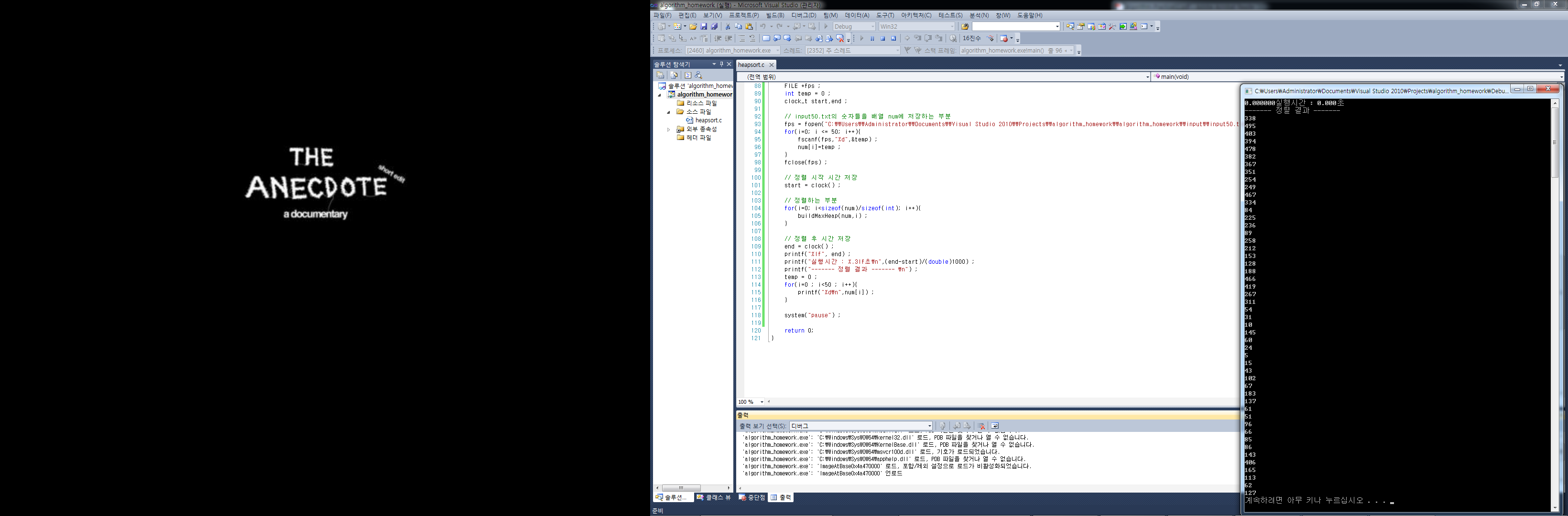
알고리즘 과제

Practice.02

**학번 : 201402432**

**이름 : 조디모데**

**Heap Sort.**

* Input50   
  - 출력 결과  
  
* 수행시간을 그래프

Input  
50 : 0  
500 : 0.01  
5000 : 1.015  
10000 : 3.778

* 알고리즘 설명  
   max[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. max힙을 구성
3. 루트를 가장 큰 수와 교환한다.
4. 2와 3을 반복한다.

* 컴파일 방법  
  input 폴더를 바탕화면에 넣는다  
  main에서 변수 n을 input의 크기에 맞게 설정  
  main함수에서 배열 num의 크기를 input의 크기+1로 설정
* 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 maxHeapify(int A[], int i, int n) {

int left = leftChild(i, n) ;

int right = rightChild(i, n) ;

int index ;

// 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¡§

maxHeapify(A ,index, n) ;

}

}

/\* Build Max-Heap

parameter A : heap implemented as array

parameter n : heap size \*/

void buildMaxHeap(int A[], int n){

int i ;

for(i = n ; i > 0 ; i--)

maxHeapify(A ,i ,n) ;

}

/\* Sort max-heap A

parameter A : max-heap implemented as array

parameter n : heap size \*/

void heapSort(int A[], int n) {

int i;

for(i = n ; i > 0 ; i--)

maxHeapify(A,i,n);

}

// main function

int main(void)

{

int i, num[10001], n ;

FILE \*fps ;

int temp = 0 ;

clock\_t start,end ;

n = 10000 ;

// input50.txt의C 숫ùy자U들ìe을¡í 배öe열¯¡© num에¯¢® 저u장a하I는¥A 부¬I분¬¨¢

fps = fopen("C:\\Users\\Administrator\\Desktop\\input\\input10000.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]) ;

