CSE222_HW06
ERCAN UCA 091044011

PART_1

✓ **HuffmanTree** Classımda

➔ Encode method implement edildi.

*Private recursive method;

```
/**
 * Search for character and when found return coded string
 * @param ch is the search character on huffman tree.
 * @param huffmanTree is the builed huffman tree.
 * @param code is the code when found ch will return.
 * @return is the for ch code on huffmantree.
 */
```

- `private String encode(char ch, BinaryTree<HuffData> huffmanTree, String code)`

* Public method;

```
/**
* Method to encode string message into Huffman encodes.
* @param message The input message as a String
* which is composed on the specified alphabet in the book
* @param huffmanTree It's created huffman code for the alphabet
* @return The encoded message as a String zero and ones.
*/
public String encode(String message, BinaryTree<HuffData> huffmanTree);
```

PART_2

➔ **AscendingOrderTraversal** Classımda;

➔ private final LinkedList<BinaryTree<E>> parents;

➔ public AscendinOrderTraversal(BinaryTree<E> root);

-> BinaryTree<E> current;

```
for (current= root; current != null; current = current.getLeftSubtree()) {
    this.parents.push(current);
}
```

ile ilk önce ağacın sol kısmı linkedliste eklenir root'dan başlayarak.

➔ **public boolean hasNext();**

```
/**
    * Look if with iterator we have next element.
 * @return true if have next element, otherwise false.
 */
```

➔ **public E next();**

```
/**
  * This method return for next element. Rule of inorder tree
  * @return next element
  */
BinaryTree current = parents.pop();
for(BinaryTree<E> child = current.getRightSubtree(); child !=null; child =
child.getLeftSubtree())
{
        this.parents.push(child);
```

```
        }
        return (E) current.getData();
```

Ilk eleman pop edilir, sonrasındaki treenin diğer elemanı eklenir linkedlist.

➔ **public Iterator<E> iterator();**

```
    /**
    * Out iterator method
    * @return reference of itself.
    */
    return this;
```

Classimizin referansını döndürür.

```
/*
        Created Binary Tree like this.
                        36
                      /    \
                     /      \
                    /        \
                   /          \
                  25          40
                 /  \         / \
                /    \       /   \
               /      \     /     \
              10      27   35     45
             / \     / \   / \    / \
            /   \   /   \ /   \  /   \
           5   15 26  28 32 37 43   50
          /\   /\  /\  /\ /\ /\ /\  /\
        null null null null null
    */
```

PART_3

➔ **PriorityQueue_ArrayList** Classımda

```
    /**
     * This Constructor take priority queue and comparator
     * @param myQueue my priority queue
     * @param comp my current comparator
     */
```

➢ **Constuctor with 2 parameters**
-> Priortiy queue elemanları arraylisteme eklenir.

```
    /**
     * Remove first element of Array queue
     * @return and return removed element.
     */
    @Override
```

➢ **public E dequeue()**

Ilk eleman silinir- remove(0) ile

```
    /**
     * Add element at end
     * @param element will be add.
```

```
     */
    @Override
```

- **public void enqueue(E element)**

Sona eleman eklenir add(size-1, item)ile.

```
    /**
     * This method, look my array queue is empty.
     * @return if empty true, otherwise false.
     */
    @Override
```

- **public boolean isEmpty()**

Arraylistimde eleman var mı ona bakar.

```
    /**
     * Return number of my array queue
     * @return the size of my array queue.
     */
    @Override
```

- **public int size()**

Arraylistimde kaç eleman var onu verir.

