

알고리즘 과제

Practice.03

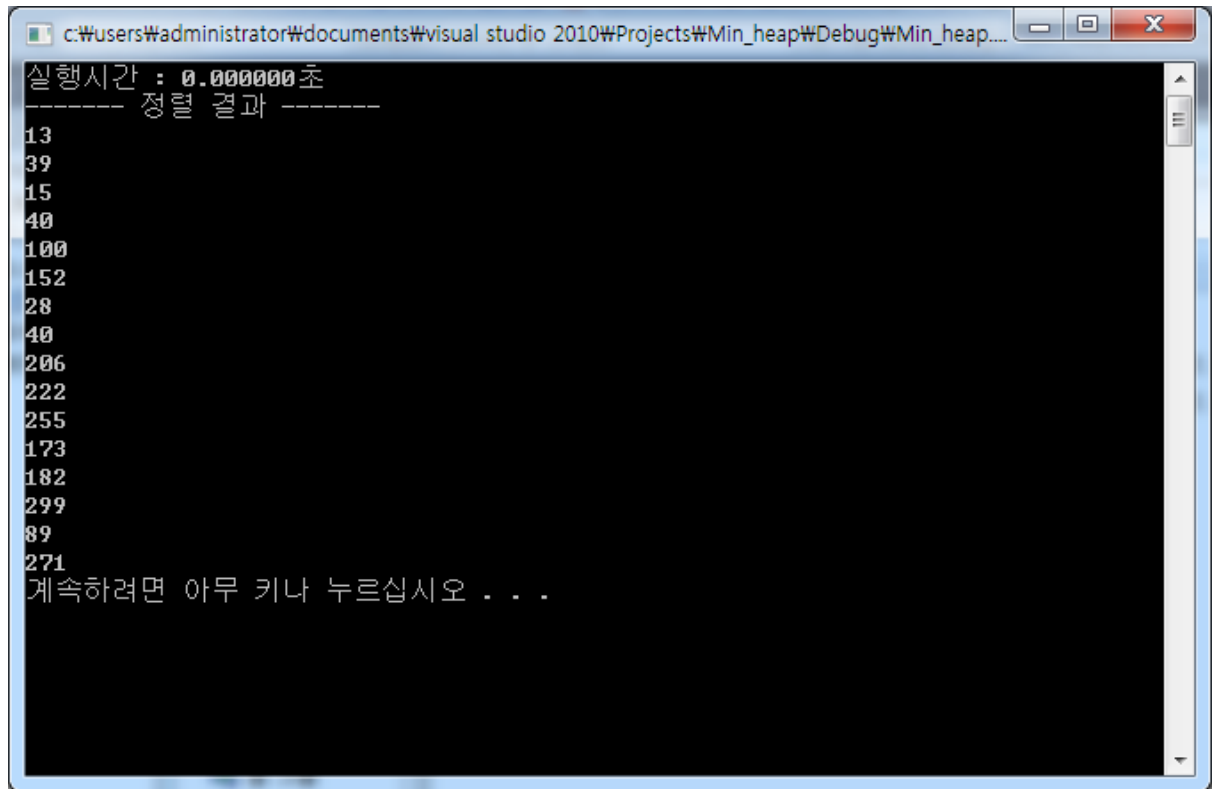
학번 : 201402432

이름 : 조디모데

Min-Heap Sort.

- Input_Small

- 출력 결과



```
c:\Users\Administrator\documents\visual studio 2010\Projects\Min_heap\Debug\Min_heap....
실행시간 : 0.000000초
----- 정렬 결과 -----
13
39
15
40
100
152
28
40
206
222
255
173
182
299
89
271
계속하려면 아무 키나 누르십시오 . . .
```

- 알고리즘 설명

- min 힙 tree 를 만들어 정렬 하는 방법

1. n 개의 노드에 대한 완전 이진 트리를 구성한다.
2. min 힙을 구성
3. 가장 끝의 원소부터 선택하여 부모 노드와 비교하여 더 작은 경우 swap 한다.
4. 2 와 3 을 반복한다.

