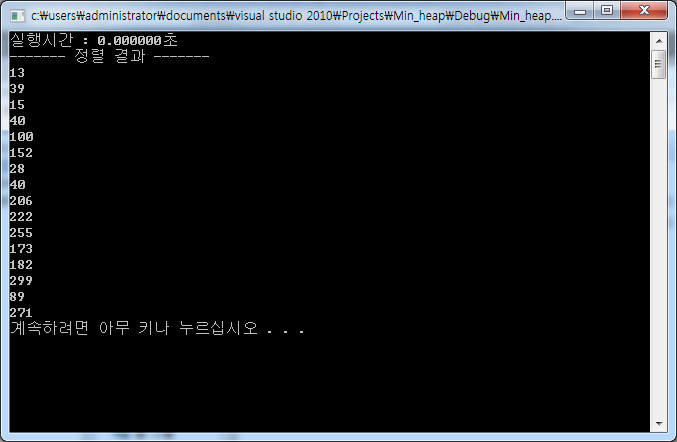
알고리즘 과제

Practice.03

**학번 : 201402432**

**이름 : 조디모데**

**Min-Heap Sort.**

* Input\_Small   
  - 출력 결과  
  
* 알고리즘 설명  
   min[힙](https://ko.wikipedia.org/wiki/%ED%9E%99_(%EC%9E%90%EB%A3%8C_%EA%B5%AC%EC%A1%B0)) tree를 만들어 정렬 하는 방법

1. n개의 노드에 대한 [완전 이진 트리](https://ko.wikipedia.org/wiki/%EC%99%84%EC%A0%84_%EC%9D%B4%EC%A7%84_%ED%8A%B8%EB%A6%AC)를 구성한다.
2. min힙을 구성
3. 가장 끝의 원소부터 선택하여 부모 노드와 비교하여 더 작은 경우 swap한다.
4. 2와 3을 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 ;

// 부¬I모¬©£보¬¢¬다¥U 작U은¨¬ 경Æ©¡우¯i

if(left <= n && A[((i-1)/2)] > A[i])

index = ((i-1)/2) ;

else

index = i ;

if(index != i){

// 부¬I모¬©£보¬¢¬다¥U 작U은¨¬ 경Æ©¡우¯i swap

swap(A, i, index) ;

//다¥U음¨ö heapify로¤I 넘øN어úi감Æ¡§

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 ;

// input의C 숫ùy자U들ìe을¡í 배öe열¯¡© num에¯¢® 저u장a하I는¥A 부¬I분¬¨¢

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) ;

// 정¢´렬¤A 시öA작U 시öA간Æ¡Ì 저u장a

start = clock() ;

// 정¢´렬¤A하I는¥A 부¬I분¬¨¢

for(i=0; i<sizeof(num)/sizeof(int); i++){

buildMinHeap(num,i) ;

}

// 정¢´렬¤A 후A 시öA간Æ¡Ì 저u장a

end = clock() ;

// 실öC행a 시öA간Æ¡Ì 출a력¤A

printf("실öC행a시öA간Æ¡Ì : %lf초E\n",(end-start)/(double)1000) ;

// 정¢´렬¤A 결Æa과Æu 출a력¤A

printf("------- 정¢´렬¤A 결Æa과Æu ------- \n") ;

temp = 0 ;

for(i=0 ; i<n ; i++){

printf("%d\n",num[i]) ;

