Data Structures 2 - Lab 1
Implementing Binary Heap & Sorting Techniques

Name : ali mohamed ali hamed
ID : 42
Name : mohamed salah abdelrazek
ID : 55

Samples of the code :

```java
public class Sort <T extends Comparable<T>> implements ISort {

    public IHeap heapSort(ArrayList unordered) {
        IHeap sorted = new Heap();
        sorted.build(unordered);

        for (int i=sorted.size()-1; i>=0; i--)
        {
            sorted.extract();
        }
        sorted.insert( element: -1);
        return sorted;
    }

    public void sortSlow(ArrayList unordered) {
        if(unordered !=null){
            int size = unordered.size();
            for (int i=0;i<size-1;i++){
                for (int j=0;j<size-i-1;j++) {
                    T first = (T) unordered.get(size-j-1);
                    T second = (T) unordered.get(size-j-2);
                   if(first.compareTo(second) <0){
                    T temp = (T) unordered.get(size-j-1);
                     unordered.set(size-j-1,unordered.get(size-j-2));
                    unordered.set(size-j-2 , temp);
                   }
                }
            }
            int g=2;
        }
    }
```

```java
void merge( ArrayList<T> arr, int l, int m, int r) {
    int n1 = m - l + 1;
    int n2 = r - m;
    ArrayList<T> left = new ArrayList<>();
    ArrayList<T> right = new ArrayList<>();

    for (int i=0; i<n1; ++i)
        left.add(arr.get(l+i));
    for (int j=0; j<n2; ++j)
        right.add(arr.get(m+1+j));

    int i = 0, j = 0;
    int k = l;
    while (i < n1 && j < n2)
    {
        if ( left.get(i).compareTo(right.get(j))<=0)
        {
            arr.set(k,left.get(i++));
        }
        else
        {
            arr.set(k,right.get(j++));
        }
        k++;
    }
    while (i < n1)
    {
        arr.set(k,left.get(i++));
        k++;
    }
    while (j < n2)
    {
        arr.set(k,right.get(j++));
        k++;
    }
}

void sort(ArrayList arr, int l, int r) {
    if (l < r)
    {
        int m = (l+r)/2;

        sort(arr, l, m);
        sort(arr , m+1, r);

        merge(arr, l, m, r);
    }
}
```