- 컴파일 방법

- input 폴더를 다음 위치에 넣는다

- C:\Users\Administrator\Desktop\input\input_small.txt

● Code (.C)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <time.h>

/*      Swap node i and node j in array A
    parameter A : heap implemented as array
    parameter i, j : index of node in array A */
void swap(int A[], int i, int j) {
    int temp ;
    temp = A[i] ;
    A[i] = A[j] ;
    A[j] = temp ;
}

/*      function to do return parent of node i
    parameter i : index of node i */
int parent(int i) {
    return ((i-1)/2) ;
}

/*      function to do return left child of node i
    parameter i : index of node i
    parameter n : heap size */
int leftChild(int i, int n) {
    int child = (2*i + 1) ;
    if(child >= n)
        return -1 ;
    return child ;
}

/*      function to do return right child of node i
    parameter i : index of node i
    parameter n : heap size */
int rightChild(int i, int n) {
    int child = (2*i + 2) ;
    if(child >= n)
        return -1 ;
    return child ;
}

/*      function to do maintain the heap property
    parameter A : heap implemented as array
    parameter i : index of subtree root node
    parameter n : heap size */
void MinHeapify(int A[], int i, int n) {
    int left = leftChild(i, n) ;
    int right = rightChild(i, n) ;
    int index ;
    // 부기모-0&보-7-다4U 작U은 " ~ 경&0 j 우~i
    if(left <= n && A[((i-1)/2)] > A[i])
        index = ((i-1)/2) ;
    else
```

```

        index = i ;

    if(index != i){
        // 부모노드와보통노드작은노드경우 swap
        swap(A, i, index) ;
        //다음노드 heapify로 넘어가기
        MinHeapify(A ,index, n) ;
    }
}

/*    Build Min-Heap
parameter A : heap implemented as array
parameter n : heap size */
void buildMinHeap(int A[], int n){
    int i ;
    for(i = n ; i > 0 ; i--){
        MinHeapify(A ,i ,n) ;
    }
}

/*    Sort Min-heap A
parameter A : Min-heap implemented as array
parameter n : heap size */
void heapSort(int A[], int n) {
    int i ;
    for(i = n ; i > 0 ; i--){
        MinHeapify(A,i,n);
    }
}

// main function
int main(void)
{
    int i, num[17], n ;
    FILE *fps ;
    int temp = 0 ;
    clock_t start,end ;

    n = 16 ;

    // input의 숫자들을 i 배열에 num에 저장하기
    fps = fopen("C:\\Users\\Administrator\\Desktop\\input\\input_small.txt", "rt");
    for(i=0; i <= n ; i++){
        fscanf(fps, "%d", &temp) ;
        num[i]=temp ;
    }
    fclose(fps) ;

    // 정렬 시간 측정 시작
    start = clock() ;

    // 정렬 시간 측정
    for(i=0; i<sizeof(num)/sizeof(int); i++){
        buildMinHeap(num,i) ;
    }
}

```

```

// 정렬 후 시간저장
end = clock() ;

// 실행 시간 출력
printf("실행 시간 : %lf초\n", (end-start)/(double)1000) ;

// 정렬 결과 출력
printf("----- 정렬 결과 ----- \n") ;
temp = 0 ;
for(i=0 ; i<n ; i++){
    printf("%d\n", num[i]) ;
}

system("pause") ;

