10 주차 실습

실습

구현 방법 (힙, heap)

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX_ELEMENT 200
typedef struct {
  int key;
} element;
typedef struct {
  element heap[MAX_ELEMENT];
  int heap_size;
} HeapType;

HeapType* create(){
  return
(HeapType*)malloc(sizeof(HeapType));
}

void init(HeapType* h){
  h->heap_size = 0;
}
```

```c
void insert_max_heap(HeapType* h, element item){
  int i;
  i = ++(h->heap_size);
  while ((i != 1) && (item.key > h->heap[i / 2].key)) {
  h->heap[i] = h->heap[i / 2];
  i /= 2;
  }
  h->heap[i] = item;
}
element delete_max_heap(HeapType* h){
  int parent, child;
  element item, temp;
  item = h->heap[1];
  temp = h->heap[(h->heap_size)--];
  parent = 1;
  child = 2;
  while (child <= h->heap_size) {
    if ((child < h->heap_size) &&
      (h->heap[child].key) < h->heap[child + 1].key)
      child++;
    if (temp.key >= h->heap[child].key) break;
    h->heap[parent] = h->heap[child];
    parent = child;
    child *= 2;
  }
  h->heap[parent] = temp;
  return item;
}
```

실습

구현 방법 (힙, heap)

```
int main(void){
  element e1 = { 10 }, e2 = { 5 }, e3 = { 30 };
  element e4, e5, e6;
  HeapType* heap;

  heap = create();
  init(heap);

  insert_max_heap(heap, e1);
  insert_max_heap(heap, e2);
  insert_max_heap(heap, e3);

  e4 = delete_max_heap(heap);
  printf("< %d > ", e4.key);
  e5 = delete_max_heap(heap);
  printf("< %d > ", e5.key);
  e6 = delete_max_heap(heap);
  printf("< %d > ₩n", e6.key);

  free(heap);
return 0;
}
```

실습

허프만 코드

- 각 글자의 빈도가 알려져 있는 메시지의 내용을 압축하는데 사용
- 텍스트 압축을 위해 널리 사용되는 방법
- 출현 빈도가 낮은 문자는 많은 비트의 코드로 변환
- 출현 빈도가 많은 문자는 적은 비트의 코드로 변환

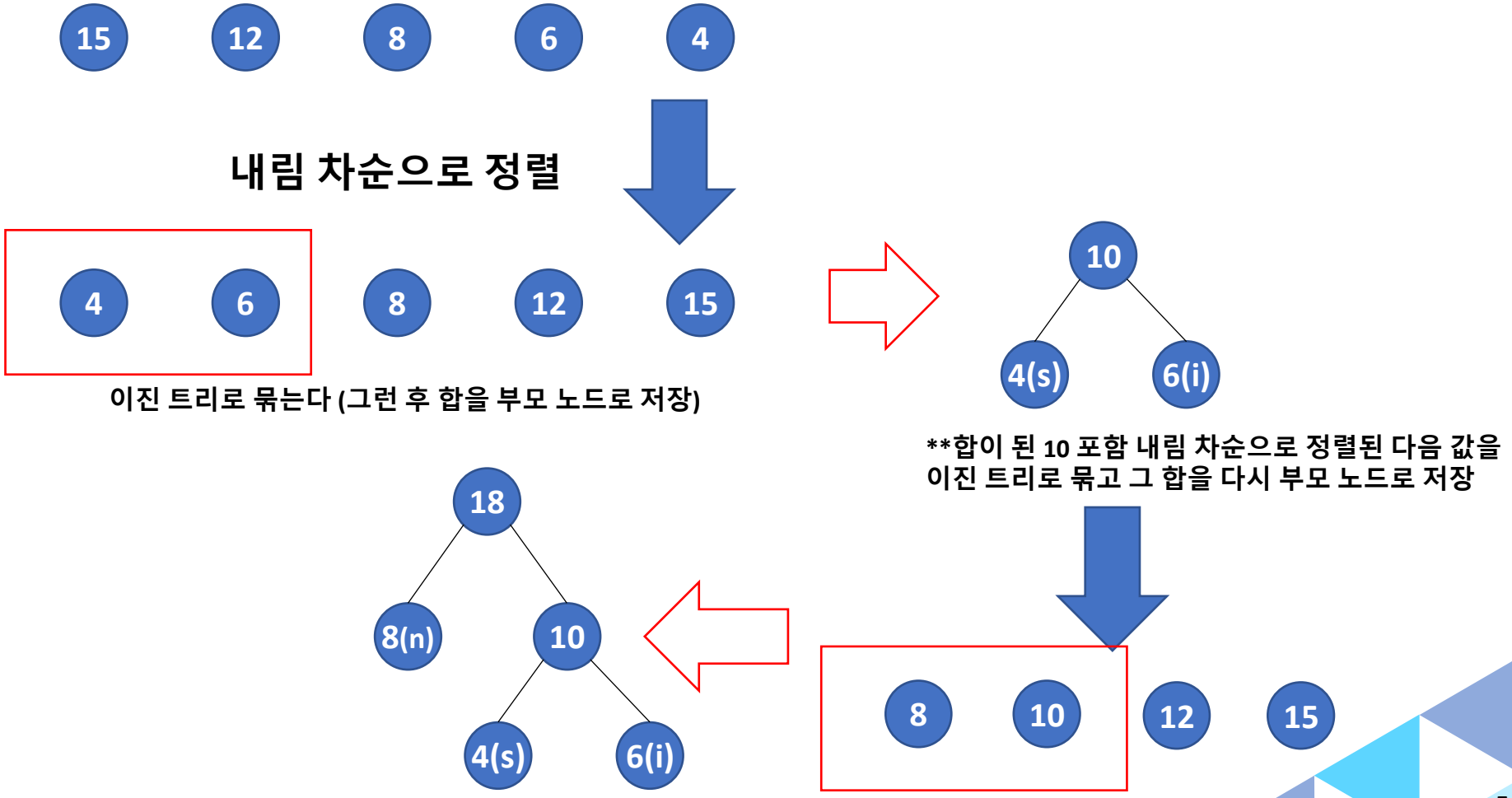
IF) 텍스트가 e, t, n, I, s의 5개의 글자로 만 이루어졌다고 가정하고 각 글자수 빈도수는 다음과 같음

글자	빈도수
e	15
t	12
n	8
i	6
s	4

비트 코드	비트 수
00	30
01	24
11	16
100	18
101	12

실습

허프만 코드


15
12
8
6
4
내림 차순으로 정렬
4
6
8
12
15
이진 트리로 묶는다 (그런 후 합을 부모 노드로 저장)
10
4(s)
6(i)
**합이 된 10 포함 내림 차순으로 정렬된 다음 값을
이진 트리로 묶고 그 합을 다시 부모 노드로 저장
8
10
12
15
18
8(n)
10
4(s)
6(i)

실습

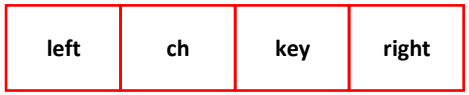
허프만 코드

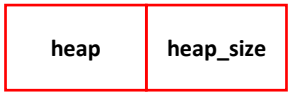
글자	빈도수
e	15
t	12
n	8
i	6
s	4

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_ELEMENT 200

typedef struct TreeNode {
  int key;
  char ch;
  struct TreeNode* left;
  struct TreeNode* right;
} element;

typedef struct {
  element* heap[MAX_ELEMENT];
  int heap_size;
} HeapType;
```





