**Computer Algorithm**

**Chapter 07**

**Heap**

**[실습] HeapSort 구현**

**실습 목표**

* **자료구조 Queue에서 최소, 최대 값이 선택되게 하는 자료구조를 이해한다.**
* **Heap과 Priority Queue에 대해 학습하고 이해한다.**
* **HeapSort, PriorityQueue 시간복잡도에 대해 생각해보고 정리한다.**

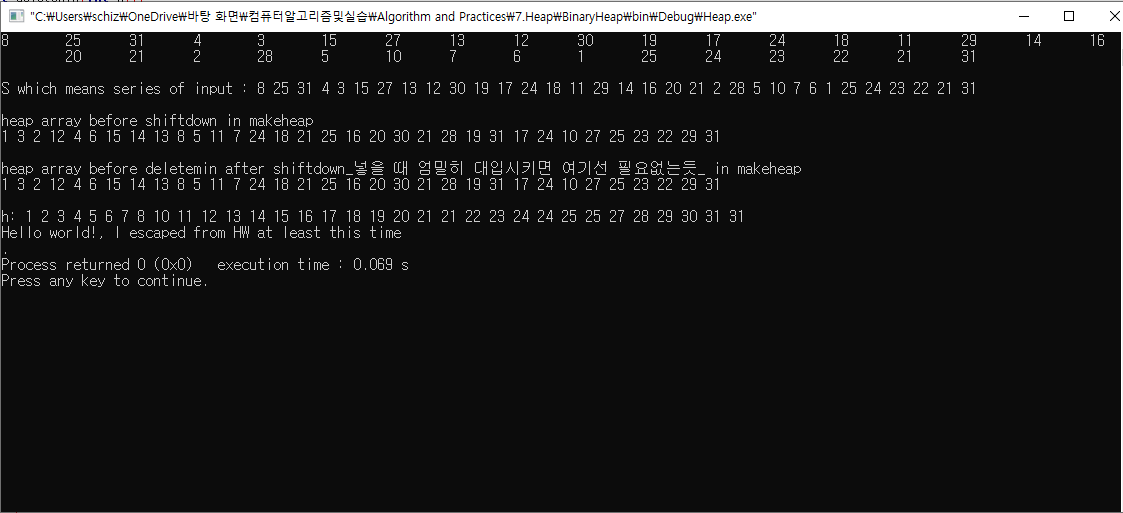
**요구사항**

* **실습과제 1) (30점)**

입력은 파일에서 받는 것으로 하며 파일이름은 input.txt 로 한다. 또 한 파일의 내용은 아래와 같으며 숫자들은 탭으로 구분된다. 이를 Heap으로 만들고 가장 작은 값부터 큰 값까지 순서대로 출력하는 프로그램을 구현한다.

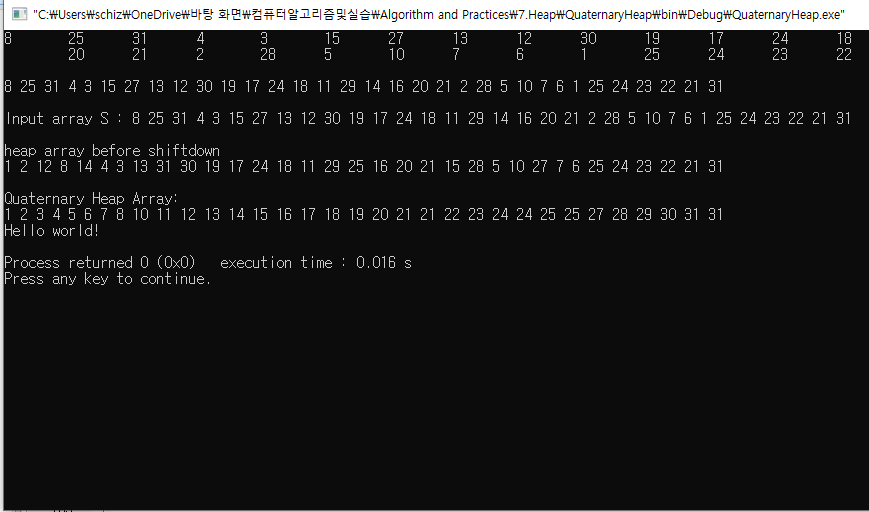
8 25 31 4 3 15 27 13 12 30 19 17 24 18 11 29 14 16 20 21 2 28 5 10 7 6 1 25 24 23 22 21 31

makeHeap()을 통해 heap을 구성한 후 deleteMin()을 통해 가장 작 은 값을 추출하게 되고 heap이 빌때까지 이 과정을 반복하여 출력 한다.



* **실습과제 2) (70점)**

실습과제1에서 Binary Heap을 구현하였다. 하지만 Binary가 아닌 Quaternary Heap (자식노드를 최대 4개까지 가지는 Heap)을 구현하 게 되면 더 빠른 속도의 Bubble Up과 Shift Down을 구현할 수 있다. 실습 문제 1의 Heap 관련 구현 내용을 Quaternary Heap을 실행시 킬 수 있도록 수정 구현하고, 실습 문제 1에서 주어진 숫자들을 Quaternary Heap을 사용하여 가장 작은 값부터 큰 값까지 순서대로 출력하는 프로그램을 구현한다.



