**hw6\_1번.cpp**

**1. Code Explanation**

1) void build\_max\_heap(HeapType\* h)

-input: we will build maxheap in h

-int parent: index of current node

-int child: index of child node

-element temp: data in i th heap

-> walk backwards through the array from n/2 to 1, moving the element on each node until it meets the heap property

(elements in index n/2+1 to n already met the heap property )

for (int i = h->heap\_size / 2;i >= 1;i--)

->implement like delete\_max\_heap function

while (child <= h->heap\_size) {

// Find a bigger child node

if ((child < h->heap\_size) &&

(h->heap[child].key) < h->heap[child + 1].key)

child++;

if (temp.key >= h->heap[child].key) break;

// Move down one level

h->heap[parent] = h->heap[child];

parent= child;

child \*= 2;

}

h->heap[parent] = temp; //define the position of element in heap[i]

2) measure running time

: use clock() function

-double start: time when 'heap\_sort' function is started

-double finish: time when 'heap\_sort' function is finished

-double duration: duration time of 'heap\_sort'

start =(double) clock()/CLOCKS\_PER\_SEC;

finish = (double)clock()/ CLOCKS\_PER\_SEC;

duration = finish - start;

->print the runtime

printf("Input size is %d, and the runtime is %.50f seconds\n",input\_size, duration);

3) other functions

: same with the skeleton code

**2. Results**

When input array size is 10

**텍스트이(가) 표시된 사진

자동 생성된 설명**runtime is very short

When input array size is 100

**텍스트이(가) 표시된 사진

자동 생성된 설명**runtime is very short

When input array size is 1000

**텍스트이(가) 표시된 사진

자동 생성된 설명**runtime is very short

When input array size is 5000**텍스트이(가) 표시된 사진

자동 생성된 설명**runtime is about 0.001 second

**hw6\_2.cpp**

**1. Code Explanation**

1) void huffman\_traversal(TreeNode\* node)

-input(TreeNode\* node): root node

-int index: index of node in huffman tree

-int code[MAX\_BIT]: huffman code

-int size: number of bits

-int row: the row index of m\_LUT

-> if node is leaf node: save huffman code and size in m\_LUT and m\_bit\_size, and return the function

if (node->l == NULL && node->r == NULL)

switch (node->data) { //decide the row of m\_LUT

-> if node is non-leaf node: use recursion

save 0 and move down to left child node using recursive call

code[index] = 0;

huffman\_traversal(node->l);

index--;size--;

save 1 and move down to right child node using recursive call

code[index] = 1;

huffman\_traversal(node->r);

index--;size--;

2) void huffman\_encoding(char\* str, bits\_stream\* bits\_str)

-char\* str: data we will encode

-bits\_stream\* bits\_str: save encoded string

-int interval: length of bits\_str

-int idx= 0: index of bits\_str

-> find huffman code in m\_LUT and save in stream

for (int i = 0;i < strlen(str);i++) {

for (int j = 0;j < m\_bit\_size[(int)str[i] - 97];j++) {

bits\_str->stream[idx++] = m\_LUT[(int)str[i] - 97][j];

}

bits\_str->length += m\_bit\_size[(int)str[i] - 97];

}

->print encoded string and totall length of bits stream

3) int huffman\_decoding(bits\_stream\* bits\_str, TreeNode\* node, char\* decoded\_str)

-bits\_stream\* bits\_str: we will decoded bits\_str->stream

-TreeNode\* node: root node of huffman tree

-char\* decoded\_str: save decoded string

-TreeNode\* parent: current node

-TreeNode\* child: child node of current node

-int index\_char: total number of decoded chars & index of decoded\_str

-> read each bit of bits\_str

-> if char is 0, then go to left child node

if (bits\_str->stream[i] == 0)

child = parent->l;

-> if child node is leaf node: save data of child node in decoded\_str

if (child->l == NULL && child->r == NULL)

-> if child if non-leaf node: go down one level

parent = child;

-> if char is 1, then go to right child node

if (bits\_str->stream[i] == 1) {

child = parent->r;

-> if child node is leaf node: save data of child node in decoded\_str

if (child->l == NULL && child->r == NULL)

-> if child if non-leaf node: go down one level

-> print decoded string and total number of decoded characters

4) other functions

:same with the skeleton code

**2. Result**

**텍스트이(가) 표시된 사진

자동 생성된 설명**