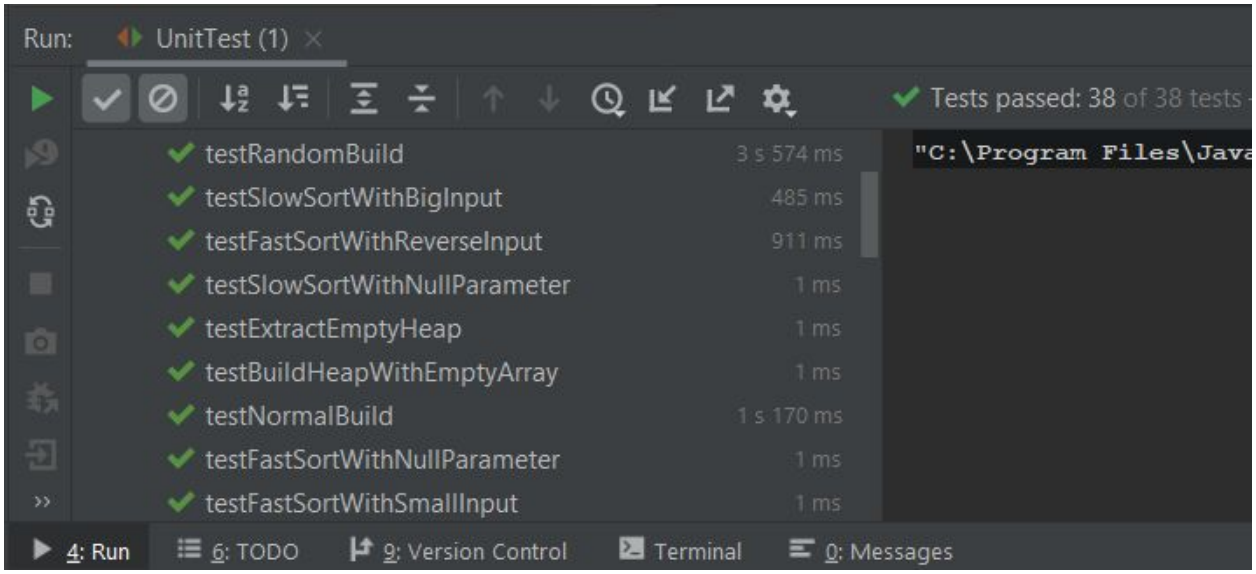
```java
public void heapify(INode<T> node) {
    if(node!=null) {
        INode <T> help =node ;
            if(node.getLeftChild()!=null){
                if(node.getRightChild() !=null && node.getLeftChild().getValue().compareTo(node.getRightChild().getValue()) >0){
                    if(node.getValue().compareTo(node.getLeftChild().getValue())<0) {
                        swap(node, node.getLeftChild());
                        node = node.getLeftChild();
                        heapify(node);
                    }
                }else if(node.getRightChild() !=null){
                    if(node.getValue().compareTo(node.getRightChild().getValue())<0) {
                        swap(node, node.getRightChild());
                        node = node.getRightChild();
                        heapify(node);
                    }
                }else if(node.getRightChild() ==null){
                    if(node.getValue().compareTo(node.getLeftChild().getValue())<0) {
                        swap(node, node.getLeftChild());
                        node = node.getLeftChild();
                        heapify(node);
                    }
                }
            }else
                return;
    }
}
```

```java
public T extract() {
    if(mySize==0){
        return null;
    }
    T root = (T) getRoot().getValue();
    if(mySize!=1){
        swap(getRoot(),nodes.get(mySize-1));
    }
    //nodes.remove(mySize-1);
    mySize--;
    mysize.set(0,mySize);
    heapify(getRoot());
    return root;
}

public void insert(T element) {
    if (element != null) {
    if (element.equals(-1)) {
        mysize.set(0, nodes.size());
        mySize =nodes.size();
    } else {
            mySize++;
            mysize.set(0, mySize);
            INode tmep = new Node( index: mySize - 1, nodes, mysize);
            tmep.setValue(element);
            nodes.add( index: mySize-1,tmep);
            int pos = mySize - 1;
            int parent = (pos - 1) / 2;
            while (nodes.get(pos).getValue().compareTo(nodes.get(parent).getValue()) > 0) {
                swap(nodes.get(pos), nodes.get(parent));
                pos = parent;
                parent = (pos - 1) / 2;
            }
        }
    }
}
```

```java
public void build(Collection<T> unordered) {

    if (unordered != null) {
        nodes.clear();
        Iterator<T> iterator = unordered.iterator();
        mySize = unordered.size();
        mysize.set(0, mySize);

        for (int i = 0; iterator.hasNext(); i++) {
            INode<T> node = new Node(i, nodes, mysize);
            node.setValue(iterator.next());
            this.nodes.add(node);
        }

        int pos = (mySize / 2) - 1;
        for (int j = pos; j >= 0; j--) {
            heapify(nodes.get(j));
        }


    }
}
```

```java
    public <T extends Comparable<T>> Node(int indx ,  ArrayList<INode> nodes,ArrayList <Integer> mysize) {
        this.indx = indx;
        this.nodes = nodes;
        this.mysize =mysize;
    }


    public INode<T> getLeftChild() {
        if(indx*2+1 >= mysize.get(0))
            return null;

        //nodes.size() - mySize;
        return (INode<T>) nodes.get(indx*2+1);
    }

    @Override
    public INode<T> getRightChild() {
        if(indx*2+2 >= mysize.get(0))
            return null;
        return (INode<T>) nodes.get(indx*2+2);
    }

    @Override
    public INode<T> getParent() {
        if(mysize.get(0)==1)
            return null;
        return(INode<T>) nodes.get((indx-1)/2);
    }

    @Override
    public T getValue() { return this.value; }


    public void setValue(Comparable value) { this.value = (T) value; }
}
```

Tests :

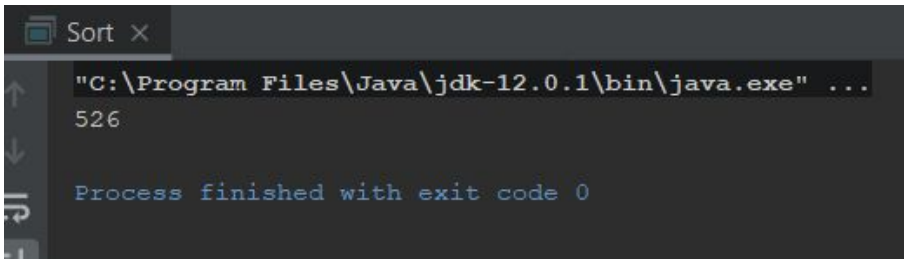


Test the time of different kinds of sort

By creating an array of size 10^4 with random numbers.

```
ArrayList<Integer> arr = new ArrayList<>();
Random random = new Random();
for(int i=0;i<10000;i++){
    Integer num = random.nextInt( bound: 1000000);
    arr.add(num);
}
```

Bubble Sort : The time was 526ms



Merge Sort: 10^4

```
Run:  Sort
"C:\Program Files\Java\jdk-12.0.1\bin\java.exe
29

Process finished with exit code 0
```

For 10^5

```
un:  Sort
"C:\Program Files\Java\jdk-12.0.1\bin
306

Process finished with exit code 0
```

For 10^6

```
un:  Sort
"C:\Program Files\Java\jdk-12.0.1\bin\java.e
1386

Process finished with exit code 0
```

Heap Sort :

For 10^4

```
Run:  Sort
"C:\Program Files\Java\jdk-12.0.1\bin\java.
40
```

For 10^5

```
Run:  Sort
"C:\Program Files\Java\jdk-12.0.1\bin\
478

Process finished with exit code 0
```

10^6

```
Sort
"C:\Program Files\Java\jdk-12.0.1\bin\java.exe"
3142

Process finished with exit code 0
```