Module 2

Exercise 2.2

1.

L=0
R=9
P=4

2.

L=0
R=3
P=3

L=0
R=9
P=4

3.

L=0
R=2
P=2

L=0
R=3
P=3

L=0
R=9
P=4

4.

L=0
R=1
P=1

L=0
R=2
P=1

L=0
R=3
P=3

L=0
R=9
P=4

5.

L=0
R=0

(Top of Stack)

L=0
R=1
P=1

L=0
R=2
P=2

L=0
R=3
P=3

L=0
R=9
P=4

6.

L=0
R=1
P=1

(recursively goes dow)

L=0
R=2
P=2

L=0
R=3
P=3

L=0
R=9
P=4

7.

L=0
R=2
P=2

L=0
R=3
P=3

L=0
R=9
P=4

8.

L=0
R=3
P=3

L=0
R=9
P=4

9.

L=0
R=9
P=4

10.

L=2
R=1

11.

L=3
R=2

12.

L=4
R=3

13.

L=5
R=9
P=8

14.

L=5
R=7
P=5

L=5
R=7
P=8

2.7

1. A G I K B C D M ———— LC=0 RC=4
Merge from left into merge spac

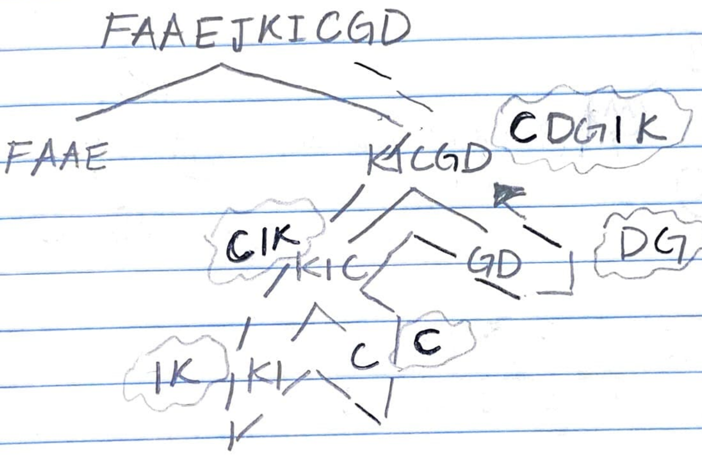
A G I K B C D M A ———— LC=1 RC=4
Merge from right

A G I K B C D M A B ———— LC=1 RC=5
Merge from right

A G I K B C D M A B C ———— LC=1 RC=6
Merge from right

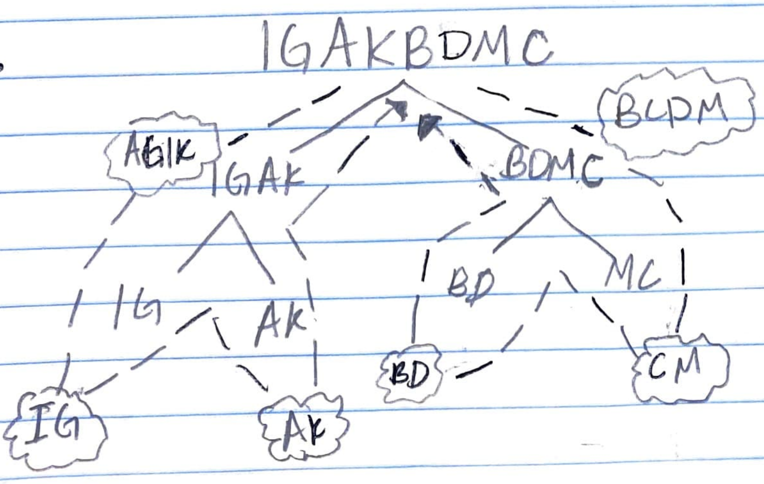
A G I K B C D M A B C D ———— LC=1 RC=7
Merge from left

2.



4. Yes you can sort either side first

5.



3. The line that we need to change to create merge space once is line :

Comparable [] mergeSpace = new Comparable [right-left+1]

to: Comparable [] MergeSpace = new Comparable [data.length]

Exercise 2.4

We don't need to check whether RC becomes negative because we already have a check for if data[RC] is greater than the partition element and the indice wouldn't dip below 0 because that would cause an out of bounds error.