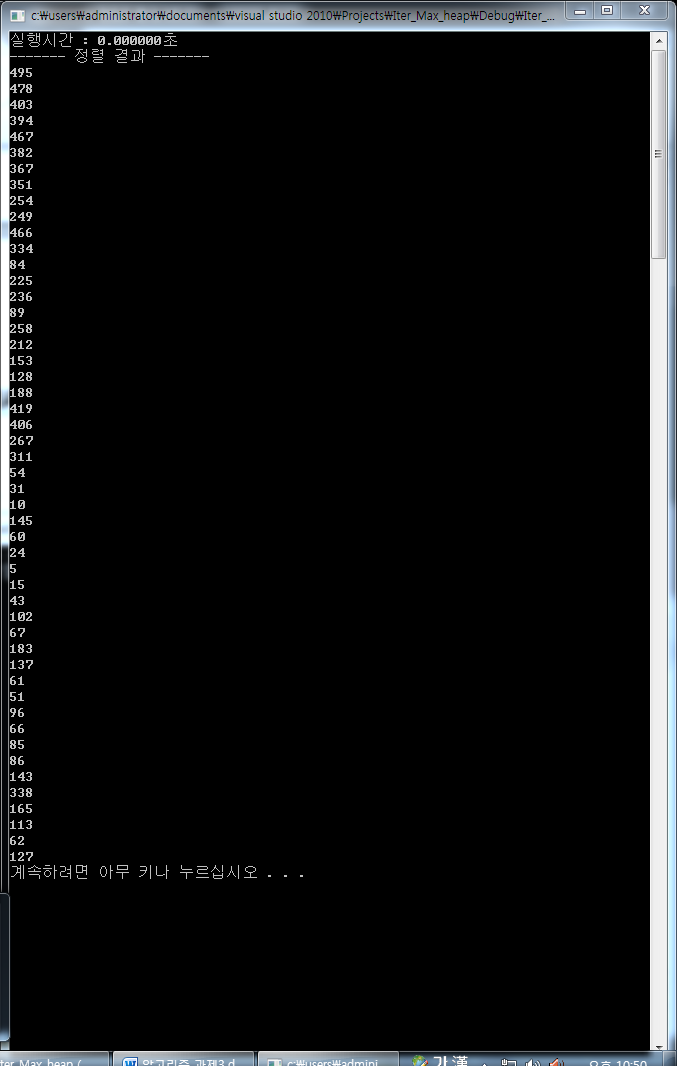
}

system("pause") ;

return 0;

}

**Iterative MAX-HEAPIFY.**

* Input50   
  - 출력 결과  
  
* 알고리즘 설명  
   기존의 heap 정렬과 같지만 reculsive 하게 함수를 실행하는 것이 아닌 iterative하게 함수를 사용한다.
* 컴파일 방법  
  input 폴더를 바탕화면에 넣는다  
  C:\\Users\\Administrator\\Desktop\\input\\input50.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) ;

// 왼¯¨­쪽E자U식öA이I 큰¡ì 경Æ©¡우¯i

if(left <= n && A[left] > A[i])

index = left ;

else

index = i ;

// 오¯A른¬¡Í쪽E 자U식öA이I 큰¡ì경Æ©¡우¯i

if(right <= n && A[right] > A[i])

index = right ;

if(index != i){

// 자U식öA의C 값Æ¨£이I 더¥o 클¡þ경Æ©¡우¯i

swap(A, i, index) ;

//다¥U음¨ö heapify로¤I 넘øN어úi감Æ¡§

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 숫ùy자U들ìe을¡í 배öe열¯¡© num에¯¢® 저u장a하I는¥A 부¬I분¬¨¢

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 시öA작U 시öA간Æ¡Ì 저u장a

start = clock() ;

// 정¢´렬¤A하I는¥A 부¬I분¬¨¢

for(i=0; i<sizeof(num)/sizeof(int); i++){

buildMaxHeap(num,i) ;

}

// 정¢´렬¤A 후A 시öA간Æ¡Ì 저u장a

end = clock() ;

// 실öC행a 시öA간Æ¡Ì 출a력¤A

printf("실öC행a시öA간Æ¡Ì : %lf초E\n",(end-start)/(double)1000) ;

// 정¢´렬¤A 결Æa과Æu 출a력¤A

printf("------- 정¢´렬¤A 결Æa과Æu ------- \n") ;

temp = 0 ;

