Homework 5
UCLA-CS180-S18

Quentin Truong

May 28, 2018

1 Question 1

```
function find_common_interlacing_subsequence(x, y)
    nums = []
    for i in range(0, min(len(x), len(y)))
        nums.append(x[i])
        nums.append(y[i])

    for i in range(min(len(x), len(y)), len(x))
        nums.append(x[i])

    for i in range(min(len(x), len(y)), len(y))
        nums.append(y[i])

    maxes = [0] * len(nums)
    subsequences = [] * len(nums)
    for curr in range(0, len(nums))
        for other in range(0, curr)
            if nums[curr] > nums[other] AND maxes[curr] <= maxes[other]
                maxes[curr] = maxes[other] + 1
                subsequences[curr] = subsequences[other] + nums[curr]

    return subsequences[get_index_of_max(maxes)]
```

2 Question 2

For the n=1 level and ignoring the rest of the tree (the children of a single quaternary tree), the expectation of the number of nodes remaining may be modeled as a binomial random variable with $n = 4^{level}$ and $p = 1/2$. Since the expectation of a binomial random variable is $E[X] = np$, the expectation of the number of nodes connected to the parent for the n=1 level is equal to $E[X_1] = 4^1 * 1/2 = 2$.

The expectation of the number of nodes connected to the parents for the n=2 level is the expected remaining number of nodes from the first multiplied by expected remaining number of nodes for a single quadtree. This is because the nodes in the second row will only have as many parents as the expected remaining nodes from the first row. Moreover, each parent can be viewed independently as its own quaternary tree, thus it will have as many remaining children nodes as the first row. $E[X_2] = E[X_1] * E[X_1]$

This pattern continues on for all remaining rows. When we sum this up for the entire tree, we obtain $E[X] = 1 + E[X_1] + ... + E[X_n]$. The 1 in front comes from $E[X_0]$, which is the root.

In summary, we determine the number of nodes still connected to the root after deletion of the edges in the following way.

$$\begin{aligned}
E[X_1] &= 4^n * 1/2 = 4 * 1/2 = 2 \\
E[X_2] &= E[X_1] * E[X_1] = 2 * 2 = 4 \\
E[X_n] &= 2^n \\
E[X] &= \sum_{i=0}^{n} E[X_i] \\
&= \sum_{i=0}^{n} 2^n
\end{aligned}$$

3 Question 3

```
function find_matches(apples, oranges)
    find_matches(apples, oranges, 0)

function find_matches(apples, oranges, offset)
    apple_index = random(0, len(apples))
    orange_index = null

    for i in range(0, len(oranges))
        if is_equal_weight(apples[apple_index], oranges[i])
            orange_index = i

    apples_less = []
    apples_greater = []
    oranges_less = []
    oranges_greater = []

    for i in range(0, len(oranges))
        if is_less_weight(apples[apple_index], oranges[i])
            oranges_less.append(oranges[i])
        if is_greater_weight(apples[apple_index], oranges[i])
            oranges_greater.append(oranges[i])

    for i in range(0, len(apples))
        if is_less_weight(oranges[orange_index], apples[i])
            apples_less.append(apples[i])
        if is_greater_weight(oranges[orange_index], apples[i])
            apples_less.append(apples[i])

    if len(apples_more) > 0
        matches.append(find_matches(apples_more, oranges_more, offset + len(apples_less)
    if len(apples_less) > 0
        matches.append(find_matches(apples_less, oranges_less, offset))

    matches.append((offset + apple_index, offset + orange_index))
    return matches
```

4 Question 4

```
hashtable universe U: all m-size combinations of all words
hashtable r: Piazza says not to specify this

function detect_plagiarism m documents
    flag = [0] * len(documents)
    init hashtable

    for document in documents
        for i in range(0, len(document) - m + 1)
            hashtable.insert(document.substring(i, i + m))

    for document_index in range(0, len(documents))
        for i in range(0, len(document) - m + 1)
            if hashtable.find(document.substring(i, i + m))
                flag[document_index] = 1

    return flag
```