- 프로그램 소스

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int findKthSmallest(int \*A, int n, int k);

/\*

function name : findKthSmallest

pre condition(전제 조건) : 리스트 A, 리스트의 크기, 찾고싶은 순위의 최소값

function(기능) : 리스트에서 k번쨰 최솟값을 찾아 리턴한다

\*/

void buildHeap(int \*A, int n);

/\*

function name : buildHeap

pre condition(전제 조건) : 정렬되어야할 리스트와 그 리스트의 크기

function(기능) : 최소힙을 상향식으로 생성한다(비재귀 버젼)

\*/

void downHeap(int \* A,int i, int last);

/\*

function name : downHeap

pre condition(전제 조건) : 리스트와 downHeap시킬 값의 위치, 리스트의 크기

function(기능) : 힙순서(최소힙) 속성을 복구한다

\*/

void arraySwap(int \*A, int n1, int n2);

/\*

function name : arraySwap

pre condition(전제 조건) : 리스트와 바꾸고자 하는 두 값의 위치 값

function(기능) : A[n1]과 A[n2]의 값을 바꿔준다

\*/

int main()

{

int n; // 리스트의 크기를 입력받아 그 값을 저장할 변수

int \* A; // 동적할당 받아 리스트를 저장할 배열

int i; // 반복문을 위한 변수

int k; // 순위를 입력받아 그 값을 저장할 변수

int e; // k번째 순위에 있는 값을 저장할 변수

printf("리스트 크기 : ");

scanf("%d",&n); // 리스트의 크기를 입력받는다

A = (int\*)malloc(sizeof(int) \* n); // n의 크기만큼 동적할당

srand((unsigned)time(NULL)); // for random

for (i=1; i<=n; i++) {

A[i] = (rand()\*rand()) % 1000000+1; // rand()는 범위가 0~32767까지만이다. 따라서 1~1000000사이의 정수를 발생시키기 위해 rand()\*rand()를 해서 범위를 넓히고 %1000000+1을 한다

}

if (n<10000) { // n이 작은 수인 경우에만 리스트 출력

printf("리스트 : ");

for (i=1; i<=n; i++)

printf("%d ",A[i]); // 리스트의 값을 출력

printf("\n");

}

for (i=1; i<4; i++) {

printf("순위: ");

scanf("%d",&k); // 원하는 순위를 입력받는다

e = findKthSmallest(A, n, k); // findKthSmallest가 k번째 작은 값을 리턴해 e에 저장

printf("원소: %d\n",e); // 출력

} // k번째 작은 값을 3번 구한다

return 0;

}

int findKthSmallest(int \*A, int n, int k) {

int i;

int e;

int last = n; // 리스트의 크기를 last에 저장한다

buildHeap(A, n); // 최소힙 만드는 buildHeap 호출

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

e = A[1]; // 최솟값 e에 저장

arraySwap(A,1,last); // 루트에 있는 최솟값과 마지막에 있는 값과 바꾼다.

last--; // 리스트 크기 1개 줄임으로써 마지막으로 간 최솟값 삭제

downHeap(A,1,last); // 힙순서(최소힙) 속성 복구

} // 최솟값을 삭제하고 리턴해주는(removeMin) for문을 k번 돌림으로써 k번째 작은 값을 e에 저장한다

return e; // k번째 최소값 e 리턴

}

void buildHeap(int \*A, int n) {

int i;

for (i=n/2; i>=1; i--) // 최초의 비독신자 노드 n/2에서부터 시작

downHeap(A, i, n); // 힙순서(최소힙) 속성 복구

// for문 종료시 최소힙 생성 완료

return;

}

void downHeap(int \* A, int i, int last) {

int left, right;

int smaller;

left = 2\*i; // i에 존재하는 노드에 대해 왼쪽 자식은 2i에 존재한다

right = 2\*i+1; // i에 존재하는 노드에 대해 왼쪽 자식은 2i+1에 존재한다

if (left > last) // 외부노드일때 리터한다

return;

smaller = left;

if (right <= last) {

if (A[right] < A[smaller]) // smaller의 위치에 있는 키값이 right의 위치에 있는 키값보다 작으면 / 최소힙

smaller = right;

}

if (A[i] <= A[smaller]) // i의 위치에 있는 값이 smaller의 위치에 있는 값보다 작거나 같아야만 리턴 / 최소힙

return;

arraySwap(A,i,smaller); // i의 위치에 있는 값과 smaller의 위치에 있는 값을 교환한다

downHeap(A, smaller, last); // 재귀함수

}

void arraySwap(int \* A, int n1, int n2) {

int temp; // 임시로 값을 저장할 변수

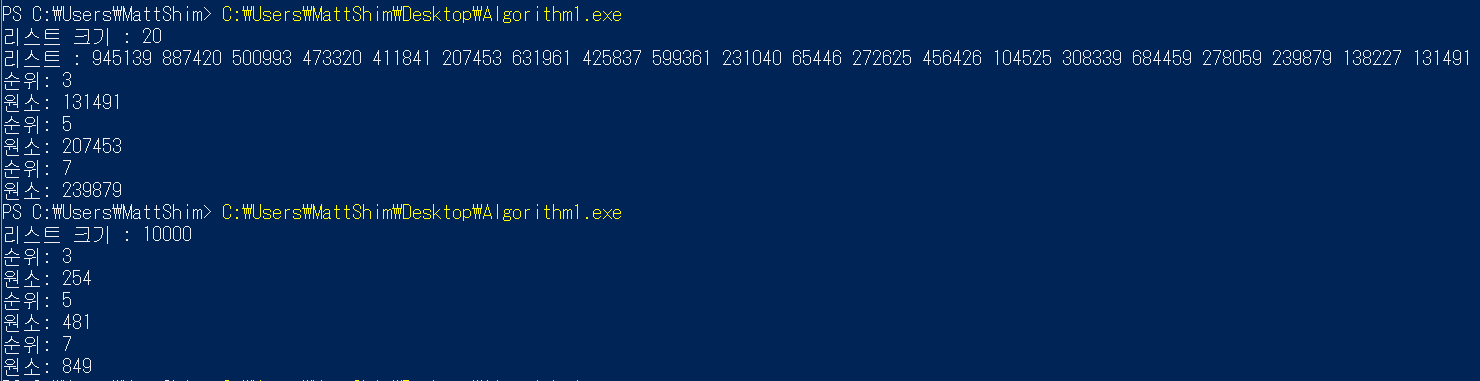
temp = A[n1];

A[n1] = A[n2];

A[n2] = temp;

// A[n1]의 값과 A[n2]의 값을 swap한다

}

- 사용자 입력과 프로그램 출력을 보여주는 이미지