➔ **PriorityQueue_LinkedList Classımda**

```
    /**
     * This Constructor take priority queue and comparator
     * @param myQueue my priority queue
     * @param comp my current comparator
     */
```

- **Constuctor with 2 parameters**

-> Priortiy queue elemanları linkedlisteme eklenir.

```
    /**
     * Remove first element of Array queue
     * @return and return removed element.
     */
    @Override
```

- **public E dequeue()**

Ilk eleman silinir- removefirst() ile

```
    /**
     * Add element at end
     * @param element will be add.
     */
    @Override
```

- **public void enqueue(E element)**

Sona eleman eklenir addLast() ile.

```
    /**
     * This method, look my link queue is empty.
     * @return if empty true, otherwise false.
     */
    @Override
```

- **public boolean isEmpty()**
  LinkedListimde eleman var mı ona bakar.

```
/**
 * Return number of my link queue
 * @return the size of my link queue.
 */
@Override
```

- **public int size()**

LinkedListimde kaç eleman var onu verir.

➔ **PriorityQueue_BinarySearchTree Classımda**

-
```
/**
 * This Constructor take priority queue and comparator
 * @param myQueue my priority queue
 * @param comp my current comparator
 */
```

- **Constuctor with 2 parameters**
  -> Priortiy queue elemanları binaryTreeme eklenir.

```
/**
 * Remove first element of Tree queue
 * @return and return removed element.
 */
@Override
```

- **public E dequeue()**

Ilk eleman silinir- delete(root.getData()) ile

```
/**
 * Add element at end
 * @param element will be add.
 */
@Override
```

- **public void enqueue(E element)**

Sona eleman eklenir add(element) ile.

```
/**
 * This method, look my tree queue is empty.
 * @return if empty true, otherwise false.
 */
@Override
```

- **public boolean isEmpty()**
  BinaryTreem de eleman var mı ona bakar.

```
/**
 * Return number of my tree queue
 * @return the size of my tree queue.
 */
@Override
```

- **public int size()**

BinaryTreem de kaç eleman var onu verir.

- ---

➔ **PriorityQueue_Vector** Classımda

-
- ```
  /**
   * This Constructor take priority queue and comparator
   * @param myQueue my priority queue
   * @param comp my current comparator
   */
  ```
- **Constuctor with 2 parameters**
  -> Priortiy queue elemanları vectore eklenir.

  ```
  /**
   * Remove first element of vector queue
   * @return and return removed element.
   */
  @Override
  ```
- **public E dequeue()**
  Ilk eleman silinir- remove(0) ile

