CSCI 3104 Assignment 3

10:00 - 10:50 Wanshan

Jackson Chen

14 February 2016

1.
```
def smallestR(a1, a2, r):
    smallestCounter = 1
    i = 0
    j = 0
    n = len(a1)
    while i < n and j < n:
        if a1[i] < a2[j]:
            if smallestCounter == r:
                return a1[i]
            smallestCounter += 1
            i += 1
        else:
            if smallestCounter == r:
                return a2[j]
            smallestCounter += 1
            j += 1

    if i == n:
        while j < n:
            if smallestCounter == r:
                return a2[j]
            smallestCounter += 1
            j += 1
    else:
        while i < n:
            if smallestCounter == r:
                return a1[i]
            smallestCounter += 1
            i += 1
```

The running time of this algorithm is $O(n)$

2. (a) Each partition is at least $\lfloor\sqrt{n}\rfloor$ in length.

(b)

$$\Theta((2\lfloor\sqrt{n}\rfloor)^2) = \Theta(4n) = \Theta(n)$$

(c)

$$T(n) = T(\lfloor\sqrt{n}\rfloor) + T(n - \lfloor\sqrt{n}\rfloor) + O(n)$$

3. (a)

$$(B_1, B_2), (B_3, B_4), (B_5, B_6), (B_7, B_8)$$

$$(B_2, B_4), (B_5, B_8)$$

$$(B_2, B_8)$$

$$\boxed{B_2}$$

(b) For the first round of weighting, the algorithm does $\frac{n}{2}$ weighings, and in the next round $\frac{n}{4}$ weighings. The number of weighing is halved as the rounds progress. As a result, the total number of rounds done is $\log_2 n$. The total number of weighings can be represented by a geometric series with first term $\frac{n}{2}$ and ratio $\frac{1}{2}$.

$$S = \frac{n}{2} * \frac{\frac{1}{2}^{\log_2 n} - 1}{\frac{1}{2} - 1} = \frac{n}{2} * \frac{n^{-1} - 1}{-\frac{1}{2}} = \frac{n}{2} * (-2\frac{1}{n} + 2)$$

$$S = n - 1$$

Therefore the number of weighings will not exceed n.

(c) The second heaviest ball must have been compared with the heaviest weight at some point during the algorithm. Checking all of weights that were compared with the heaviest weight requires an additional $\log_2 n$ of weighings.

(d) The elements that were compared with the heaviest weight B_2 are B_1, B_4, B_8. We must find the maximum in this dataset. So we would compare B_1 to B_4, B_4 to B_8, and B_1 to B_8 to find that B_8 is the second largest weight.

4. The algorithm randomly chooses a pivot, and partitions the array around that pivot. Then recursively call the partition with the larger cumulative weight (including the pivot). However, it also increases the pivot weight by the weight of the eliminated partition to account for the loss of weight. The time complexity is $O(n)$.