COMP S265F
Assignment 1
15.4.2021

Name : Jiawei Wang
ID : s1239587

77

Question 1 (15 marks)

(a) $(265, 380) \to (265, 115) \to (150, 115) \to (35, 115)$
$\to (35, 80) \to (35, 45) \to (35, 10)$
$\to (25, 10) \to (15, 10) \to (5, 10)$
$\to (5, 5) \Rightarrow$ g.c.d $= 5$ ✓

(b) $(265, 380) \to (265, 115) \to (115, 35) \to (35, 10)$
$\to (10, 5) \Rightarrow$ g.c.d $= 5$ ✓

15

Question 2 (35 marks)

(a) line 3 takes average $\frac{0+1+2+\cdots+n-1}{n} = \lfloor \frac{n-1}{2} \rfloor$ additions. — $O(n)$ time

line 4 takes $O(1)$ time to store the sum to $Y[i][j]$ — $O(1)$ time

So the for-loop in line 1 and line 2 takes $\underbrace{(0+1+2+\cdots+n-1)}_{\text{line 1 and line 2}} \cdot \underbrace{O(n+1)}_{\text{line 3 and line 4}}$

$= \frac{n(n-1)}{2} \cdot O(n+1) = O(n^3)$ time.

Therefore, the time complexity of function is $O(n^3)$ ✓ 10

(b) The bottleneck is that there are a lot of repeated calculations.
For example: the value of adding all the entries $X[\lfloor \frac{n}{2} \rfloor]$ to $X[n-1]$.
We do this addition for about $\lfloor \frac{n}{2} \rfloor$ times, we can reuse the sum we already put into Y. ✓ 5

(c) Please check q2.py . function func1()

10

Scanned with CamScanner

Question 2 (cont'd)

(d) the pseudo-code of q2.py func1():

```
1. for i <- 0 to n-1 do
2.     for j <- i to n-1 do
3.         if j is 0
4.             then Y[i][j] = X[0]
5.         end if
6.         else Y[i][j] = Y[i][j-1] + X[j]
7.     end for
8. end for
```

from line 3 to line 6: takes $O(1)$ time

So the for-loop in line 1 and lin 2: takes $(0+1+2+\dots+n-1)\cdot O(1)$

$= \frac{n(n-1)}{2} \cdot O(1) = O(n^2)$ time

Therefore, the time complexity of func1() is $O(n^2)$.

10

Question 3 (25 marks)

(a) local minimum.

left complete binary tree: 0

right complete binary tree: 1

5

(b) Please check q3.py. function: traverse1()

0

- run time error for some case.
- not handling right side.

(c) In function traverse1(). I used a property of the local minimum number:

— We must have at least one local minimum number in every possible route from the top of the tree to bottom.

So we just need to search one route to find one of the local minimum number. I traverse from top to bottom, and the local minimum is on the leaf. I called traverse1(node.left) recursively till I hit the leaf.

Then we have:

In the worst case:

$$T(n) = T\left(\frac{n}{2}\right) + O(1)$$

10

Since $n^{\log_b a} = n^{\log_2 1} = 1$, we can apply case 2 of the master theroem.

We can conclude that the time complexity of our function is $O(n^0 \log n) = O(\log n)$.

Question 4 (25 marks)

(a) please check q4.py.

10

(b) The algorithm in (a) is a greedy algorithm.

— Every step we try to find the largest value of coin to minimize the total amount of coins.

Because of that, we can make sure our final output is the minimal list of coins in descending order.

In each amount of coins, first we caculate the number of it by using "//" then we substract the total value of this amount coins to make sure the final coin value equals $value.

When "value // 1" is zero, which means the final coin value equals $value.

2