return 0;
}

```

Iterative MAX-HEAPIFY.

- Input50
 - 출력 결과



```
c:\users\administrator\documents\visual studio 2010\Projects\Iter_Max_heap\Debug\Iter_...
실행시간 : 0.000000초
----- 정렬 결과 -----
495
478
403
394
467
382
367
351
254
249
466
334
84
225
236
89
258
212
153
128
188
419
406
267
311
54
31
10
145
60
24
5
15
43
102
67
183
137
61
51
96
66
85
86
143
338
165
113
62
127
계속하려면 아무 키나 누르십시오 . . .
```

- 알고리즘 설명

기존의 heap 정렬과 같지만 **reculsive** 하게 함수를 실행하는 것이 아닌 **iterative** 하게 함수를 사용한다.

- 컴파일 방법

input 폴더를 바탕화면에 넣는다

C:\Users\WWAdministrator\Desktop\WWinput\WWinput50.txt

● Code (.C)

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

/*      Swap node i and node j in array A
    parameter A : heap implemented as array
    parameter i, j : index of node in array A */
void swap(int A[], int i, int j) {
    int temp ;
    temp = A[i] ;
    A[i] = A[j] ;
    A[j] = temp ;
}

/*      function to do return parent of node i
    parameter i : index of node i */
int parent(int i) {
    return ((i-1)/2) ;
}

/*      function to do return left child of node i
    parameter i : index of node i
    parameter n : heap size */
int leftChild(int i, int n) {
    int child = (2*i + 1) ;
    if(child >= n)
        return -1 ;
    return child ;
}

/*      function to do return right child of node i
    parameter i : index of node i
    parameter n : heap size */
int rightChild(int i, int n) {
    int child = (2*i + 2) ;
    if(child >= n)
        return -1 ;
    return child ;
}

/*      Build Max-Heap
    parameter A : heap implemented as array
    parameter n : heap size */
void buildMaxHeap(int A[], int n){
    int i, index, left, right ;
    for(i = n ; i >= 0 ; i--){
        left = leftChild(i, n) ;
        right = rightChild(i, n) ;
        // 왼쪽 자식 또는 오른쪽 자식 중 큰 쪽을 index로 지정
        if(left <= n && A[left] > A[i])
            index = left ;
    }
}

```



```

        else
            index = i ;
        // 오`A를`이 í쪽E 자U식`A이í 큰 í 경A`í 우`i
        if(right <= n && A[right] > A[i])
            index = right ;
        if(index != i){
            // 자U식`A의C 값A``이í 더`o 클 í 경A`í 우`i
            swap(A, i, index) ;
            //다`U음``õ heapify로`í 넘`N어`í값A í §
            i++ ;
            continue ;
        }
    }
}

// main function
int main(void)
{
    int i, num[51], n ;
    FILE *fps ;
    int temp = 0 ;
    clock_t start,end ;

    n = 50 ;

    // input50.txt의C 숫`y자U들`ie을 í í 배`e열` í @ num에``@ 저u장a하`는`A 부`기분```
    fps = fopen("C:\\Users\\Administrator\\Desktop\\input\\input50.txt","rt");
    for(i=0; i <= n ; i++){
        fscanf(fps,"%d",&temp) ;
        num[i]=temp ;
    }
    fclose(fps) ;

    // 정```렬`A 시`A작U 시`A간A í í 저u장a
    start = clock() ;

    // 정```렬`A하`는`A 부`기분```
    for(i=0; i<sizeof(num)/sizeof(int); i++){
        buildMaxHeap(num,i) ;
    }

    // 정```렬`A 후A 시`A간A í í 저u장a
    end = clock() ;

    // 실`C행a 시`A간A í í 출a력`A
    printf("실`C행a시`A간A í í : %f초E\\n",(end-start)/(double)1000) ;

    // 정```렬`A 결Aa과Aeu 출a력`A
    printf("----- 정```렬`A 결Aa과Aeu ----- \\n") ;
    temp = 0 ;

```

```
for(i=0 ; i<n ; i++){  
    printf( "%d\\n",num[i] ) ;  
}  
  
system( "pause" ) ;  
  
return 0;  
}
```

90402432 - 문제풀기.

← 1.

