REPORT

Functionality

First, I created array and priority queue with malloc function:

```
int *numberArray = (int*)malloc( _Size: NUM * sizeof(int));
int *pri_que = (int*)malloc( _Size: NUM * sizeof(int));
int currentSize = -1;
int *pSize = &currentSize;
```

Then I read elements from file and added elements to the numberArray:

```
FILE *myFile;
myFile = fopen( _FileName: "sayilar.txt",  _Mode: "r");

if (myFile == NULL) {
    printf( _Format: "Error Reading File\n");
    exit( _Code: 0);
}

for (int i = 0; i < NUM; i++) {
    fscanf(myFile,  _Format: "%d ", &numberArray[i]);
}

fclose(myFile);
```

After that I built the queue and measure the method execution time:

```
clock_t t;
t = clock();

for (int j = 0; j < NUM; ++j) {
    queue(numberArray[j], pri_que, pSize);
}

t = clock() - t;
double time_taken = ((double)t)/CLOCKS_PER_SEC;
printf( _Format: "Building queue took %f seconds to execute. ", time_taken);
```

I converted priority queue array into the structure with pointers:

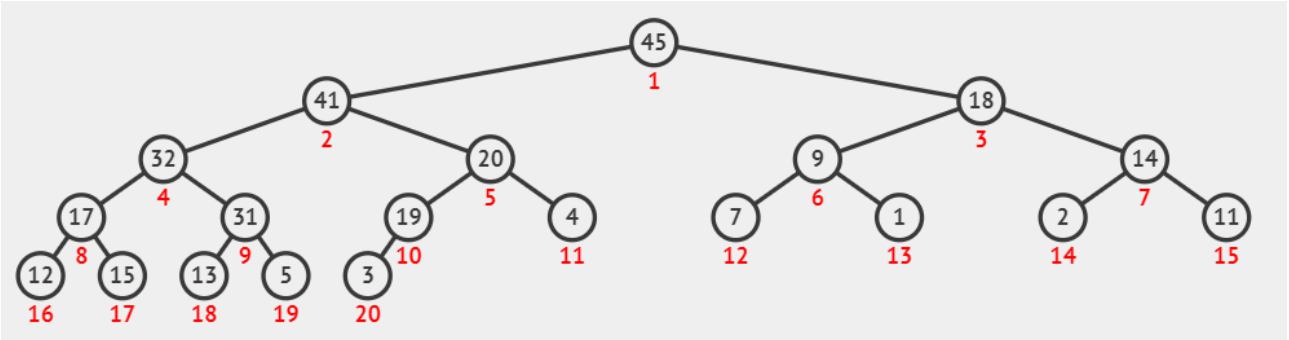
```
node *root = convert(pri_que, start: 0, end: NUM-1);
```

Then I printed resulting structure by breadth first traversal to the output file. For breadth first traversal, I used source from GeeksForGeeks (Level Order Binary Tree Traversal – GeeksforGeeks):

```
// open the file in write mode
fptr = fopen( _FileName: "integers.txt", _Mode: "w");

if (fptr != NULL) {
    printf( _Format: "File created successfully!\n");
}
else {
    printf( _Format: "Failed to create the file.\n");
    // exit status for OS that an error occurred
    return -1;
}

printBreadthFirst(root);

// close connection
fclose(fptr);
```

Input file content: [45 20 14 12 31 7 11 13 15 3 4 9 1 2 18 17 32 41 5 19]

Max heap tree of this file content should look like the picture below:



Program result (Output file content):

```
45 41 18 32 20 9 14 17 31 19 4 7 1 2 11 12 15 13 5 3
```

Node 1: data = 45, left = 41 (Node 2), right = 18 (Node 3)
Node 2: data = 41, left = 32 (Node 4), right = 20 (Node 5)
Node 3: data = 18, left = 9 (Node 6), right = 14 (Node 7)
Node 4: data = 32 ...
...

Performance Results

I used 2^24 elements because of limitations due to computer settings.

In C, building queue took 0.666 seconds with 16777216 (2^24) elements.

```
Number of integers: 16777216
Building queue took 0.666000 seconds to execute.
```

In Java, it took 20.351 seconds with the same number of elements.

```
Number of integers: 16777216
Building queue in milliseconds: 20351
```

Onur Cihangir
250201049