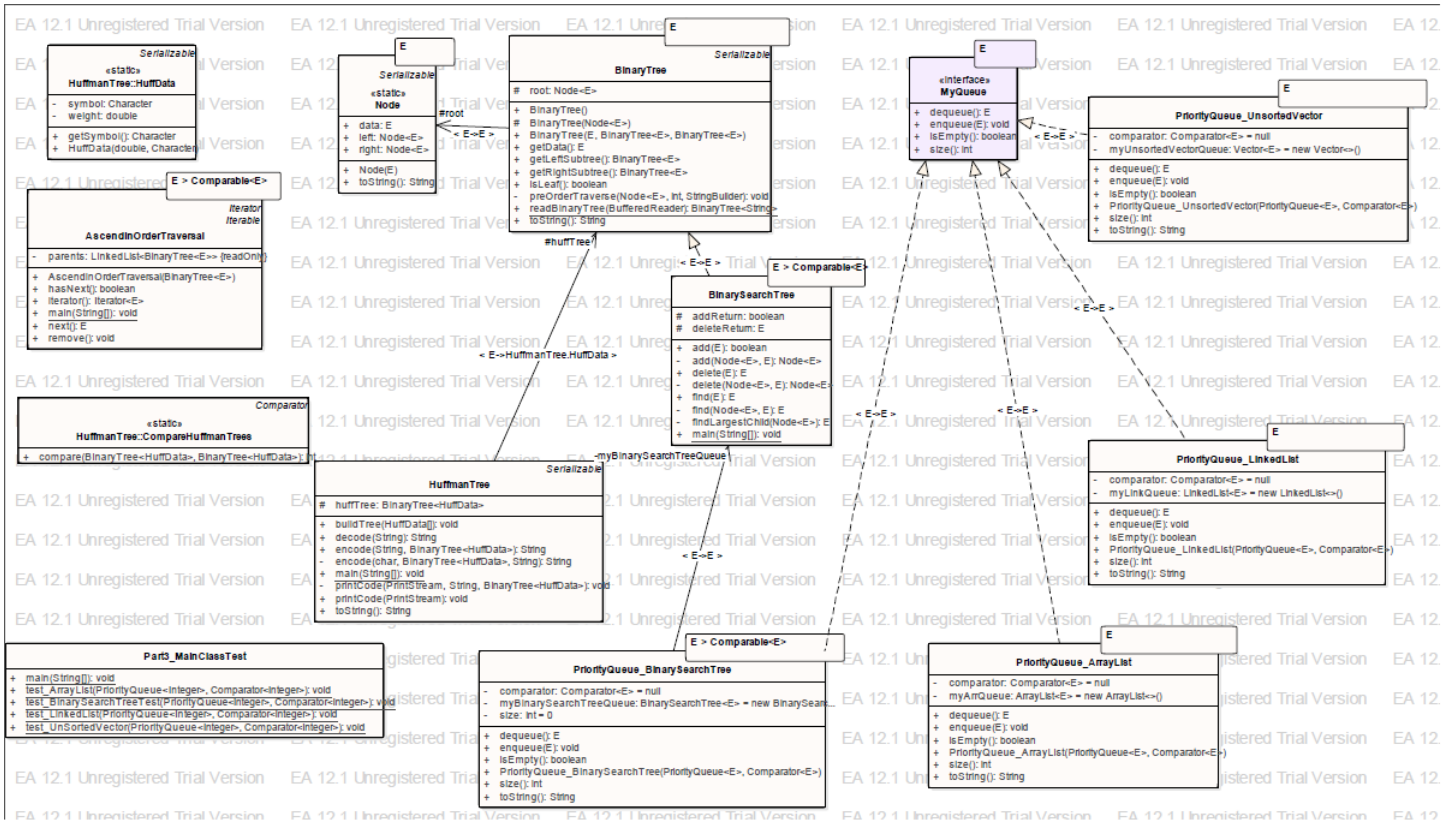
  ```
  /**
   * Add element at end
   * @param element will be add.
   */
  @Override
  ```
- **public void enqueue(E element)**
  Sona eleman eklenir add(size, element) ile.

  ```
  /**
   * This method, look my vector queue is empty.
   * @return if empty true, otherwise false.
   */
  @Override
  ```
- **public boolean isEmpty()**
  Vectorumde eleman var mı ona bakar.