실습

구현 방법 (허프만 코드)

```
HeapType* create() {
return (HeapType*)malloc(sizeof(HeapType));
}

void init(HeapType* h) {
  h->heap_size = 0;
}

void insert_min_heap(HeapType* h, element* item) {

}
element* delete_min_heap(HeapType* h) {

}
void print_codes(element* root, int codes[], int top) {
  if (root->left) {
    codes[top] = 1;
    print_codes(root->left, codes, top + 1);
  }
  if (root->right) {
    codes[top] = 0;
    print_codes(root->right, codes, top + 1);
  }
  if (is_leaf(root)) {
    printf("%c: ", root->ch);
    print_array(codes, top);
  }
}
```

```
element* make_tree(element* left, element* right)
{
  element* node =
(element*)malloc(sizeof(element));
  node->left = left;
  node->right = right;
  return node;
}

void destroy_tree(element* root) {
  if (root == NULL) return;
  destroy_tree(root->left);
  destroy_tree(root->right);
  free(root);
}

int is_leaf(element* root) {
  return !(root->left) && !(root->right);
}
void print_array(int codes[], int n) {
  for (int i = 0; i < n; i++)
    printf("%d", codes[i]);
  printf("₩n");
}
```

실습

구현 방법 (허프만 코드)

```
void huffman_tree(int freq[], char ch_list[], int n){
   int i;           Treenode *node, *x;         HeapType * heap;
   element e, e1, e2;     int codes[100];         int top = 0;
   ...
  element* e = delete_min_heap(heap);
  int top = 0;
  int codes[100];
  print_codes(e, codes, top);
}
int main(void){
  char ch_list[] = { 's', 'i', 'n', 't', 'e' };
  int freq[] = { 4, 6, 8, 12, 15 };
  huffman_tree(freq, ch_list, 5);
  return 0;
}
```

1. 힙을 생성하고 초기화 한다.
2. For문을 돌면서 트리 구조체(자식 노드가 없는)를 생성하고 key 값과 ch 값을 넣어준다. 이 때, 힙도 추가해준다.
3. For문을 다시 돌면서 힙 삭제를 통해 최소 값 2개를 뽑아 key(가중치) 를 더해주고 이를 힙에 추가
4. 한 개 남은 힙 데이터를 뽑으면 최종 트리가 나오고 이를 code 형태 출력

실습

구현 방법 (허프만 코드)

```
char* string = "This study is toward interactive dynamic mapping on web
based on open source.₩
 Among available interactive mapping of open source libraries, D3.js was
chosen.₩
 It is a JavaScript library with capability to bind arbitrary data and
provides ₩
 open source mapping framework. Analysis of geo-statistical data is
designed using R,₩
 a package and programming language for statistical data analysis. ₩
 This work implemented an integrated user interface with these separate
frameworks, ₩
 and provides a mobile web app application for client sides.";
```

이러한 문자열이 있다고 가정할 때 위 허프만 코드를 적용하기 위해서 할 수 있는 방법은??