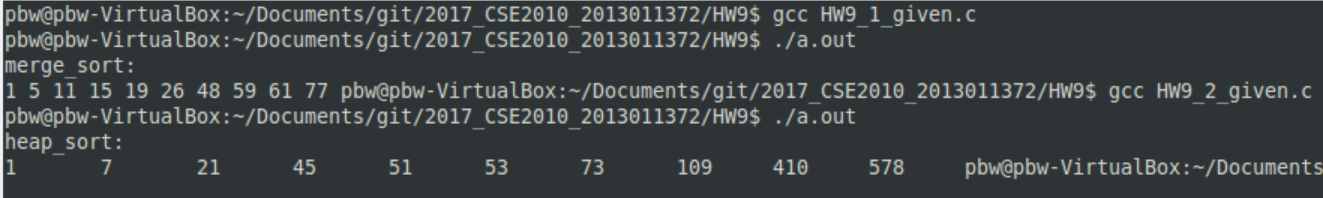
2013011372 박병욱 자료구조론 HW9

1. 실행결과

.

1. 소스코드

* 머지소트

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define SWAP(x, y, t) {t=x; x=y; y=t;}

#define MAX\_SIZE 10

#define NAME\_SIZE 32

/\* i는 정렬된 왼쪽리스트에 대한 인덱스

j는 정렬된 오른쪽리스트에 대한 인덱스

k는 정렬될 리스트에 대한 인덱스 \*/

void merge(int list[], int left, int mid, int right)

{

int i, j, k, l;

i=left; j=mid+1; k=left;

int tmp[MAX\_SIZE];

int t;

memmove(tmp,list,MAX\_SIZE\*4);

while(i <= mid && j <= right){

if(tmp[i] <= tmp[j]){

SWAP(list[k],tmp[i],t);

k++;

i++;

}else{

SWAP(list[k],tmp[j],t);

k++;

j++;

}

}

while(i <= mid)

{

SWAP(list[k],tmp[i],t);

k++;

i++;

}

while(j <= right)

{

SWAP(list[k],tmp[j],t);

k++;

j++;

}

}

void merge\_sort(int list[], int left, int right)

{

int mid;

if(left<right){

mid = (left+right)/2; /\* 리스트의 균등 분할 \*/

merge\_sort(list, left, mid); /\* 부분 리스트 정렬 \*/

merge\_sort(list, mid+1, right); /\* 부분 리스트 정렬 \*/

merge(list, left, mid, right); /\* 합병 \*/

}

}

void main()

{

FILE \*f;

int i;

int list1[MAX\_SIZE];

f= fopen("input1.txt","r");

fscanf(f,"%d %d %d %d %d %d %d %d %d %d",

&list1[0],&list1[1],&list1[2],&list1[3],&list1[4],

&list1[5],&list1[6],&list1[7],&list1[8],&list1[9]);

//합병정렬

merge\_sort(list1, 0, MAX\_SIZE-1); /\* 선택정렬 호출 \*/

printf("merge\_sort: \n");

for(i=0; i<MAX\_SIZE; i++) printf("%d ", list1[i]);

fclose(f);

}

* 힙소트

#include <stdio.h>

#include <stdlib.h>

#define SWAP(x, y, t) {t=x; x=y; y=t;}

#define MAX\_SIZE 10

#define NAME\_SIZE 32

//adjust 함수의 구현은 수업시간에 다루었고 동일한 알로기즘으로 구현되어 있음

void adjust(int heap[], int root, int n)

{

int j, k;

k=heap[root];

j=2\*root;

while ((j<=n)) /\* Iterate until child exist \*/

{

if (j<n) /\* Is right child exist? \*/

if (heap[j]<heap[j+1]) /\* Take bigger child \*/

j=j+1;

if (k>=heap[j])

break;

else{

heap[j/2]=heap[j]; /\* Swap with child \*/

j=2\*j; }

}

heap[j/2]=k;

}

void heap\_sort(int list[], int n)

{

int i, j;

int temp;

//Except leaves, take adjust function

for(i=n/2; i>=0; i--)

adjust(list,i,n);

//Swap, root and rear of list

//then re adjust root

for(i=n-1; i>=0;i--)

{

SWAP(list[0], list[i+1],temp);

adjust(list, 0, i);

}

}

void main()

{

FILE \*f;

int i;

int list1[MAX\_SIZE];

f= fopen("input2.txt","r");

fscanf(f,"%d %d %d %d %d %d %d %d %d %d",

&list1[0],&list1[1],&list1[2],&list1[3],&list1[4],

&list1[5],&list1[6],&list1[7],&list1[8],&list1[9]);

//힙정렬

heap\_sort(list1, MAX\_SIZE); /\* 선택정렬 호출 \*/

printf("heap\_sort: \n");

for(i=1; i<=MAX\_SIZE; i++) printf("%d \t",list1[i] );

}

1. 코드에 대한 설명

머지소트:

-mergesort()함수를 이용하여, 리스트를 재귀적으로 반으로 나누고, 나눈 부분리스트들을 merge()한다.

- mergesort()를 재귀적으로 사용하여 크기가 각각 2, 2 인 부분리스트 들은, 이미 그 안에 1, 1부분 리스트들이 merge가 되어 정렬이 된상태로 merge()를 하게 된다.

-merge()함수는 임시로, tmp[]리스트를 만들고, memmove를 이용하여, 실제 리스트의 정보를 복사하고, i는 왼쪽 부분리스트의 첫번째부터, j는 오른쪽 부분리스트의 첫번째 부터 가리키고, 각각 부분리스트는 오름차순으로 정렬되었으므로, tmp[i], tmp[j]를 비교하여 둘중의 작은 것과 list[k]를 스왑한다.그리고 tmp[i]가 작다면, i++, tmp[j]가 작다면 j++, 그후 k++을 실행한다. SWAP(list[k++],tmp[i++],t);를 시도해보려 하였으나, 매크로와 ++ 연산자간에 충돌이 있었다.

- 한 부분리스트의 끝까지 i나 j가 이동하였으면, 남은 부분 리스트를 그대로 list로 옮긴다.

힙소트

-heapsort()함수를 이용하여 leaves를 제외한 노드들에 adjust()함수를 적용시킨다.

-adjust()함수는 힙정렬된 왼쪽 트리와, 오른쪽 트리를 다시 힙정렬한다.

Adjust()함수에선 루트로 부터 자식을 얻고, 왼쪽 오른쪽 자식중 큰 자식을 택하고 그 큰자식의 부분트리안에 list[root]보다 큰값을 차례로 위로 올린다.