1 히프 개념과 추상 자료형

1 히프(Heap)의 개념

- 완전 이진 트리에 있는 노드 중에서 키 값이 가장 큰 노드나 키 값이 가장 작은 노드를 찾기 위해서 만든 자료구조
- 일반적으로 히프는 최대 히프를 의미
- 같은 키 값의 노드가 중복 되어 있을 수도 있음

1 히프(Heap)의 개념

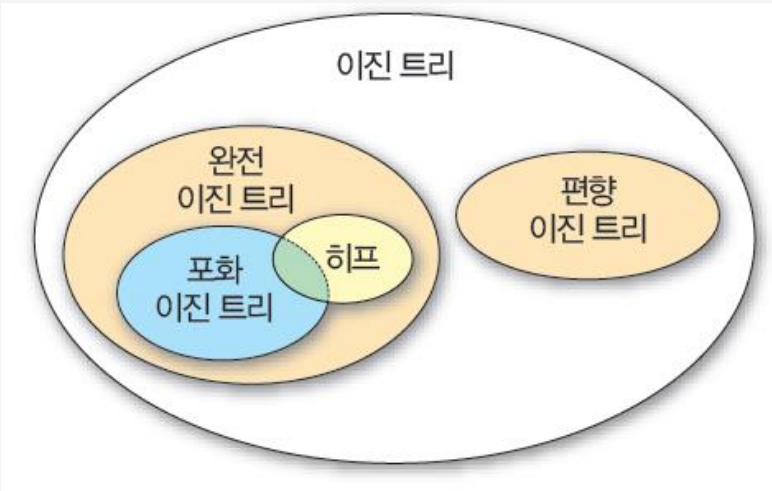
최대 히프(Max Heap)

- 키값이 가장 큰 노드를 찾기 위한 완전 이진 트리
- {부모노드의 키값 ≥ 자식노드의 키값}
- 루트 노드 : 키값이 가장 큰 노드

최소 히프(Min Heap)

- 키값이 가장 작은 노드를 찾기 위한 완전 이진 트리
- {부모노드의 키값 ≤ 자식노드의 키값}
- 루트 노드 : 키값이 가장 작은 노드

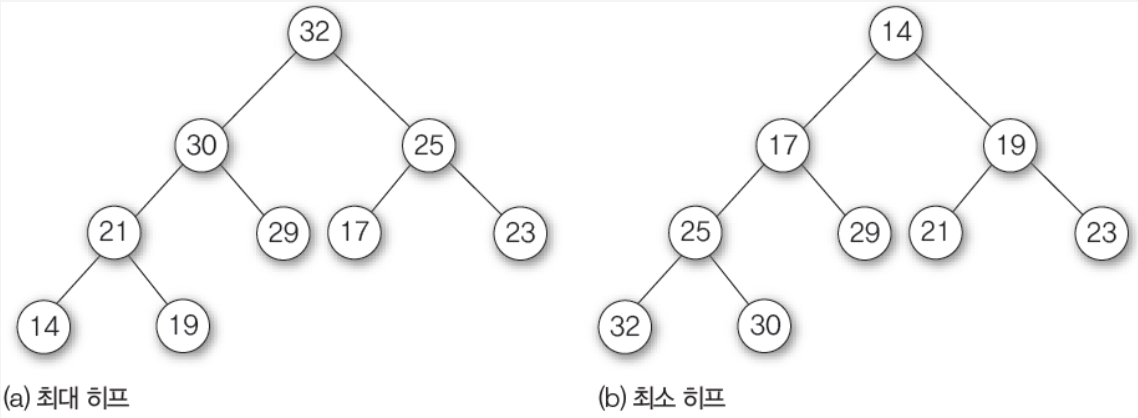
2 히프의 예



[이진 트리와 히프의 관계]

