Sorting Generically

Thursday, October 29, 2015 11:35 AM

Assume an ArrayList 'list'

```
insertionSort(...)
    for(int i=0; i<list.size(); i++) {
        int j=i;
        while(j>0 && list.get(j) < list.get(j-1)) {
            swap(list, j, j-1);
        }
    }
```

(*) ? ? O(1)

```
private void swap(ArrayList<E> list, int i, int j) {
    E temp = list.get(i);
    list.set(i, list.get(j));
    list.set(j, temp);
}
```

```
new LinkedList<String>
new Stack<Integer>
```

defined

```
public class Sorter<E> {

    public void insertionSort(ArrayList<E> list) {
        (*)
    }

    private void swap(ArrayList<E> list, int i, int j) {
        E temp = list.get(i);
        :
    }
}

main() {
    Sorter<String> sorter = new Sorter<String>();
    sorter.insertionSort(listOfStrings);
}
```

```
}
```

0 → equal
positive → greater
negative → less than

```
public interface Comparable<E> {
    public int compareTo(E that);
}
```

java.lang.Comparable

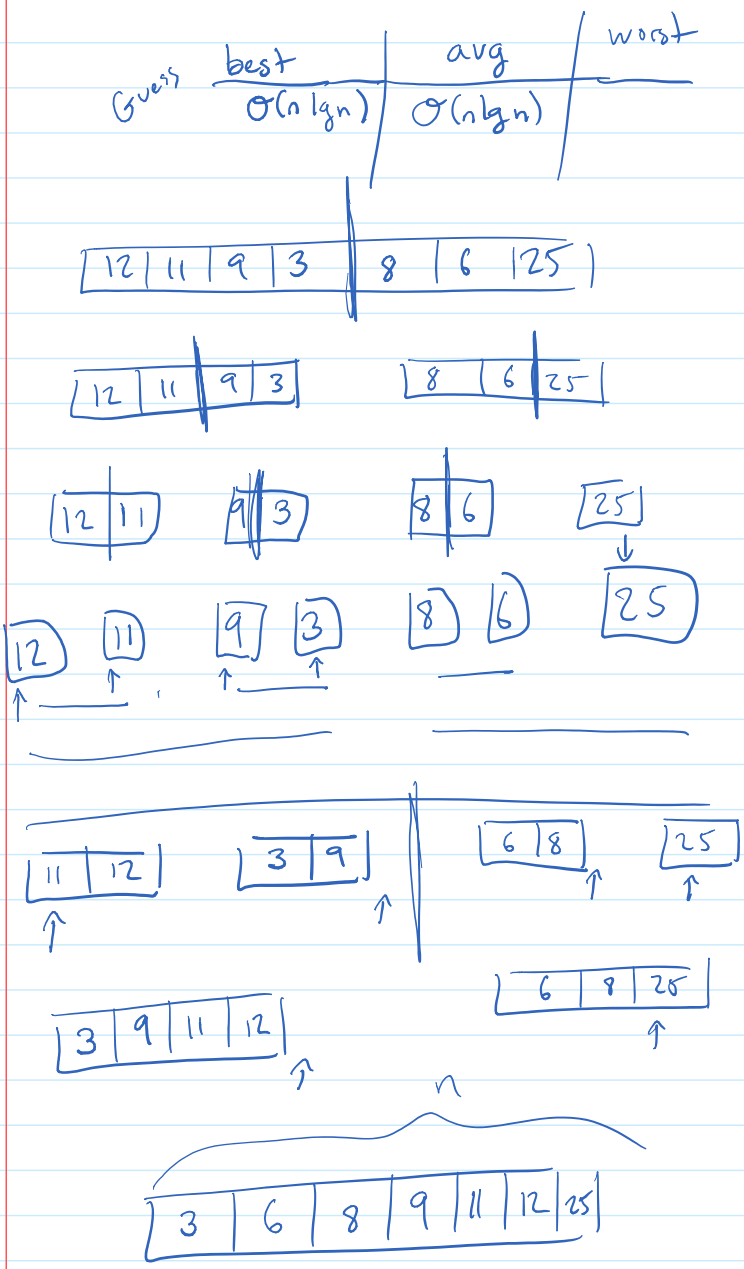
```
Integer theInt = 1;
Double theDouble = 2;
theInt.compareTo(theDouble)
```

theInt > or < or == theDouble → negative

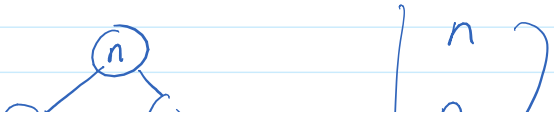
"Dar".compareTo("Zach") → negative
String String