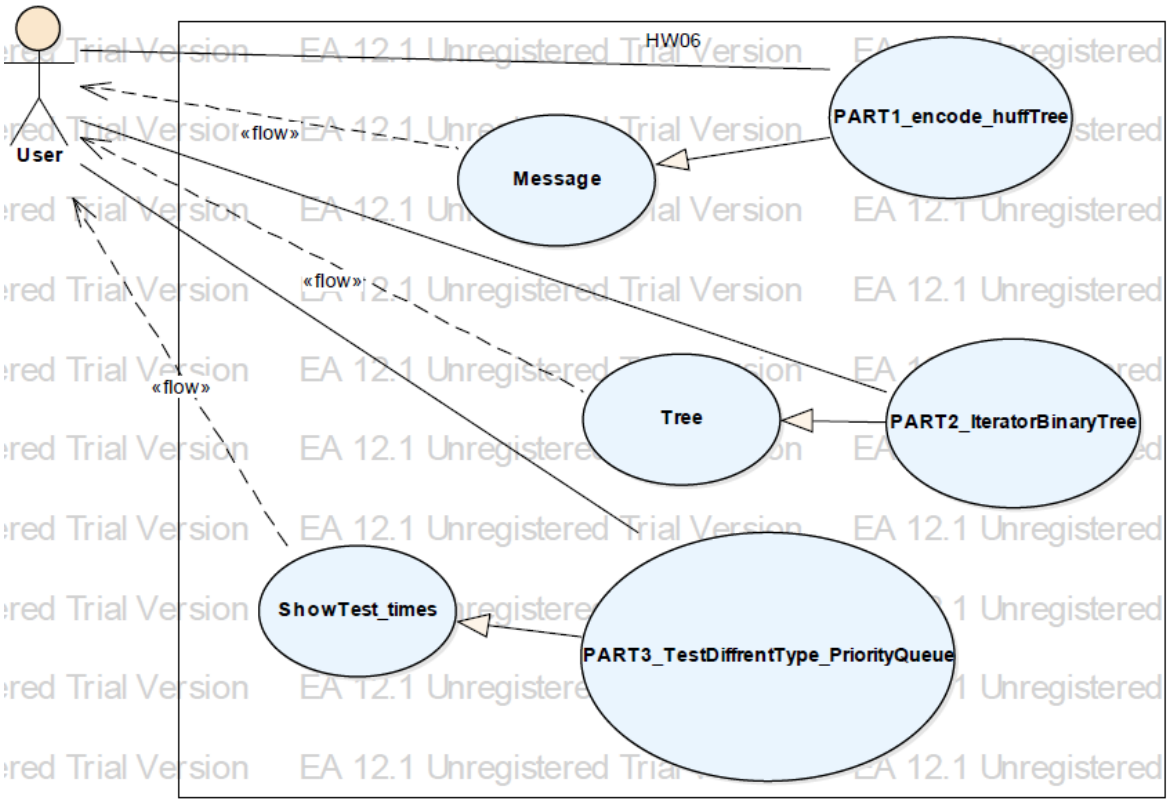
  ```
  /**
   * Return number of my tree queue
   * @return the size of my tree queue.
   */
  @Override
  ```
- **public int size()**
  Vectorumde kaç eleman var onu verir.