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

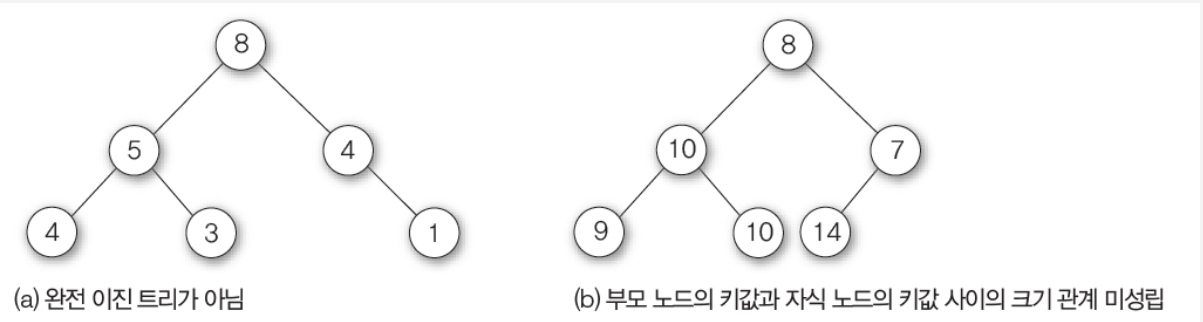
2 히프의 예



(a) 최대 히프 (b) 최소 히프

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 히프가 아닌 이진 트리의 예



(a) 완전 이진 트리가 아님

(b) 부모 노드의 키값과 자식 노드의 키값 사이의 크기 관계 미성립

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

4 히프의 추상 자료형

ADT 7-2 히프의 추상 자료형

```
ADT Heap
데이터 : 원소 n개로 구성된 완전 이진 트리로서 각 노드의 키값은 자식 노드의 키값보다 크거나 같다
        (부모 노드의 키값 ≥ 자식 노드의 키값).
연산 :
    heap∈Heap; item∈Element;

    // 공백 히프를 생성하는 연산
    createHeap() ::= create an empty heap;

    // 히프가 공백인지 검사하는 연산
    isEmpty(heap) ::= if (heap is empty) then return true;
                      else return false;

    // 히프의 적당한 위치에 원소(item)를 삽입하는 연산
    insertHeap(heap, item) ::= insert item into heap;
    // 히프에서 키값이 가장 큰 원소를 삭제하고 반환하는 연산
    deleteHeap(heap) ::= if (isEmpty(heap)) then return error;
                         else {
                           item ← 히프에서 가장 큰 원소;
                           remove {히프에서 가장 큰 원소};
                           return item;
                         }
End Heap()
```

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

2 히프의 삽입, 삭제 연산

1 히프의 삽입 연산

- 1단계 : 완전 이진 트리의 조건이 만족하도록 다음 자리를 확장
  - 노드가 n개인 완전 이진 트리에서 다음 노드의 확장 자리는 n+1번의 노드
  - n+1번 자리에 노드를 확장하고, 그 자리에 삽입할 원소를 임시 저장

1 히프의 삽입 연산

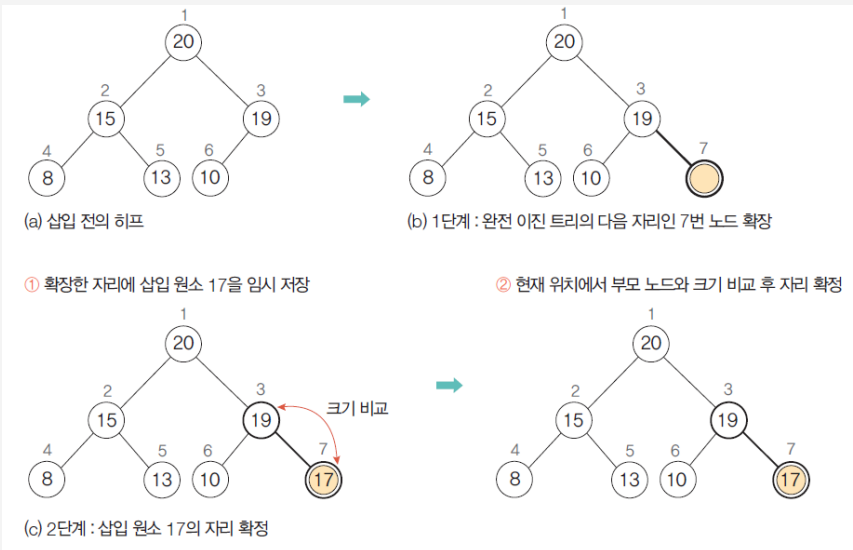
- 2단계 : 부모 노드와 크기 조건이 만족하도록
  삽입 원소의 위치를 찾음
  - 현재 위치에서 부모노드와 비교하여 크기 관계 확인
  - {현재 부모노드의 키값 ≥ 삽입 원소의 키값}의
    관계가 성립하지 않으면, 현재 부모노드의
    원소와 삽입 원소의 자리를 서로 바꿈

