Homework 4

Problem 1:

Use the Master Theorem — if possible — to solve the following recurrences. In some cases, the recurrence can be solved but the Master Theorem cannot be applied. Use the version of the MT in the book, not the one at Wikipedia.

1. $T(n) = 3T(n/2) + n \log n$.

2. $T(n) = 4T(n/2) + \sqrt{n} + 2$

3. $T(n) = 2^n T(n/3) + n^2$

4. $T(n) = 3T(n/4) + n \log n$.

5. $T(n) = \frac{1}{2}T(\frac{2}{3}n) + \sqrt{n}$.

6. $T(n) = 2T(n/4) + \sqrt{n} + 2$

7. $T(n) = 4T(n/3) + \log(n)n$.

8. $T(n) = 3T(n/3) + \sqrt{n}$.

9. $T(n) = 2T(n/2) + n \log(n)$

10. $T(n) = 16T(n/4) + n$

Problem 2:

Given the following two divide-and-conquer algorithms, describe their run times with a recurrence relation and apply, if possible the MT.

```
import numpy

def bs(array):
    lng = len(array)
    array[0], array[-1] = min(array[0], array[-1]), max(array[0], array[-1])
    if len(array) <= 2:
        return
    else:
        bs(array[ : 2*lng//3])
        bs(array[lng//3 : ])
        bs(array[ : 2*lng//3])
        bs(array[lng//3 : ])
    return

if __name__ == '__main__':
    array = numpy.random.randint(0, 100, 20)
    bs(array)
    print(array)
```

Notice that the code above only works with numpy arrays as slices will create new arrays.

```
def bs(number):
    if number <=2:
        return number
    else:
        return (bs(number-1)+bs(number-2))/3
```

We can assume that number is a positive integer.