- ---
  ```

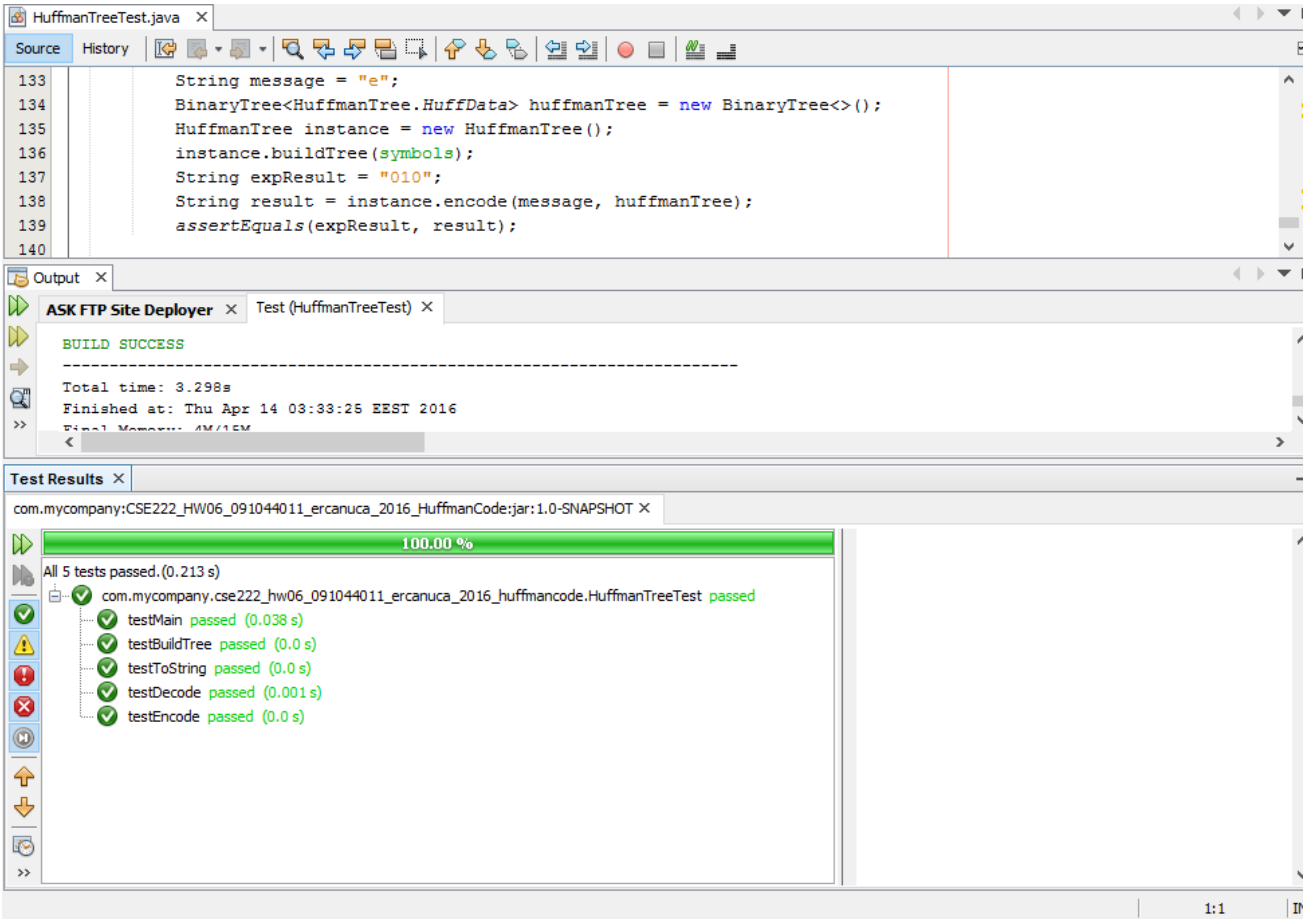
Class Diagramı

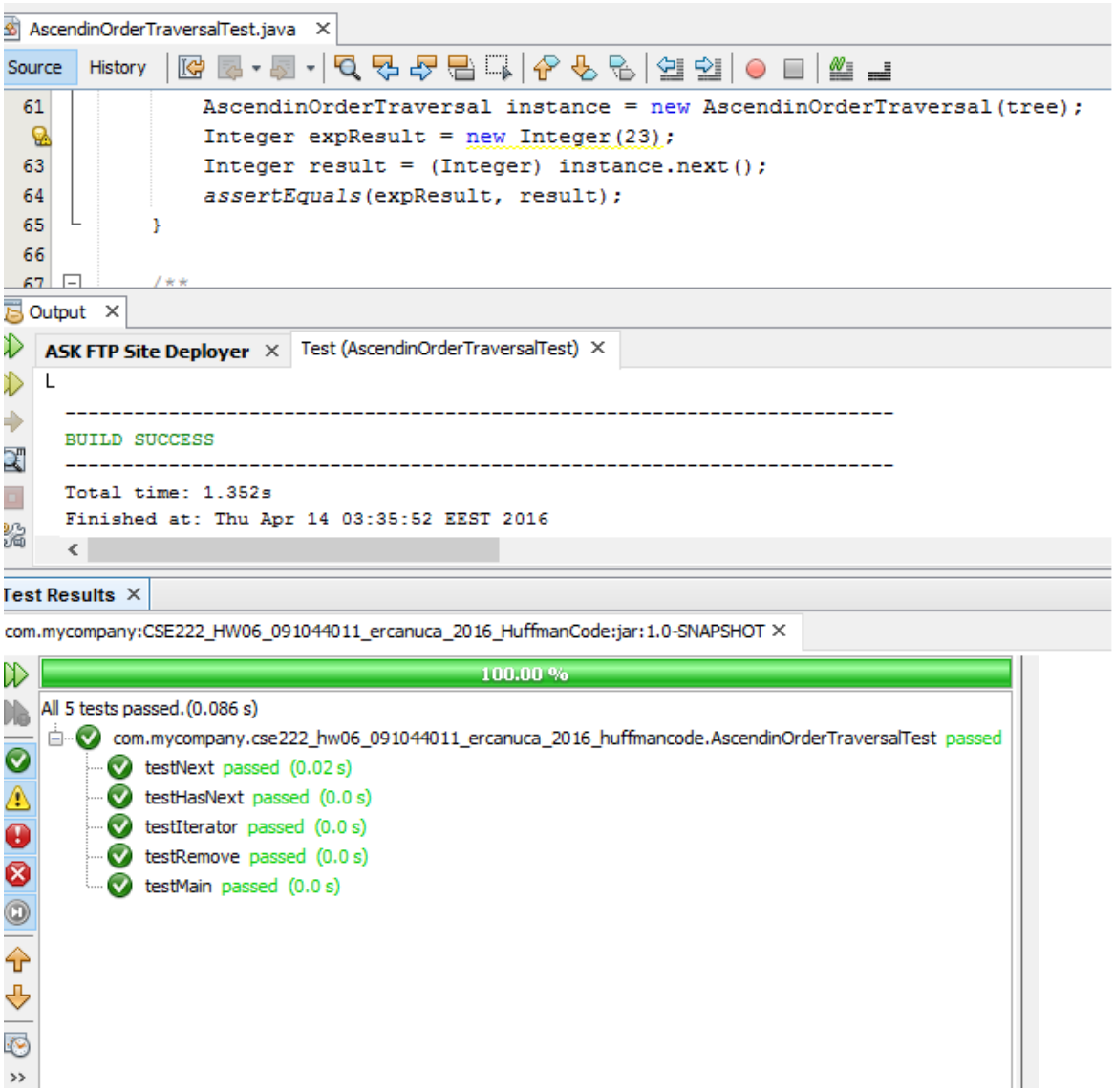


Use Case Diagrams



Tests


HuffmanTreeTest.java
Source
History
133 String message = "e";
134 BinaryTree<HuffmanTree.HuffData> huffmanTree = new BinaryTree<>();
135 HuffmanTree instance = new HuffmanTree();
136 instance.buildTree(symbols);
137 String expResult = "010";
138 String result = instance.encode(message, huffmanTree);
139 assertEquals(expResult, result);
140
Output
ASK FTP Site Deployer
Test (HuffmanTreeTest)
BUILD SUCCESS
Total time: 3.298s