1 히프의 삽입 연산

- 히프에서의 삽입 연산 예1) 17을 삽입하는 경우
  - 노드를 확장하여 임시로 저장한 위치에서의 부모 노드와 크기를 비교하여 히프의 크기관계가 성립하므로, 현재 위치를 삽입 원소의 자리로 확정

1 히프의 삽입 연산

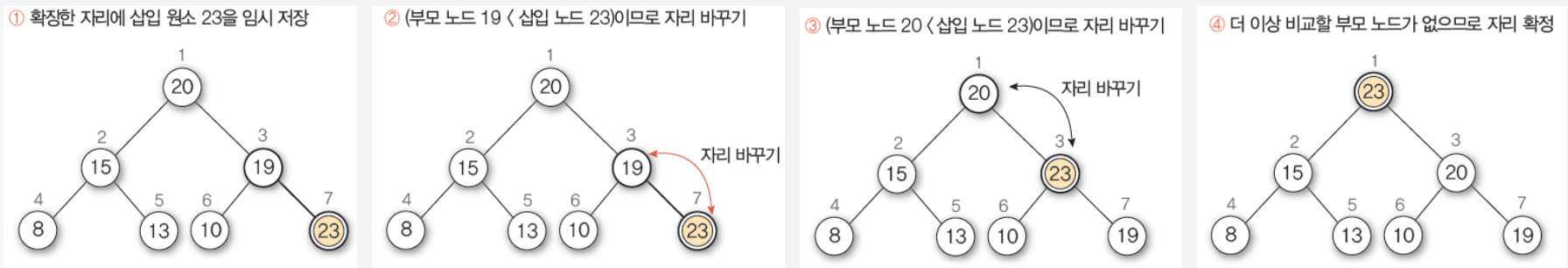
예1) 17을 삽입하는 경우



※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

1 히프의 삽입 연산

▶ 예2) 23을 삽입하는 경우



※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

1 히프의 삽입 연산

히프에서의 삽입 연산 알고리즘

알고리즘 7-11 최대 히프의 노드 삽입

```
insertHeap(heap, item)
     if (n = heapSize) then heapFull();
   ❶ n ←n + 1;
   ❷ for (i ← n; ;) do {
        if (i = 1) then exit;
      ❸ if (item ≤ heap[⌊i/2⌋]) then exit;
      ❹ heap[i] ← heap[⌊i/2⌋];
      ❺ i ← ⌊i/2⌋;
       }
   ❻ heap[i] ←item;
end insertHeap()
```

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

1 히프의 삽입 연산

▶ 히프에서의 삽입 연산 알고리즘

① 현재 히프의 크기를 하나 증가시켜서
노드 위치를 확장
② 확장한 노드 번호가 임시 삽입 위치 i가 됨
③ 삽입할 원소 item과 부모 노드 heap[⌊i/2⌋]를
비교하여 부모 노드보다 작거나 같으면 임시 삽입
위치 i를 삽입 원소의 위치로 확정, ⑥을 수행

1 히프의 삽입 연산

▶ 히프에서의 삽입 연산 알고리즘

④ 삽입할 원소 item이 부모 노드보다 크면,
부모 노드와 자식 노드의 자리를 맞바꿔 최대 히프의 관계를 만들어야 하므로 부모 노드 heap[⌊i/2⌋]의 원소를 현재의 임시 삽입 위치 heap[i]에 저장

⑤ ⌊i/2⌋ 를 임시 삽입 위치 i로 하여 ②~⑤를 반복하면서 item을 삽입할 위치를 찾음

⑥ 찾은 위치에 삽입할 원소 item을 저장하면 최대 히프의 재구성 작업이 완성되므로 삽입 연산을 종료

2 히프의 삭제 연산

- 히프에서는 루트 노드의 원소만을 삭제 할 수 있음

① 1단계 : 루트 노드의 원소를 삭제하여 반환
② 2단계 : 원소의 개수가 n-1개로 줄었으므로,
노드의 수가 n-1인 완전 이진 트리로 조정

- 노드가 n개인 완전 이진 트리에서 노드 수 n-1개의 완전 이진 트리가 되기 위해서 마지막 노드, 즉 n번 노드를 삭제
- 삭제된 n번 노드에 있던 원소는 비어있는 루트노드에 임시 저장