for(i=0 ; i<n ; i++){

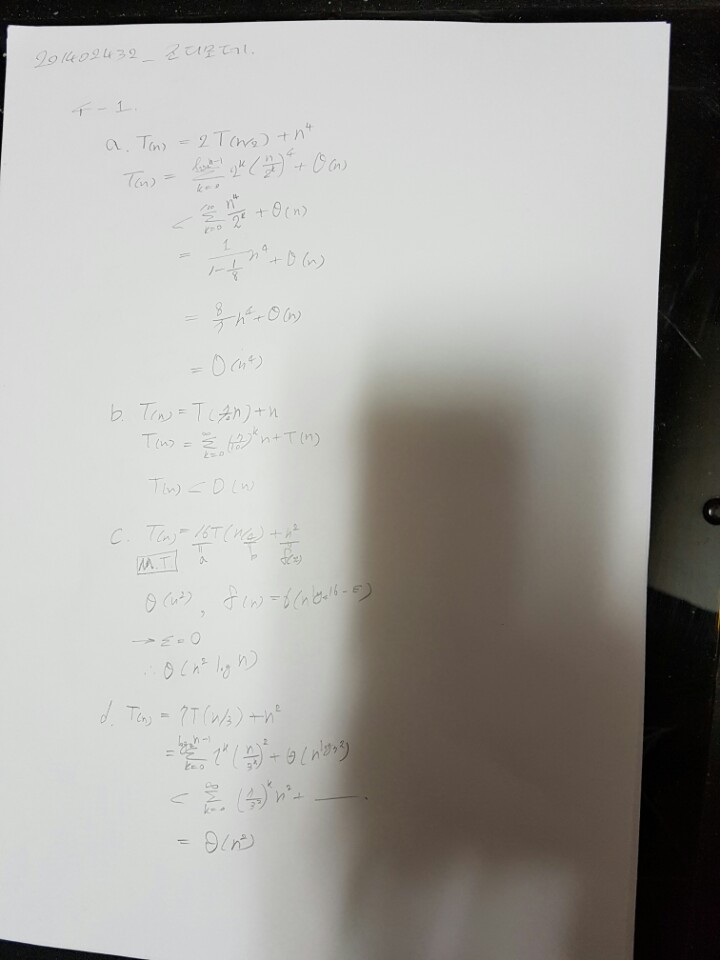
printf("%d\n",num[i]) ;

