Foundation of Algorithms

605.421.83

Programming Assignment 3

Name: Guan Yue Wang

ID: gwang39

E-mail: gwang39@jhu.edu

Phone: +16479888206

Date: Dec. 11th, 2017

(a) [50 points] Give an efficient algorithm that takes strings *s*, *x*, and *y* and decides if *s* is an interleaving of *x* and *y*. Derive the computational complexity of your algorithm.

To start with, we can simplify the question by repeating x and y to x' and y' which has the exact same length of string s.

As a result, our question becomes if s is an interleaving of x' and y'.

By doing so, we no longer need to wrap around and consider multiple periods of x' and y' because each of them is already has the same number of characters as s

After that, we can denote jth character of s and s[j] and let s[1:j] be the first j characters of s. We define the analogous notation for x' and y'. We know that if s is an interleaving of x' and y', then its last character comes from either x' or y'. If we remove this character, we then get a smaller recursive problem on s[1:n-1] and prefixes x' and y'

Consequently, this can be considered as sub problems defined by prefixes x' and y'. Let M[i,j] = YES if s[1:i+j] is an interleaving of x'[1:i] and y'[1:j]. If there is such as interleaving, then the final character is either x'[u] or y'[j]

The basic recurrence can be shown as

(M[i,j] = YES) <=>[(M[i-1,j) = YES)AND(s[i+j]=x'[i])]OR[(M[i,j-1) = YES)AND(s[i+j]=y'[j])]

M[i,j] can be generated in a recursive algorithm as below:

```
M[0,0] = YES
for k = 1 to n do
        for all pairs (i,j) such that i+j=k do
                if M[i-1,j] = YES and s[i+j] = =x'[i] then
                        M[i,j] = YES
                else if M[i,j-1] = true and s[i+j] = =y'[i] then
                        M[i,j] = YES
                else
                        M[i,j] = NO

        end do
end do
return YES if and only if there is some pair (i,j) with i+j = n where M[i,j] = YES
```

There are $O(n^2)$ values M[i,j] to build up, and each takes constant time to fill in from the result on previous sub problem. Therefore, the total running time is $O(n^2)$

(b) [50 points] Implement your algorithm above and test its run time to verify your analysis. Remember that CPU time is not a valid measure for testing run time. You must use something such as the number of comparisons.

- Implement the above data structure: please run the main program and check the console output

- Runtime can be validated based on number of checks displayed at the end of the run

- Below table includes the number of checks for different length of the string

s	x	y	run time (check count)
100010101	101	00	100
10010	10	0	36
1010101010	1010	10	121

- Based on table above, we can see the running time of this algorithm can be shown as $O(n^2)$ where n is the number of characters string in s