2 히프의 삭제 연산

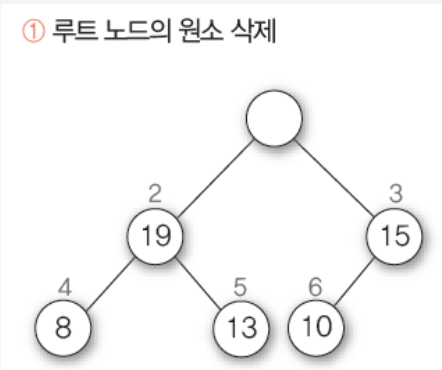
- 히프에서는 루트 노드의 원소만을 삭제 할 수 있음

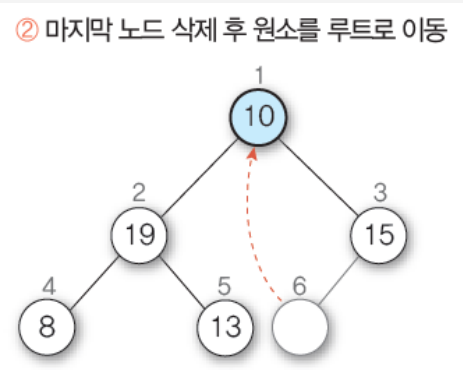
③ 3단계 : 완전 이진 트리 내에서 루트에 임시 저장된 원소의 제자리를 찾음

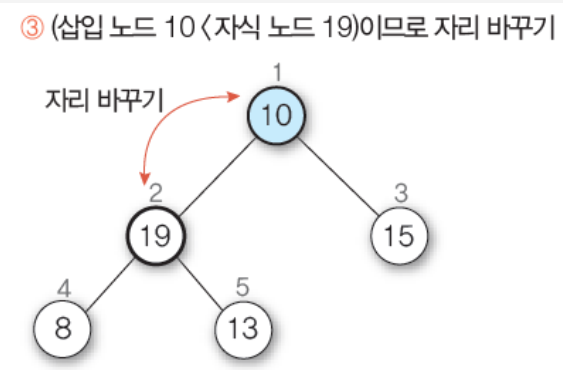
- 현재 위치에서 자식노드와 비교하여 크기 관계를 확인
- {임시 저장 원소의 키값 ≥ 현재 자식노드의 키값 } 의 관계가 성립하지 않으면, 현재 자식노드의 원소와 임시 저장 원소의 자리를 서로 바꿈

2 히프의 삭제 연산

히프에서의 삭제 연산 예



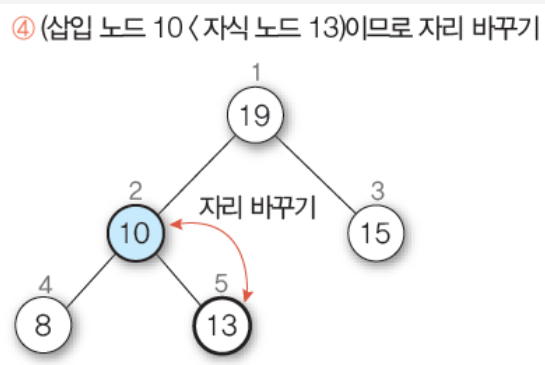


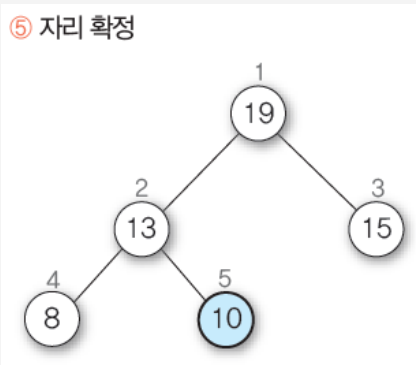


※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

2 히프의 삭제 연산

히프에서의 삭제 연산 예





※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

2 히프의 삭제 연산

알고리즘 7-12 최대 히프의 노드 삭제

```
deleteHeap(heap)
      if (n = 0) then return error;
   ❶ item ← heap[1];
   ❷ temp ← heap[n];
   ❸ n ← n - 1;
   ❹ i ← 1;
     j ← 2;
     while (j ≤ n) do {
❺      if (j < n) then
         if (heap[j] < heap[j + 1]) then j ← j + 1;
       if (temp ≥ heap[j]) then exit;
       heap[i] ← heap[j];
❻      i ← j;
       j ← j * 2;
     }
   ❼ heap[i] ← temp;
   ❽ return item;
end deleteHeap()
```

