

Problem 1 & Problem 2

Problem 1.

The asymptotic running time is $\Theta(n) + \Theta(n^2)$.
 $= \Theta(n^2)$.

Problem 2.

Algorithm merge(a, b)

Input: Two sorted arrays.

Output: A combined sorted array

sortedArray \leftarrow new int[a.length + b.length] -
i \leftarrow 0
x \leftarrow 0

y \leftarrow 0

while ($x < a.length \ \&\& \ y < b.length$) do

if $a[x] < b[y]$ then

sortedArray[i] = $a[x]$
x = x + 1

else if $b[y] < a[x]$ then

sortedArray[i] = $b[y]$

else

sortedArray[i] = $a[x]$

x = x + 1

y = y + 1

i = i + 1

while ($x < a.length$) do

sortedArray[i] = $a[x]$

x = x + 1

i = i + 1

```

while (y < b.length) do
    SortedArray[i] = b[y]
    y = y + 1
    i = i + 1
return SortedArray

```

b) The asymptotic runtime is $O(n)$.

Problem 3 & Problem 5

Problem 3

3. The asymptotic runtime of recursive algorithm can be calculated by determining the number of self calls. n

$$\# \text{ of self calls} = n$$

$$\text{Therefore the runtime} = \Theta(n) \\ \text{or } \underline{\underline{O(n)}}$$

Problem 5

$$T(n) = T\left(\frac{n}{2}\right) + n;$$

$$T(1) = 1$$

Using the Master Formula;

$$a = 1, b = 2, c = 1, k = 1$$

$$1 < 2^1$$

$$a < b^k$$

The asymptotic runtime is $\underline{\underline{\Theta(n)}}$.

Problem 4

```
public LinkedList<Set<Integer>> powerSet(LinkedList<Integer>
inputList){
    LinkedList<Set<Integer>> resultSet = new LinkedList<>();
    Set<Integer> emptySet = new HashSet<>();
    resultSet.add(emptySet);
    while(!inputList.isEmpty()){
        Integer f = inputList.removeFirst();
        int size = resultSet.size();
        //Set<Integer> currentSet;
        for(int i = 0; i < size; i++){
            Set<Integer> union = new HashSet<>();
            Set<Integer> currentSet = resultSet.get(i);
            union.add(f);
            union.addAll(currentSet);
            resultSet.add(union);
        }
    }
    return resultSet;
}
```