1. My sources: https://www.geeksforgeeks.org/sort-an-array-of-0s-1s-and-2s/ and https://www.quora.com/How-can-I-solve-the-Dutch-National-Flag-problem-for-four-colors
Dutch national flag algorithm. Have four pointers. Low, mid, mid2, and high. Low, mid, and mid2 all start at 0. High starts at the length of the array - 1. While mid2 is less than high get the value of mid2 in the array. If houses[mid2] is 'Gryffindor' then swap the item at low and mid2, increment low, if mid is less than low then increment mid, if mid2 is less than low then increment mid2. This creates a section for 'Gryffindor' where low is the value right after the section so when we want to add to that section later we'll use low. If either of the mids is lower than low the 'Gryffindor' section wouldn't be the lowest one so we have to increment one or both of them. If houses[mid2] is 'Hufflepuff' just increment mid2. This is because mid2 holds the 'Hufflepuff' section like how low holds the 'Gryffindor' section. If houses[mid2] is 'Ravenclaw' swap the item at mid and mid2 then increment both. This is so that 'Hufflepuff' is always after 'Ravenclaw'. If houses[mid2] is 'Slytherin' swap the item at high and mid2 then decrement high. This always puts 'Slytherin' at the end. My "pseudocode" is this:

```
houses = ['h','h','g','r','r','h','h','r','r','g','h','s','h','h','h','r','s','s','h','g','g','r','s','g','g']

low = 0
high = len(houses) - 1
mid = 0
mid2 = 0

while mid2 <= high:
    if houses[mid2] == 'g':
        houses[low], houses[mid2] = houses[mid2], houses[low]
        low += 1
        if mid < low:
            mid += 1
        if mid2 < low:
            mid2 += 1
    elif houses[mid2] == 'h':
        mid2 += 1
    elif houses[mid2] == 'r':
        houses[mid], houses[mid2] = houses[mid2], houses[mid]
        mid += 1
        mid2 += 1
    else:
        houses[mid2], houses[high] = houses[high], houses[mid2]
        high -= 1

print(houses)
```

2.
   a. The height of the tree is $\log_4 n$ and the work done at each level is ((n^2) * (3^i)) / (16^i) the (3^i) / (16^i) decreases over time so the sum of the work done is (n^2)
   b. The height of the tree is n because n is decremented presumably until it reaches 0. The work done at each level is O(n^2) because the same amount of work is done at n^2 as it is at (n-1)^2. So, the sum of the work done is n^3
   c. The work done at each level is 3n because n/4 + n/3 + n/6 + n/8 + n/8 = n but each branch is multiplied by 3. The height of the tree is logn because the amount of work done does not increase or decrease. So the sum of the work done is 3nlogn or O(nlogn)

d. The height of the tree is n because n is decremented presumably until it reaches 0. The work done at each level is O(1) because it's just 5 without any n. so the sum of the work done is (5^i)*n which is just O(n)

3.

a. I'm guessing that A(n) = O(n). For the base case use 1. A(1) = A(1-1) + 1 = 0 + 1 = 1. To prove A(n) = O(n) we have to show that A(n) <= cn. For the base case this is true when c >= 1. For the induction hypothesis I'll say that A(n) <= cn for all n. So I'll assume that A(n - 1) <= c(n-1) <= cn. So c(n – 1) + 1 <= cn. Which becomes cn – c + 1 <= cn. Subtract cn and add c to both sides and you get 1 <= c. so A(n) = O(n) for all n where c is greater than 0.

b. I'm guessing that B(n) = O(n). For the base case use 1 which I know equals 0. To show B(n) = O(n) we have to show that B(n) <= cn. For the base case this is true when c >= 0. For the induction hypothesis I'll say that B(n) <= cn for all n. So I'll assume that B(n - 5) + 2 <= c(n – 5) + 2 <= cn. So cn - c5 + 2 <= cn which becomes 2 <= c5 which then becomes c >= 2/5. So to make the base case and the induction hypothesis true B(n) = O(n) for all n where c >= 2/5

c. I'm guessing that C(n) = O(1). For the base case if C(n) % 6 is 0 then it equals 1 if it's 1 then it equals 2, if it's 2 then it equals 2, if it's 3 then it equals 1, if it's 4 then it equals 1/2, if it's 5 then it equals ½. All of these cases are less than or equal to c*1 when c = 2. For the induction hypothesis I'll use C(n + 6^i) where i = 1 and show that all of those cases are the same as C(n) where n is 0 through 5 like the base case. C(6) = 1, C(7) = 2, C(8) = 2, C(9) = 1, C(10) = 1/2, C(11) = ½. These values are also less than 1* c where c = 2

d. I'm guessing that D(n) = O(1). For the base case I'll use 0 which gives D(0) = 0 <= 1*c. which is true for any c. for the induction step I'll use n+1. This will build on the base case so D(0 + 1) = D(1) = 1 / (2-D(1-1)) = 1 / (2-0) = ½ . For both of these values it is shown that D(n) = O(1) or D(n) <= 1*c for c = 1.