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

2 히프의 삭제 연산

① 루트 노드 heap[1]을 변수 item에 저장
② 마지막 노드의 원소 heap[n]을 변수 temp에 임시로 저장
③ 마지막 노드를 삭제하였으므로 히프 배열의 원소 개수가 하나 감소
④ 마지막 노드의 원소였던 temp의 임시 저장 위치 i는 루트 노드의 자리인 1번이 됨

2 히프의 삭제 연산

⑤ 현재 저장 위치에서 왼쪽 자식 노드 heap[j]와 오른쪽 자식 노드 heap[j+1]이 있을 때, 키값이 큰 자식 노드의 키값과 temp를 비교하여 temp가 크거나 같으면 현재의 임시 저장 위치를 temp 자리로 확정하고, ⑦을 수행

⑥ temp가 자식 노드 heap[j]보다 작으면 자식 노드와 자리를 바꾸고 ⑤~⑥을 반복하면서 temp의 자리를 찾음

2 히프의 삭제 연산

⑦ 찾은 위치에 temp를 저장하여 최대 히프의 재구성 작업을 완성
⑧ 처음에 삭제된 루트 노드를 저장한 변수 item을 반환, 삭제 연산을 종료

3 순차자료구조를 이용한 히프의 구현

3 | 순차자료구조를 이용한 히프의 구현

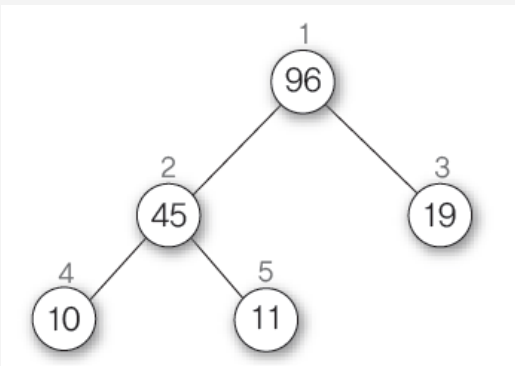
1 히프의 구현

- 1차원 배열의 순차 자료구조를 이용하면 인덱스 관계를 이용하여 부모 노드를 찾기가 쉬움
- 히프를 순차 자료구조로 표현한 예

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 공백 히프에 원소 다섯 개(10, 45, 19,11, 96)를 삽입하여 최대 히프를 구성



※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

1 히프의 구현

▶ 최대 히프의 알고리즘을 구현한 프로그램

```
#define MAX_ELEMENT 100
// 히프에 대한 1차원 배열과 히프 원소의 개수를 구조체로 묶어서 선언
06 typedef struct {
07          int heap[MAX_ELEMENT];
08          int heap_size;    } heapType;

12 heapType* createHeap() {   // 공백 히프를 생성하는 연산
13          heapType *h = (heapType*)malloc(sizeof(heapType));
14          h->heap_size = 0;
15          return h;    }
```

1 히프의 구현

최대 히프의 알고리즘을 구현한 프로그램

```
// 히프에 item을 삽입하는 연산
19 void insertHeap(heapType *h, int item) {
20        int  i;
21        h->heap_size = h->heap_size + 1;
22        i = h->heap_size;
23        while ((i != 1) && (item > h->heap[i / 2])) {
24                  h->heap[i] = h->heap[i / 2];
25                  i /= 2;
26        }
27        h->heap[i] = item;    }
```

1 히프의 구현

▶ 최대 히프의 알고리즘을 구현한 프로그램

```
// 히프의 루트를 삭제하여 반환하는 연산(1)
31 int deleteHeap(heapType *h) {
32          int parent, child;
33          int item, temp;
34          item = h->heap[1];
35          temp = h->heap[h->heap_size];
36          h->heap_size = h->heap_size - 1;
37          parent = 1;
38          child = 2;
}
```