Finished at: Thu Apr 14 03:33:25 EEST 2016
Test Results
com.mycompany:CSE222_HW06_091044011_ercanuca_2016_HuffmanCode:jar:1.0-SNAPSHOT
100.00 %
All 5 tests passed. (0.213 s)
com.mycompany.cse222_hw06_091044011_ercanuca_2016_huffmancode.HuffmanTreeTest passed
testMain passed (0.038 s)
testBuildTree passed (0.0 s)
testToString passed (0.0 s)
testDecode passed (0.001 s)
testEncode passed (0.0 s)
1:1



Main tests

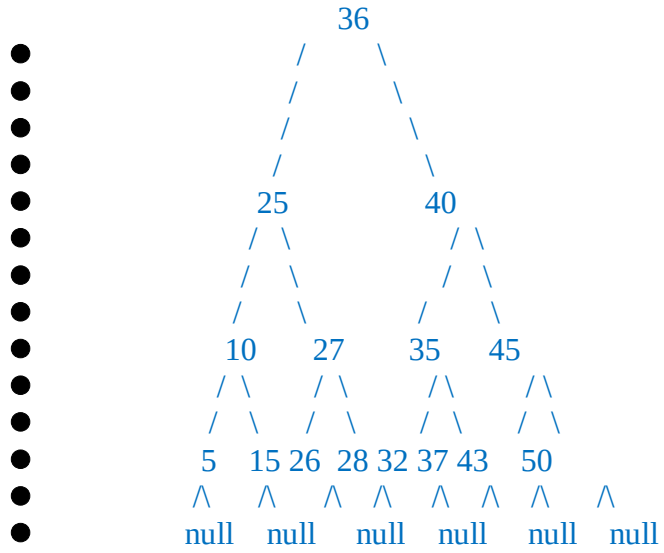
----------PART_1----------

Encoded Codes :
c: 00000
u: 00001
h: 0001
r: 0010
s: 0011
e: 010
i: 0110
n: 0111
b: 100000
g: 100001
p: 100010
y: 100011
o: 1001
a: 1010
l: 10110
d: 10111
v: 1100000
j: 1100001000

q: 1100001001
x: 1100001010
z: 1100001011
k: 11000011
w: 110001
m: 110010
f: 110011
t: 1101
space: 111

Code to Message :
1100001001111111100101000011 : q rg
Message to code :
merhaba ben ercan uca :
1100100100010000110101000001010111100000010011111101000100000010100111111000010000001010

----------PART_1_END----------
----------PART_2----------



36
 25
  10
   5
    null
    null
   15
    null
    null
  27
   26
    null
    null
   28
    null
    null
 40