}

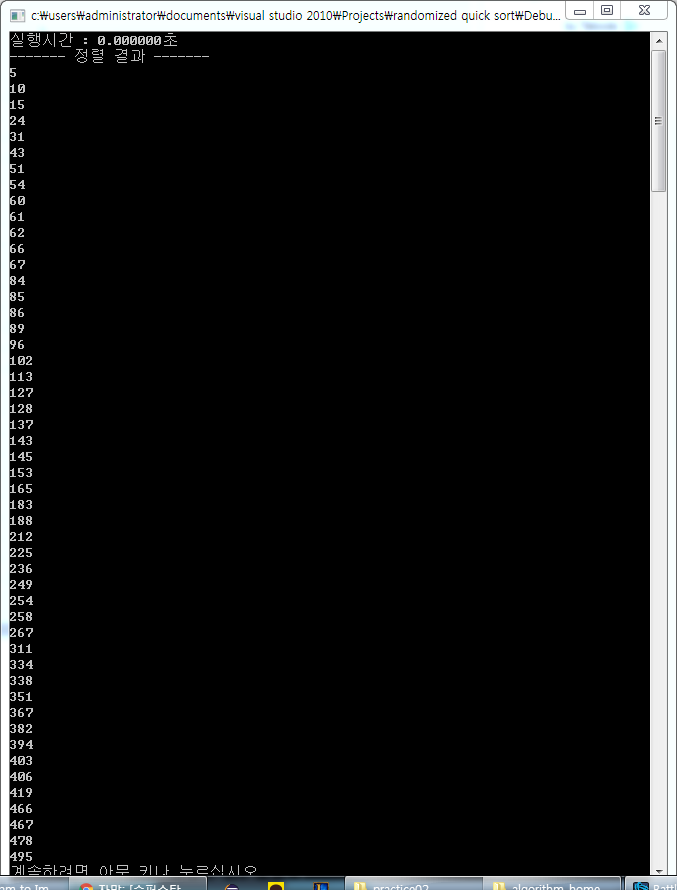
\*/

system("pause") ;

return 0;

}

**Quick Sort.**

* Input50   
  - 출력 결과  
  
* 수행시간을 그래프

Input  
50 : 0.00  
500 : 0.00  
5000 : 0.002  
10000 : 0.004

* 알고리즘 설명  
   기존의 퀵 정렬과 같지만 피봇을 정할 때 이전 피봇과 0 또는 배열의 크기 값 사이 중 적절한 값을 무작위로 선택하여 사용한다.
* 컴파일 방법  
  input 폴더를 바탕화면에 넣는다  
  main에서 변수 n을 input의 크기에 맞게 설정  
  main함수에서 배열 num의 크기를 input의 크기+1로 설정
* Code ( .C )

#include <stdio.h>

#include <stdlib.h>

#include<time.h>

/\* function to do swap in array A

parameter A : array

parameter i, j : index to be swapped\*/

void swap(int A[], int i, int j) {

int temp ;

temp = A[i] ;

A[i] = A[j] ;

A[j] = temp ;

}

/\* Randomized-Partition

parameter A : array to be sorted

parameter p, r :index of start, end point in array \*/

int partition(int A[], int p, int r) {

int pivotIndex = p + rand()%(r - p + 1); //랜¤¡Ì덤¥y한N 수ùo를¬| pivot으¢¬로¤I 설ù©ø정¢´

int pivot;

int i = p - 1;

int j;

pivot = A[pivotIndex];

swap(A, pivotIndex, r) ;

for (j = p; j < r; j++)

{

if (A[j] < pivot)

{

i++;

swap(A, i, j) ;

}

}

swap(A, i+1, r) ;

return i + 1 ;

}

/\* Randomized-Quicksort

parameter A : array to be sorted

parameter p, r : index of start, end point in array \*/

void quickSort(int A[], int p, int r) {

int j ;

if (p < r)

{

j = partition(A, p, r) ;

quickSort(A, p, j-1) ;

quickSort(A, j+1, r) ;

}

}

// main fucntion

int main(void)

{

int i, num[10001], n ;

FILE \*fps ;

int temp = 0 ;

clock\_t start,end ;

n = 10000 ;

// input50.txt의C 숫ùy자U들ìe을¡í 배öe열¯¡© num에¯¢® 저u장a하I는¥A 부¬I분¬¨¢

fps = fopen("C:\\Users\\Administrator\\Desktop\\input\\input10000.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¬¨¢

quickSort(num, 0, n-1) ;

// 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;

}