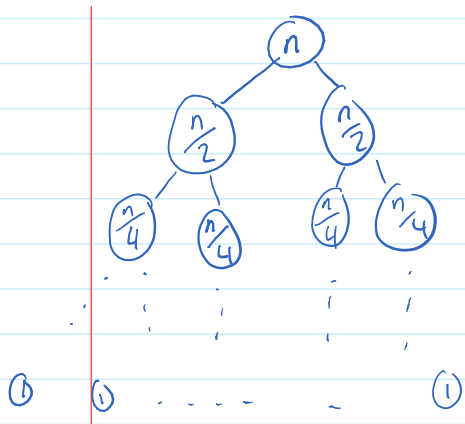
```
public class Sorter<E extends Comparable<E>> {

    public void insertionSort(ArrayList<E> list) {
        for (int i=0; i<list.size(); i++) {
            int j=i;
            while (j>0 && list.get(j).compareTo(list.get(j-1))<0) {
                swap(list, j, j-1);
            }
        }
    }

    swap(...)

public class Integer implements Comparable<Integers> {
    int theInt
    @Override
    public int compareTo(Integer that) {
        return this - that;
    }
}

public class String implements Comparable<String> {
    :
    public int compareTo(String that) {
        :
    }
}
```

```
Mergesort (list):    // Divide & Conquer
    (left, right) = divide(list)
        Mergesort(left)
        Mergesort(right)
    return Merge(left, right)   //O(n)
End Function
```



best	worst	avg
$O(n \lg n)$	$O(n \lg n)$	$O(n \lg n)$





n
n
n
h

$n \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdots = 1$ (h factors)

$n \cdot \left(\frac{1}{2}\right)^h = 1$

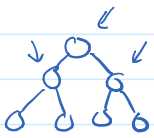
$\lg n = 2^h$

$h = \lg n$

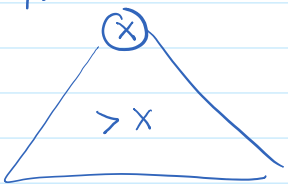
$O(n \cdot \lg n)$ time

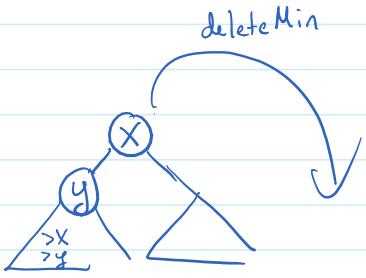
	Avg	
Insertion sort	$\Omega(n \lg n)$	$\rightarrow O(n^2)$
Quicksort	$\Omega(n \lg n)$	
Merge sort	$\Omega(n \lg n)$	

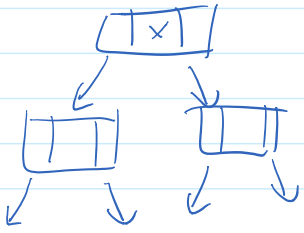
Heap



Min Heaps







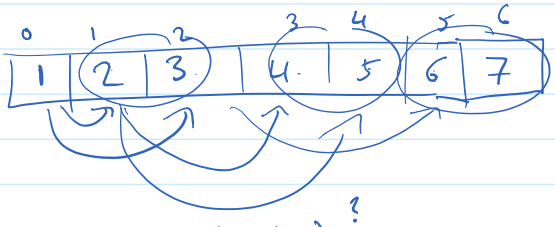
Min Heap

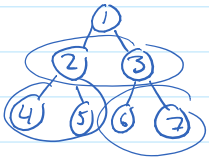
deleteMin() fast

insert(...)

getMin()

heapify(list) fast





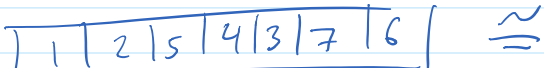
left(index) = 2*index + 1 ?

right(index) = left(index) + 1

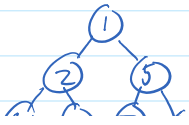
$f(0) = 1$

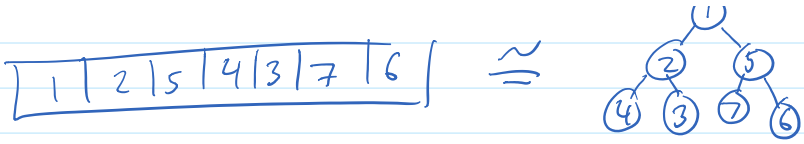
$f(1) = 3$

$f(2) = 5$



$\simeq$




1 2 5 4 3 7 6

1
2
5
4
3
7
6