Exercise 2.12

We know $n! <= n^n$

We also know $n! >= \left(\frac{n}{2}\right)^{n/2}$

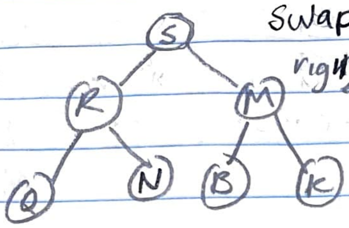
So $\log n! >= \left(\frac{n}{2}\right)(\log n - \log 2)$ is around the same as $n \log n$

So $O(\log n!) = O(n \log n)$

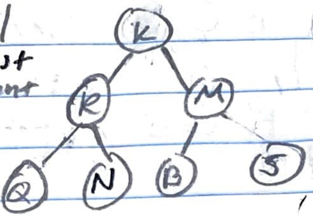
Exercise 2.8 Mergesort is a stable algorithm because when merging two halves we use the line if data[left] <= data[right] and with that we favor left halve values over right halve values if they are equal.

Exercise 2.9

1.

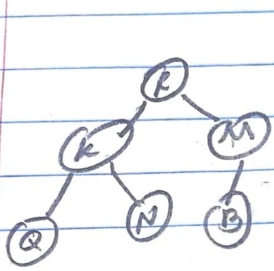


Swap w/ rightmost element

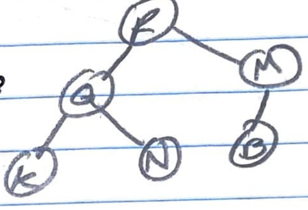


(3) Violates heap order ?
Swap w/ largest

(extract from heap)



(4) Violates heap order?
⇒ Swap



2.

R	Q	M	K	N	B
0	1	2	3	4	5

M >

3. $n = \sum_{i=0}^{m} (1 + 2^1 + 2^2 + ... + 2^m)$

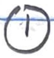
Exercise 2.6

Algorithm: quickSort Recursive (data, left, right)
Input: data array, indices left and right

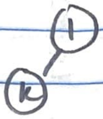
```
IF (right - left)+1 < 10
  for i = 0 to right
    for j = i to j > 0 & data[left] < data[right]
      Int temp = data[j]
      data[j] = data[j-1]
        data[j-1] = temp
    end for
  end for
end if
else
   If left+1 == right
     If data[left] > data[right]
       Int temp = data[left]
       data[left] = data[right]
       data[right] = temp
    end if
  end if
  else if (left < right)
    Partition pr;
    quickSortThreeWay (data, left, pr.left -1)
    quickSortThreeWay (data, pr.right +1, right)
  end else if
end else
```

Exercise 2.10

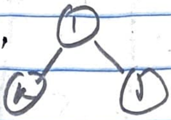
1.



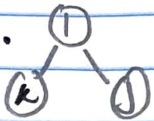
2.



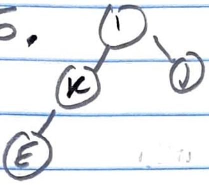
3.



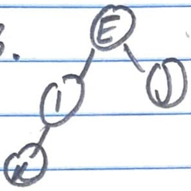
4.



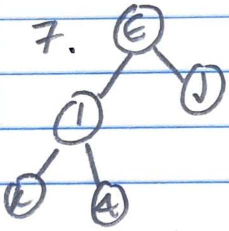
5.



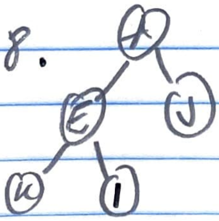
6.



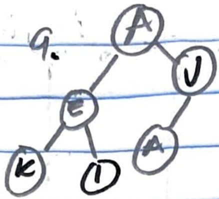
7.



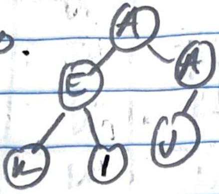
8.



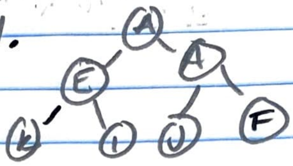
9.



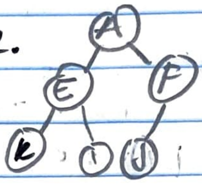
10.



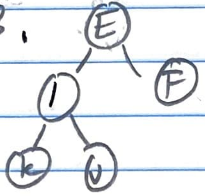
11.



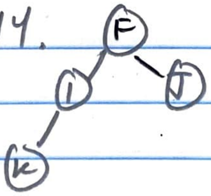
12.



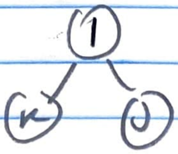
13.



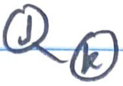
14.



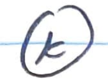
15.



16.



17.



Exercise 2.11

						S
					R	S
				Q	R	S
			N	Q	R	S
		M	N	Q	R	S
	k	M	N	Q	R	S
B	k	M	N	Q	R	S

Exercise 2.5

The work complexity is 8/10 and you have Unequal Sides 80 % w 20 %.