$$a. T(n) = 2T(n/2) + n^4$$

$$T(n) = \sum_{k=0}^{\log_2 n - 1} 2^k \left(\frac{n}{2^k}\right)^4 + O(n)$$

$$< \sum_{k=0}^{\log_2 n} \frac{n^4}{2^k} + O(n)$$

$$= \frac{1}{1 - \frac{1}{2}} n^4 + O(n)$$

$$= \frac{2}{1} n^4 + O(n)$$

$$= O(n^4)$$

$$b. T(n) = T(n/2) + n$$

$$T(n) = \sum_{k=0}^{\log_2 n} \left(\frac{n}{2^k}\right) + T(n)$$

$$T(n) = O(n)$$

$$c. T(n) = \frac{1}{2}T(n/2) + \frac{n^2}{2}$$

$$O(n^2), f(n) = O(n^{2+\epsilon})$$

$$\rightarrow \epsilon = 0$$

$$\therefore O(n^2 \log n)$$

$$d. T(n) = 7T(n/3) + n^2$$

$$= \sum_{k=0}^{\log_3 n - 1} 7^k \left(\frac{n}{3^k}\right)^2 + O(n^2 \log n)$$

$$< \sum_{k=0}^{\log_3 n} \left(\frac{7}{9}\right)^k n^2 + \dots$$

$$= O(n^2)$$

$$e. T(n) = 2T(n/2) + n^2$$

MT

$$a=2$$

$$b=2$$

$$f(n)=n^2$$

$$O(n^{\log_2 2}) \rightarrow E = \log_2 1 - 2$$

$$= O(n^{\log_2 2})$$

$$f. T(n) = 2T(n/4) + \sqrt{n}$$

MT

$$a=2$$

$$b=4$$

$$f(n)=\sqrt{n}$$

$$E=0$$

$$\Rightarrow O(\sqrt{n} \log n)$$

$$g. T(n) = T(n-2) + n^2$$

$$T(n-2) = T(n-4) + (n-2)^2$$

$$T(n) = T(n-4) + (n-2)^2 + n^2$$

$$= T(2) + n^2 + (n-2)^2 + \dots + 2^2$$

$$= \sum_{k=0}^{\frac{n-1}{2}} (n-2k)^2 + T(2)$$

$$= \sum_{k=0}^{\frac{n-1}{2}} (n^2 - 4nk + 4k^2) + T(2)$$

$$\Rightarrow \underline{\underline{O(n^3)}}$$

6.1-2

$$\sum_{i=0}^{h-1} 2^i = 2^0 + 2^1 + 2^2 + \dots + 2^{h-1}$$

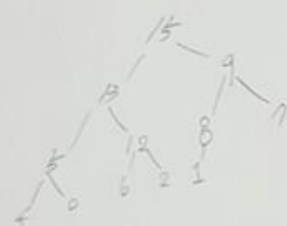
$$= \sum_{i=0}^{h-1} 2^i$$

$$= \frac{2^h - 1}{2 - 1}$$

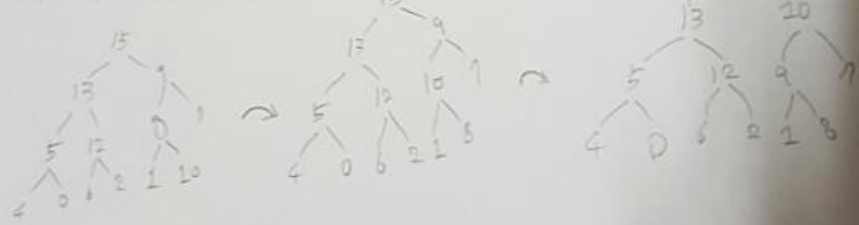
높이 h 가 있는 완전 이진 트리의 노드 수 $n = 2^h - 1$
 \rightarrow 노드 수 n 가 주어지면 높이 $h = \lceil \log_2(n+1) \rceil$

6.5-2

Heap A



Max-Heap-Insert(A, 10)

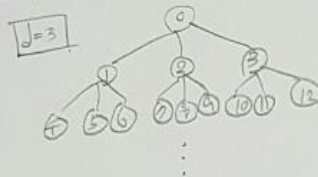


$\Rightarrow A[15, 13, 10, 5, 12, 9, 7, 4, 0, 6, 2, 1, 8]$

6-2.

(a)

가장 첫 원소를 부모노드로 사용하여 그 다음 노드 j 개가 그 자식노드가 된다.
 $j+1$ 다음 노드는 3번째 노드로 사용되는 식으로 배열을 표현한다.



(b)

$$\lfloor \log_2 N \rfloor$$

원리는 이진트리와 경우와 같다.