```
   35
     32
       null
       null
     37
       null
       null
   45
     43
       null
       null
     50
       null
       null


5
10
15
25
26
27
28
36
32
35
37
40
43
45
50
----------------------PART_2_END----------------------
-----------------------PART_3-----------------------
--------------ARRAYLIST_QUEUE_TEST------------------
****************For 1 integer******************
**************Enqueu 1 element*****************
Took approximately 18566 ns
**************Dequeu 1 element*****************
Took approximately 18112 ns
****************End for 1 number******************


*****************For 10 integer*******************
**************Enqueu 10 elements*****************
Took approximately 27169 ns
**************Dequeu 10 elements*****************
Took approximately 11320 ns
************End for 10 numbers**********************


**************For 100 integer***********************
************Enqueu 100 elements**********************
```

Took approximately 153505 ns
**********Dequeu 100 elements**********
Took approximately 68828 ns
**********End for 100 number**********

**********For 1000 integer**********
**********Enqueu 1000 elements**********
Took approximately 1336715 ns
**********Dequeu 1000 elements**********
Took approximately 950463 ns
**********End for 1000 number**********

**********For 10000 integer**********
**********Enqueu 10000 elements**********
Took approximately 4 ms
**********Dequeu 10000 elements**********
Took approximately 14 ms
**********End for 10000 number**********

**********For 100000 integer**********
**********Enqueu 100000 elements**********
Took approximately 21 ms
**********Dequeu 100000 elements**********
Took approximately 1923 ms
**********End for 100000 number**********

**********For 1000000 integer**********
**********Enqueu 1000000 elements**********
Took approximately 265 ms
**********Dequeu 1000000 elements**********
Took approximately 274361 ms
**********End for 1000000 number**********

----------END_ARRAYLIST_QUEUE_TEST----------
----------LINKEDLIST_QUEUE_TEST----------
**********For 1 integer**********
**********Enqueu 1 element**********
Took approximately 20377 ns
**********Dequeu 1 element**********
Took approximately 26716 ns
**********End for 1 number**********

**********For 10 integer**********
**********Enqueu 10 elements**********
Took approximately 6793 ns
**********Dequeu 10 elements**********

Took approximately 14943 ns
**********End for 10 numbers**********

**********For 100 integer**********
**********Enqueu 100 elements**********
Took approximately 62941 ns
**********Dequeu 100 elements**********
Took approximately 64753 ns
**********End for 100 number**********

**********For 1000 integer**********
**********Enqueu 1000 elements**********
Took approximately 573266 ns
**********Dequeu 1000 elements**********
Took approximately 499910 ns
**********End for 1000 number**********

**********For 10000 integer**********
**********Enqueu 10000 elements**********
Took approximately 2 ms
**********Dequeu 10000 elements**********
Took approximately 1 ms
**********End for 10000 number**********

**********For 100000 integer**********
**********Enqueu 100000 elements**********
Took approximately 15 ms
**********Dequeu 100000 elements**********
Took approximately 5 ms
**********End for 100000 number**********

**********For 1000000 integer**********
**********Enqueu 1000000 elements**********
Took approximately 267 ms
**********Dequeu 1000000 elements**********
Took approximately 22 ms
**********End for 1000000 number**********

----------END_LINKEDLIST_QUEUE_TEST----------
----------UNSORTED_VECTOR_QUEUE_TEST----------
**********For 1 integer**********
**********Enqueu 1 element**********
Took approximately 13131 ns
**********Dequeu 1 element**********
Took approximately 12226 ns
**********End for 1 number**********

**********For 10 integer**********
**********Enqueu 10 elements**********
Took approximately 8151 ns
**********Dequeu 10 elements**********
Took approximately 9510 ns
**********End for 10 numbers**********

**********For 100 integer**********
**********Enqueu 100 elements**********
Took approximately 81960 ns
**********Dequeu 100 elements**********
Took approximately 57960 ns
**********End for 100 number**********

**********For 1000 integer**********
**********Enqueu 1000 elements**********
Took approximately 662471 ns
**********Dequeu 1000 elements**********
Took approximately 935067 ns
**********End for 1000 number**********

**********For 10000 integer**********
**********Enqueu 10000 elements**********
Took approximately 3 ms
**********Dequeu 10000 elements**********
Took approximately 12 ms
**********End for 10000 number**********

**********For 100000 integer**********
**********Enqueu 100000 elements**********
Took approximately 15 ms
**********Dequeu 100000 elements**********
Took approximately 1286 ms
**********End for 100000 number**********

**********For 1000000 integer**********
**********Enqueu 1000000 elements**********
Took approximately 171 ms
**********Dequeu 1000000 elements**********
Took approximately 261435 ms
**********End for 1000000 number**********

----------END_UNSORTED_VECTOR_QUEUE_TEST----------
---------BINARY_SEARCH_TREE_QUEUE_TEST----------

**********For 1 integer**********
**********Enqueu 1 element**********
Took approximately 6792 ns
**********Dequeu 1 element**********
Took approximately 1358 ns
**********End for 1 number**********

**********For 10 integer**********
**********Enqueu 10 elements**********
Took approximately 3169 ns
**********Dequeu 10 elements**********
Took approximately 2265 ns
**********End for 10 numbers**********

**********For 100 integer**********
**********Enqueu 100 elements**********
Took approximately 31697 ns
**********Dequeu 100 elements**********
Took approximately 9056 ns
**********End for 100 number**********

**********For 1000 integer**********
**********Enqueu 1000 elements**********
Took approximately 156675 ns
**********Dequeu 1000 elements**********
Took approximately 155316 ns
**********End for 1000 number**********

**********For 10000 integer**********
**********Enqueu 10000 elements**********
Took approximately 1 ms
**********Dequeu 10000 elements**********
Took approximately 10 ms
**********End for 10000 number**********

**********For 100000 integer**********
**********Enqueu 100000 elements**********
Took approximately 23 ms
**********Dequeu 100000 elements**********
Took approximately 1348 ms
**********End for 100000 number**********

**********For 1000000 integer**********
**********Enqueu 1000000 elements**********
Took approximately 172 ms
**********Dequeu 1000000 elements**********

Took approximately 287336 ms
**********End for 1000000 number**********

----------END_BINARY_SEARCH_TREE_QUEUE_TEST---------

----------PART_3_END----------

Ödev github linki
https://github.com/erccanuca/cse222_hw06_BinarySeachTree_HuffmanTree_encoding.git
(Ödev teslim süresi geçince public yapılacak.)