

자료구조 실습과제 13

- 솔루션 및 프로젝트 명칭 : Proj_13_이름이니셜
- 제출방법 : 아래 문제를 해결하기 위한 프로그램을 구현한 후 컴파일 및 실행한 후, 오류가 없으면 메뉴에서 솔루션 정리를 수행한 후 윈도우 탐색기에서 솔루션 폴더를 찾아 압축하여 E-class에 올림

각 문제를 테스트하기 위하여 아래와 같이 구현하시오.

```
#define _CRT_SECURE_NO_WARNINGS

#define PROB 1 // 각각의 문제를 구현하고 해당 문제 번호를 변경하여 테스트

#if PROB == 1
// 1번 문제를 해결하기 위한 프로그램 작성

#elif PROB == 2
// 2번 문제를 해결하기 위한 프로그램 작성

#endif
```

문제 1) 힙을 이용한 우선순위 큐를 만들어 힙정렬을 구현하고 선택 정렬과 정렬 시간 비교

우선순위 큐를 구현하는 방법으로 힙을 활용하여 오름차순으로 정수형 데이터를 정렬하는 힙정렬을 구현하시오. 이론적으로 계산량이 $O(n \log n)$ 이 된다. 반면 선택 정렬은 $O(n^2)$ 의 시간 복잡도를 가진다. 두 알고리즘을 구현하여 아래의 main() 함수 내의 코드를 이용하여 실행 결과를 보이시오. 실행 예와 같이 데이터 개수가 많은 경우에 대해서는 원데이터와 정렬된 데이터를 출력하지 말고, 소수의 데이터를 이용한 경우 실행 결과를 출력하여 정렬이 정상적으로 이루어짐을 보이시오.

오름차순으로 정렬하는 선택정렬은 정렬되지 않은 데이터에서 가장 작은 데이터를 찾아 정렬된 데이터의 마지막에 추가하는 방식으로 정렬하는 방법이다.

```
selection_sort(A, n)

for i ← 0 to n-2 do
    least ← A[i], A[i+1], ..., A[n-1] 중에서 가장 작은 값의 인덱스;
    A[i]와 A[least]의 교환;
    i++;
```

```

int main(void)
{
    element* list, *sorted_list; //원데이터와 정렬된 데이터 각각 저장용
    int i, s_time, e_time;

    srand(100);
    list = (element*)malloc(sizeof(element) * SIZE);
    sorted_list = (element*)malloc(sizeof(element) * SIZE);

    for (i = 0; i < SIZE; i++) list[i] = rand();

    printf("정수형 데이터 %d개 정렬 \n", SIZE);
    if (SIZE < 100) {
        for (int i = 0; i < SIZE; i++) printf("%d ", list[i]);
        printf("\n\n");
    }

    s_time = clock();
    heap_sort(list, sorted_list, SIZE);
    e_time = clock();
    printf("Heap Sort 걸린 시간 : %d msec\n", e_time - s_time);

    if (SIZE < 100) {
        for (int i = 0; i < SIZE; i++) printf("%d ", sorted_list[i]);
        printf("\n\n");
    }


    s_time = clock();
    selection_sort(list, sorted_list, SIZE);
    e_time = clock();
    printf("Selection Sort 걸린 시간 : %d msec\n", e_time - s_time);

    if (SIZE < 100) {
        for (int i = 0; i < SIZE; i++) printf("%d ", sorted_list[i]);
        printf("\n");
    }

    return 0;
}

```

실행 예 : (데이터 개수가 많은 경우, SIZE를 100000으로 설정)

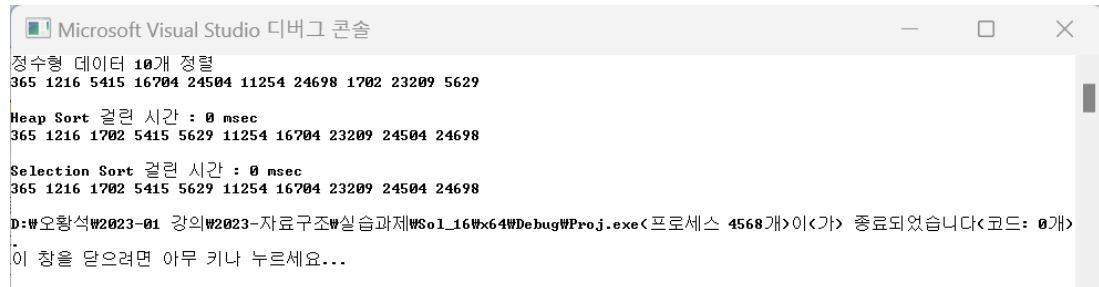


```

Microsoft Visual Studio 디버그 콘솔
정수형 데이터 100000개 정렬
Heap Sort 걸린 시간 : 17 msec
Selection Sort 걸린 시간 : 6552 msec
D:\오형석\2023-01 강의\2023-자료구조\실습과제\Sol_16\Debug\Proj.exe<프로세스 29632개>이<가> 종료되었습니다.<코드: 0>
>.
이 창을 닫으려면 아무 키나 누르세요...