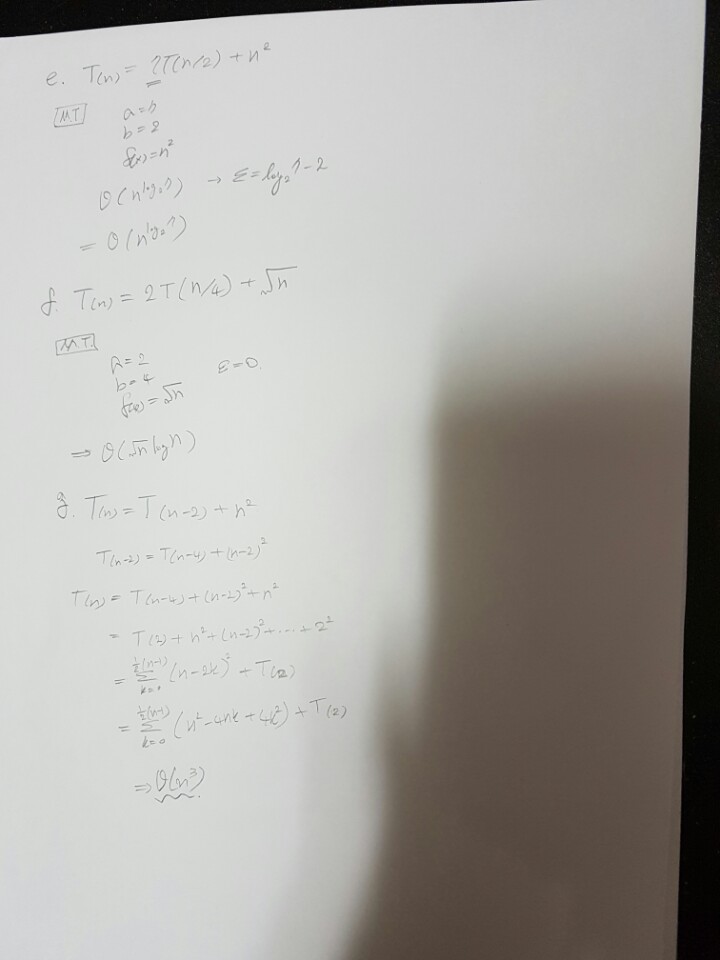
}

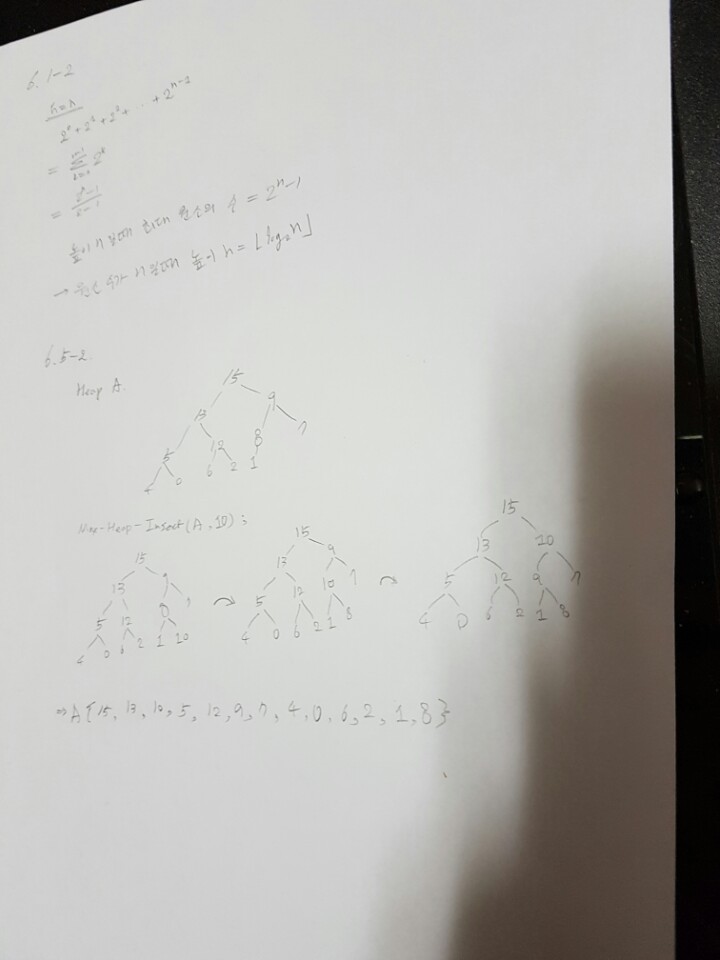
system("pause") ;

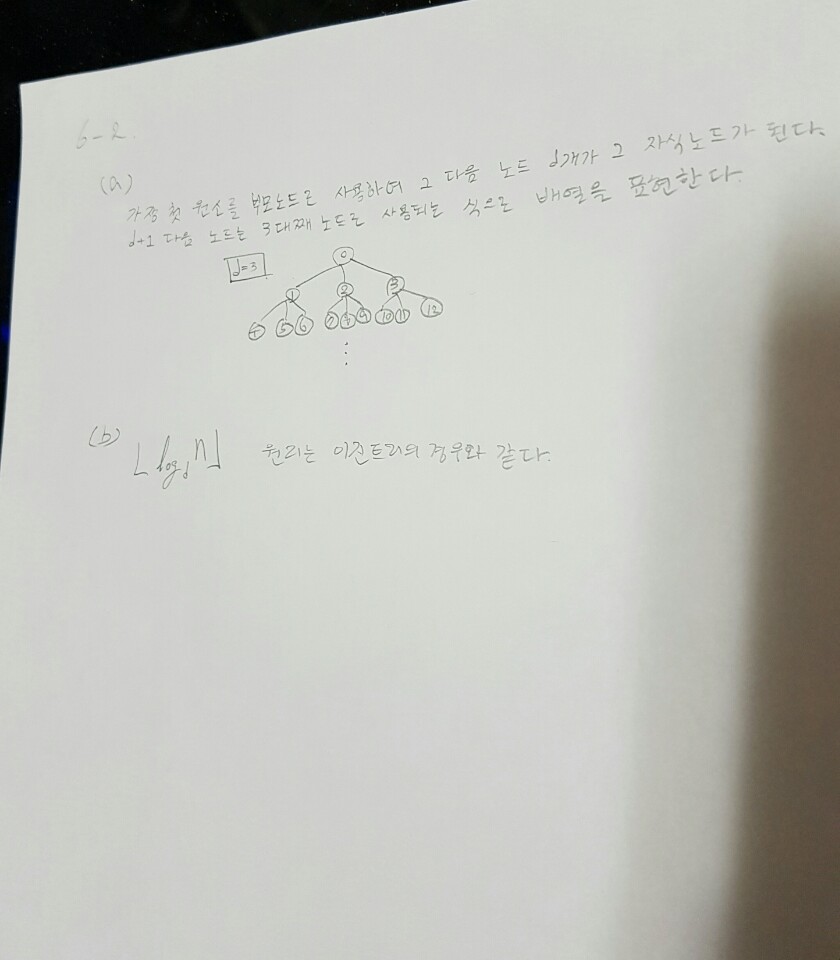
return 0;

}

****

****

****

****