1 히프의 구현

최대 히프의 알고리즘을 구현한 프로그램

```
// 히프의 루트를 삭제하여 반환하는 연산(2)
31 int deleteHeap(heapType *h) {
39        while (child <= h->heap_size) {
40                if ((child < h->heap_size) && (h->heap[child]) < h->heap[child + 1])
41                        child++;
42                if (temp >= h->heap[child]) break;
43                else {      h->heap[parent] = h->heap[child];
44                        parent = child;   child = child * 2;     }
45        }  h->heap[parent] = temp;
47        return item;   }
```

1 히프의 구현

▶ 최대 히프의 알고리즘을 구현한 프로그램

```
61  void main() {
62          int i, n, item;
63          heapType *heap = createHeap();
64          insertHeap(heap, 10);       insertHeap(heap, 45);
65          insertHeap(heap, 19);       insertHeap(heap, 11);
66          insertHeap(heap, 96);
67          n = heap->heap_size;
68          for (i = 1; i <= n; i++) {   item = deleteHeap(heap);
69                                       printf("₩n delete : [%d] ", item);    }
70  }
```

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 19~27행의 insertHeap()은 히프에 item값을 삽입하는 연산을 수행
  - 31~47행의 deleteHeap()은 히프의 루트 노드를 삭제하고 나머지 노드들을 히프로 재구성한 후에 삭제한 루트노드를 반환하는 연산을 수행

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 63~66행은 공백 히프를 생성한 후 원소를 하나씩 삽입
    - ✓ 64행 insertHeap(heap, 10);
      : 공백 히프에 원소 10을 삽입
      : 원소 10은 히프의 루트 노드가 됨

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 63~66행은 공백 히프를 생성한 후 원소를 하나씩 삽입
    - ✓ 64행 insertHeap(heap, 45);
      : 히프에 원소 45를 삽입
      : 2번 노드를 확장하고 삽입
      : 삽입 노드 45가 부모 노드 10보다 크기가 크므로 부모 노드와 자리를 맞바꿈

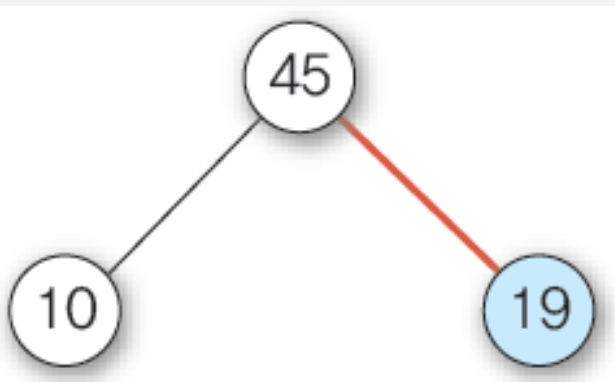
※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 63~66행은 공백 히프를 생성한 후 원소를 하나씩 삽입
    - ✓ 65행 insertHeap(heap, 19);
      : 히프에 원소 19를 삽입
      : 3번 노드를 확장하고 삽입한 후에 부모 노드와 크기를 비교
      : 삽입 노드 19는 부모 노드 45보다 작으므로 현재 위치를 확정

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 63~66행은 공백 히프를 생성한 후 원소를 하나씩 삽입
    - ✓ 65행



※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 63~66행은 공백 히프를 생성한 후 원소를 하나씩 삽입
    - ✓ 65행 insertHeap(heap, 11);
      : 히프에 원소 11을 삽입
      : 4번 노드를 확장하고 삽입한 후에
      부모 노드와 크기를 비교
      : 삽입 노드 11은 부모 노드 10보다 크므로
      부모 노드와 자리를 맞바꿈
      : 다시 현재 위치에서 부모 노드와 크기를 비교
      : 삽입 노드 11은 부모 노드 45보다 작으므로
      현재 위치를 확정

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 63~66행은 공백 히프를 생성한 후 원소를 하나씩 삽입
    - ✓ 65행

