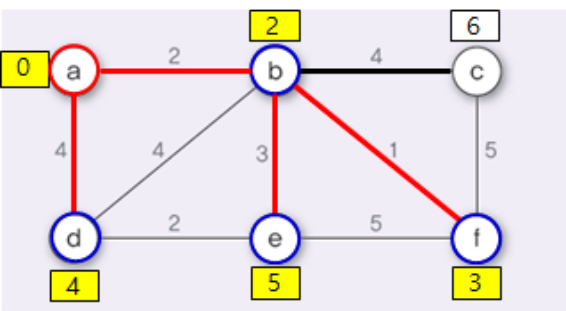
4) 최소 경로값의 정점 **d** 선택. 현재 d의 경로값을 최소값으로 확정하고, d를 거쳐서 가는 경로에 대해 최소 경로 수정 여부 확인. 수정 사항 없음.



$$S = \{a, b, d, f\}$$

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

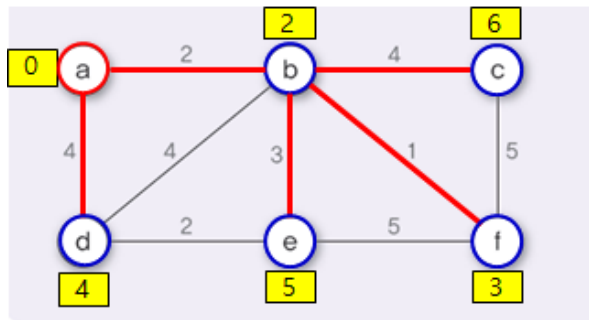
5) 최소 경로값의 정점 **e** 선택. 현재 e의 경로값을 최소값으로 확정하고, e를 거쳐서 가는 경로에 대해 최소 경로 수정 여부 확인. 수정 사항 없음.



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

	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 수정 여부 확인. 수정 사항 없음.



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

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

p.494

다음 가중 그래프에 대해서 모든 정점 사이의 최단 경로를 플로이드 알고리즘을 적용하여 구하시오.



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^0 에서 정점 b를 추가로 거쳐서 가는 경로를 고려하여 최단 경로를 수정한다.

A¹

	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) A² : 두 정점 사이의 최단 경로에서 정점 c를 거쳐서 가는 경로를 고려하여 최단 경로를 수정한다. ➡ **변경사항 없음**. A² = A¹

5) A³ : 두 정점 사이의 최단 경로에서 정점 d를 거쳐서 가는 경로를 고려하여 최단 경로를 수정한다. ➡ **변경사항 없음**. A³ = A²

6) A⁴ : 두 정점 사이의 최단 경로에서 정점 e를 거쳐서 가는 경로를 고려하여 최단 경로를 수정한다. ➡ **변경사항 없음**. A⁴ = A³

7) A⁵ : 두 정점 사이의 최단 경로에서 정점 f를 거쳐서 가는 경로를 고려하여 최단 경로를 수정한다. ➡ **변경사항 없음**. A⁵ = A⁴

9장

p.535

피벗 위치를 정렬할 원소의 마지막 원소(오른쪽 끝에 있는 원소)로 정하여 퀵 정렬을 수행하는 프로그램을 [예제 9-3]을 수정하여 작성하시오.



[예제 9-3]의 [quickSort1.c]의 10행 수정

```
pivot = end;
```

p.562

트리 정렬은 이진 탐색 트리를 이용한 정렬 방법이다. 7장에서 학습한 AVL 트리를 정렬로 이용할 수 있는지 설명하시오.



AVL 트리를 트리 정렬에 이용할 수 있는가?

AVL 트리도 정렬에 사용할 수 있다. 정렬할 원소를 AVL 트리로 구성한 후에 중위 순회를 수행하면 오름차순 정렬된 결과를 구할 수 있다.

하지만, AVL 트리는 이진 탐색 트리 연산에 균형을 유지하기 위한 회전 연산이 추가로 필요하기 때문에 이진 탐색 트리보다 재구성 연산이 복잡해진다.

AVL 트리는 정렬에도 이용 가능하지만 탐색에 최적화된 트리이다.

10장

p.606

위에서 설명한 선형 개방 주소법을 사용하여 오버플로를 처리하는 과정을 해시 테이블의 버킷이 슬롯을 두 개 가지고 있는 경우에 대해서 설명하시오.



슬롯이 두 개인 경우의 선형 개방 주소법

키값은 {45, 9, 10, 96, 25}

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

$h(45) = 45 \bmod 5 = 0$ → 버킷0의 슬롯0에 저장

$h(9) = 9 \bmod 5 = 4$ → 버킷4의 슬롯0에 저장

$h(10) = 10 \bmod 5 = 0$ → 버킷0의 슬롯1에 저장

$h(96) = 96 \bmod 5 = 1$ → 버킷1의 슬롯0에 저장

$h(25) = 25 \bmod 5 = 0$ → 버킷0에 빈 슬롯이 없으므로, 그 다음 슬롯을 순차적으로 검색한다. 버킷1의 슬롯1이 비어 있으므로 저장한다.