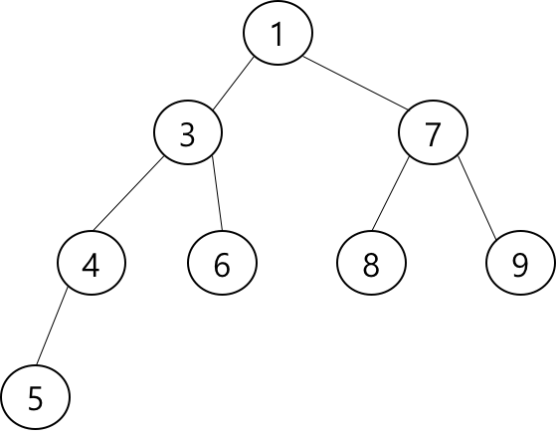
**배경지식**

Priority Queue는 최대 또는 최소 값이 Queue에서 선택되도록 하는 자료구조로서, 정렬하지 않은 배열을 통해 구현될 수 있다. 하지만 이 경우 새로운 값을 추가할 때에는 빠르게 진행될 수 있으나 최소값 또는 최대값을 얻을 경우에는 의 시간이 걸린다. 이러한 Priority Queue는 Heap이라는 자료구조로 구현될 경우 빠른 () 최대값 또는 최소값 추출이 가능하게 된다.

Heap은 기본적으로 Complete Tree의 모양으로 구성되며 자료가 트리의 각 레벨에서 좌측에서 우측 방향으로 채워지는 배열이다. Heap에는 Complete Tree외에 추가적인 특성을 가지는 데 Heap의 모든 노드의 값은 그 자식 노드보다 작거나 같다는 특성을 가진다. 따라서 루트 노드는 가장 작은 값을 가지게 된다.

Heap은 완전 이진 트리의 형태를 가지고 있기 때문에 배열로 구현할 수 있다. 배열로 구현 할 경우 자식 노드는 부모 노드의 배열 상의 위치 i의 2i+1, 2i+2에 위치하게 된다.

다음은 Heap의 예이며 이 그림을 Heap으로 구현한 배열은 아래와 같다.



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 7 | 4 | 6 | 8 | 9 | 5 |

Heap의 특성상 가장 작은 값이 루트에 위치하므로 가장 작은 값을 추출할 때 시간이 적게 걸릴 수 밖에 없으나 Heap의 특성을 유지해야 하므로 shiftdown을 통해 계속적으로 가장 작은 값이 루트에 위치하며 Heap의 특성을 유지할 수 있어야 한다.

새로운 값을 삽입할 경우에도 Heap의 특성을 유지하기 위해 적절한 위치에 삽입할 수 있어야 하는데, 우선 새 값을 배열의 가장 끝이 놓은 후, bubbleup을 통해 데이터들을 이동시켜 Heap의 특성을 유지하도록 한다.

shiftdown은 루트의 값을 제거했을 경우 루트의 값이 비게 되므로 우선 배열 가장 뒤의 원소를 루트에 이동시킨 후, 그 값을 Heap의 자식 노드들로 따라 내려가며 적절한 곳에 위치시키는 동작을 말한다.

bubbleup은 새로운 값을 우선 배열 가장 뒤의 값에 위치시킨 후, 그 값의 부모 노드들로 따라 올라가며 적절한 곳에 위치시키는 동작을 말한다.

shiftdown과 bubbleup을 포함한 Heap의 구성에 필요한 함수들은 아래의 pseudo code로 표현된다.

void insert(h[], x) // insert new data x to heap h[]

bubbleup(h, x, |h|)

int deleteMin(h[]) // delete minimum value from heap h[]

if |h| = 0

return -1000

else

// 배열 h의 크기가 0이 아니면 shiftdown을 한다.

x = h[0]

shiftdown(h, h[|h|], 0)

return x

void makeheap(S[], h[]) // build heap h[] from array S[]

copy each x in array S to array h

for i = |S| to 1

shiftdown(h, h[i],i)

void bubbleup(h[], x, i)

// place element x in h[i] and let it bubble up

p = ceil(i/2)

while i != 1 and h[p] > x

h[i] = h[p]

i = p

p = ceil(i/2)

h[i] = x

void shiftdown(h[], x, i)

// place element x in h[i] and let it shift down

/\* h[i]의 child중 작은 값을 가지는 child의 인덱스 값을 c로 한다. 단 c 값이 힙의 크기보다 클 경우에는 -1을 리턴한다. \*/

c = mininum child of h[i]

while c != -1 and h[c] < x

h[i] = h[c]

i = c

c = minimum child of h[i]

h[i] = x

**제출방법**

* 보고서 작성방법: 실습문제 번호별로 결과가 나온 화면의 내용을

캡쳐하여 보고서에 붙여 놓는다.

* 소스코드의 파일이름에 연습문제 번호를 붙이는 것을 잊지 않는다. 예) ex-1.c, ex-2.c
* 결과 보고서에 이름과 작성 날짜를 기입하는 것을 잊지 않는다. 예) 김웅섭\_2020\_09\_01.doc
* 실행결과를 보고서에 작성하여 소스코드와 함께 제출한다.
* 제출 마감 : e-class 제출 마감시간까지