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

1 히프의 구현

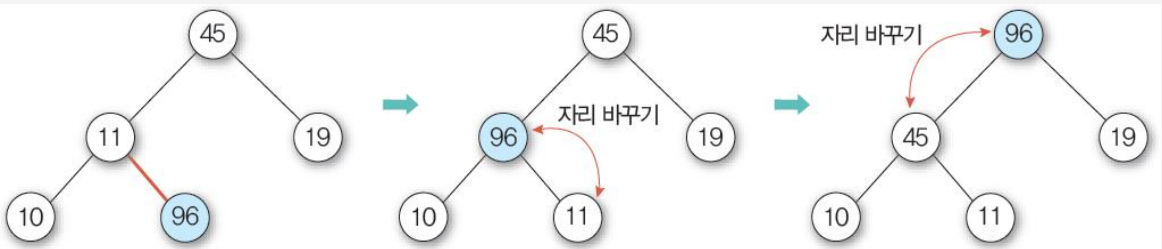
- 최대 히프의 알고리즘을 구현한 프로그램
  - 63~66행은 공백 히프를 생성한 후 원소를 하나씩 삽입
    - ✓ 66행 insertHeap(heap, 96);
      : 히프에 원소 96을 삽입
      : 5번 노드를 확장하고 삽입한 후에
      부모 노드와 크기를 비교
      : 삽입 노드 96은 부모 노드 11보다 크므로
      부모 노드와 자리를 맞바꿈
      : 새로운 현재 위치에서 다시 부모 노드와
      크기를 비교

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 63~66행은 공백 히프를 생성한 후 원소를 하나씩 삽입
    - ✓ 66행
      : 삽입 노드 96은 현재의 부모 노드 45보다 크므로 부모 노드와 자리를 맞바꿈
      : 새로운 현재 위치가 루트여서 더 이상 비교할 부모 노드가 없으므로 현재 위치를 확정

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 63~66행은 공백 히프를 생성한 후 원소를 하나씩 삽입
    - ✓ 66행



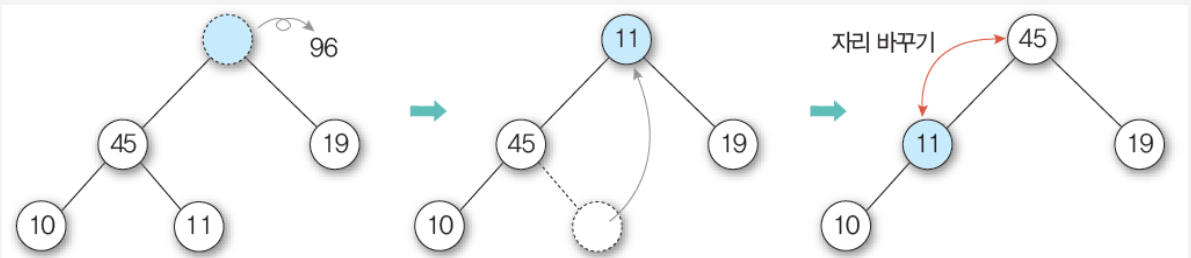
※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 68~69행은 히프의 원소 개수만큼 삭제 연산을 반복 수행하면서 삭제된 원소를 출력
    - ✓ i = 1일 때
      : 루트 노드를 삭제하고 나머지 원소들에 대해 히프를 재구성한 후에, 삭제된 루트 노드의 원소 96을 출력
      : 루트 노드의 원소를 삭제하여 원소가 네 개로 줄었으므로, 5번 노드의 자리를 삭제하고 5번 자리에 있던 원소 11은 루트 노드 자리에 임시로 저장

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 68~69행은 히프의 원소 개수만큼 삭제 연산을 반복 수행하면서 삭제된 원소를 출력
    - ✓ i = 1일 때