```

실행 예 : (데이터 개수가 적은 경우 원데이터와 정렬된 데이터 출력, SIZE를 10으로 설정)



```
Microsoft Visual Studio 디버그 콘솔
정수형 데이터 10개 정렬
365 1216 5415 16704 24504 11254 24698 1702 23209 5629
Heap Sort 걸린 시간 : 0 msec
365 1216 1702 5415 5629 11254 16704 23209 24504 24698
Selection Sort 걸린 시간 : 0 msec
365 1216 1702 5415 5629 11254 16704 23209 24504 24698
D:\오형석\2023-01 강의\2023-자료구조\실습과제\Sol_16\Debug\Proj.exe <프로세스 4568개>이<가> 종료되었습니다<코드: 0개>
이 창을 닫으려면 아무 키나 누르세요...
```

문제 2) 힙을 이용한 우선순위 큐를 만들어 허프만코드 트리를 만들어 인코딩된 문자 디코딩 하기

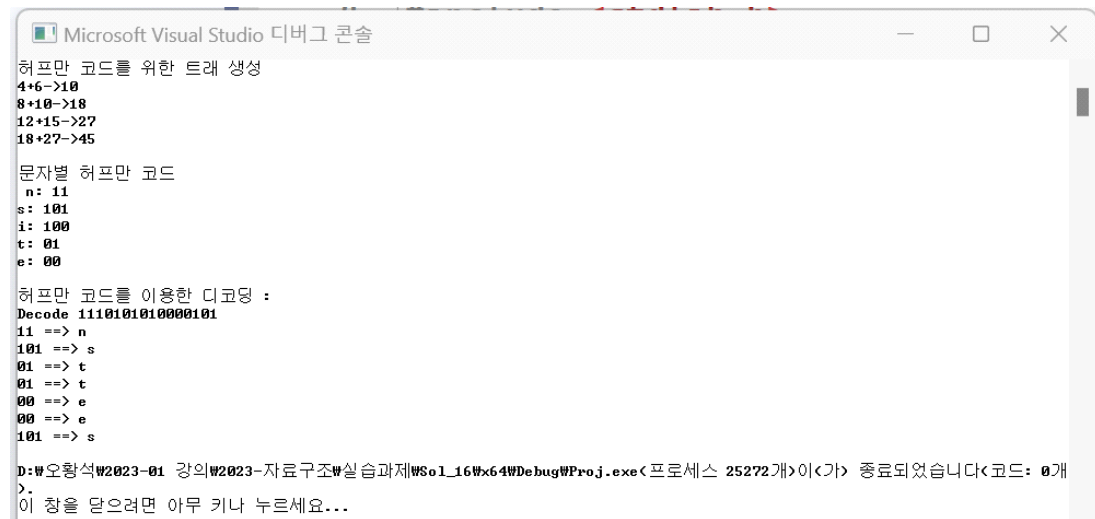
문자별로 발생하는 횟수를 이용하여 빈도가 많은 경우 짧은 코드를 할당하고, 빈도가 매우 적은 경우 긴 코드를 할당하여 전체 문서의 저장 용량을 줄이는 방법으로 허프만 코딩을 많이 활용한다. 교재의 허프만 코딩을 위한 코드 생성하는 과정에서 힙을 이용하는 방법을 활용하여 허프만 코딩을 위한 트리를 만들고, 이를 이용해서 들어오는 압축된 문자열을 디코딩하는 프로그램을 작성하시오. 아래에 기술된 구조체와 main 함수의 코드, 실행 결과를 참고하시오.

```
#define MAX_ELEMENT 200
typedef struct TreeNode {
    int weight;
    char ch;
    struct TreeNode* left;
    struct TreeNode* right;
} TreeNode;
typedef TreeNode* element;
typedef struct {
    element heap[MAX_ELEMENT];
    int heap_size;
} HeapType;
```

```
int main(void)
{
    char ch_list[] = { 's', 'i', 'n', 't', 'e' };
    int freq[] = { 4, 6, 8, 12, 15 };
    TreeNode* head;

    head = make_huffman_tree(freq, ch_list, 5);
    huffman_decode(head, "1110101010000101");
    destroy_tree(head);
    return 0;
}
```

실행 예 :



```
Microsoft Visual Studio 디버거 콘솔

허프만 코드를 위한 트래 생성
4*6->10
8*10->18
12*15->27
18*27->45

문자별 허프만 코드
n: 11
s: 101
i: 100
t: 01
e: 00

허프만 코드를 이용한 디코딩 :
Decode 1110101010000101
11 ==> n
101 ==> s
01 ==> t
01 ==> t
00 ==> e
00 ==> e
101 ==> s

D:\오황석\2023-01 강의\2023-자료구조\실습과제\So1_16\Debug\Proj.exe (프로세스 25272개)이<가> 종료되었습니다<코드: 0개>.
이 창을 닫으려면 아무 키나 누르세요...
```