**CSCI 2300 — Algo**
**Homework 4**
Hayden Fuller

- **Q1**

  (a) $T(n) = 8T(n/4) + O(n)$
  $a = 8$, $b = 4$, $d = 1$, $\log_b a = 1.5 < d$, $O(n^{1.5})$

  (b) $T(n) = 2T(n/4) + O(\sqrt{n})$
  $a = 2$, $b = 4$, $d = .5$, $\log_b a = .5 = d$, $O(n^{0.5} \log n)$

  (c) $T(n) = T(n-4) + O(n^2)$
  c

  (d) $T(n) = T(\sqrt{n}) + O(n)$
  d

- **Q2** Let A be an array of n integers, and let R be the range of values in A. $R = \max(A) - \min(A)$. Give O(n+R) time algorithm to sort all the values in A.
  count sort. Create an array "count" length R and for each item in A, add one to that index in count (compensating for the min value offset, count[0] holds the smallest value) then go through count and, to an output array, add the index (again, compensating for the min value offset) count[i] times.

  this code isn't exactly right, it's actually $O(n + \max(A))$, because it's 11:58 and I don't have time to compensate for the min value offset and I need to turn this in, but it's close enough to demonstrate the algorithm.

```
def sort(arr):
    size=len(arr)
    output=[0]*size

    maxi=arr[0]
    mini=maxi
    for i in arr:
        if i>maxi: maxi=i
        if i<mini: mini=i

    count=[0]*(maxi+1)
    for i in range(size):
        count[arr[i]] += 1
    for i in range(1, maxi+1):
        count[i]+=count[i-1]
    i = size - 1
    while i >= 0:
        output[count[arr[i]]-1]=arr[i]
        count[arr[i]]-=1
        i-=1
    for i in range(size):
        arr[i]=output[i]
```

- **Q3**

  (a) ceiling($n/2$) at the first level, and ceiling(ceiling($n/2$)/2) on the second, etc. This eventually sums to $n-1$ comparisons in the worst case

  (b) this could be done by 1) returning a pair at each step, allowing each comparison to be of 4 numbers, where it then returns the pair of the smallest 2 of the 4.
  or 2) you can run down the tree again, looking at all values compared to your minimum, keeping track of the minimum of those values, since the second smallest number must have been compared to the smallest at some point.

  (c) using the second method above, we run through ceiling($\log_2 n$) $-$ 1 levels of the tree (ignoring the root), and add this to the origional $n-1$ that found the minimum. ceiling($\log_2 n$) $-1+n-1 = n+$ceiling($\log_2 n$) $-$ 2.