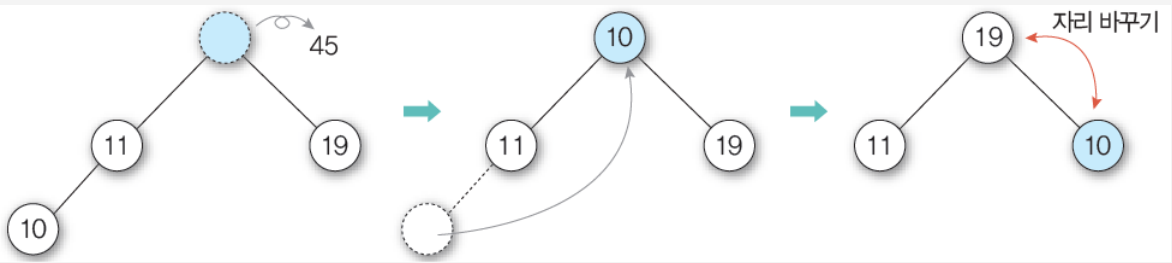
※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 68~69행은 히프의 원소 개수만큼 삭제 연산을 반복 수행하면서 삭제된 원소를 출력
    - ✓ i = 2일 때
      : 루트 노드를 삭제하고 나머지 원소들에 대해 히프를 재구성한 후에, 삭제된 루트 노드의 원소 45를 출력
      : 루트 노드의 원소를 삭제하여 원소가 세 개로 줄었으므로, 4번 노드의 자리를 삭제하고 4번 자리에 있던 원소 10은 루트 노드 자리에 임시로 저장

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 68~69행은 히프의 원소 개수만큼 삭제 연산을 반복 수행하면서 삭제된 원소를 출력
    - ✓ i = 2일 때



※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

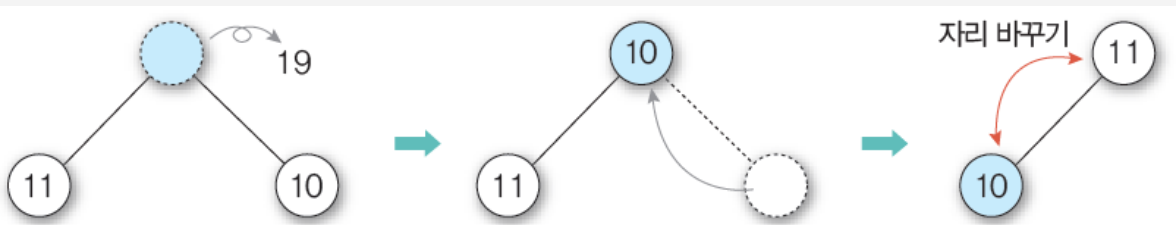
1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 68~69행은 히프의 원소 개수만큼 삭제 연산을 반복 수행하면서 삭제된 원소를 출력
    - ✓ i = 3일 때
      : 루트 노드를 삭제하고 나머지 원소들에 대해 히프를 재구성한 후에, 삭제된 루트 노드의 원소 19를 출력
      : 루트 노드의 원소를 삭제하여 원소가 두 개로 줄었으므로, 3번 노드의 자리를 삭제하고 3번 자리에 있던 원소 10은 루트 노드 자리에 임시로 저장

3 | 순차자료구조를 이용한 히프의 구현

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 68~69행은 히프의 원소 개수만큼 삭제 연산을 반복 수행하면서 삭제된 원소를 출력
    - ✓ i = 3일 때



※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 68~69행은 히프의 원소 개수만큼 삭제 연산을 반복 수행하면서 삭제된 원소를 출력
    - ✓ i = 4일 때
      : 루트 노드를 삭제하고 나머지 원소들에 대해 히프를 재구성한 후에, 삭제된 루트 노드의 원소 11을 출력
      : 루트 노드의 원소를 삭제하여 원소가 한 개로 줄었으므로, 2번 노드의 자리를 삭제하고 2번 자리에 있던 원소 10은 루트 노드 자리에 임시로 저장

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 68~69행은 히프의 원소 개수만큼 삭제 연산을 반복 수행하면서 삭제된 원소를 출력
    - ✓ i = 4일 때
      : 원소 10의 현재 위치가 단말 노드로 비교할 자식 노드가 없으므로 현재 위치를 확정

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

1 히프의 구현

- 최대 히프의 알고리즘을 구현한 프로그램
  - 68~69행은 히프의 원소 개수만큼 삭제 연산을 반복 수행하면서 삭제된 원소를 출력
    - ✓ i = 5일 때
      : 루트 노드를 삭제하고 삭제된 루트 노드의 원소 10을 출력
      : 원소가 다섯 개인 히프에서 삭제 연산을 다섯 번 수행하였으므로 히프가 공백이 됨
      : f or 문의 반복 연산을 종료

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어