Conder Shou

COMS W1004 Problem Set 1

Chapter 1: 10a – *Use values 22 and 30*

1. Get two positive integers as input: call the larger value "I" and the smaller value "J"

   I = 30, J = 22, R = N/A

2. Divide I by J, and call the remainder R

   I /J = 30/22 = 1 with a remainder of 8

   I = 30, J = 22, R = 8

3. If R is not 0, then reset I to the value of J, reset J to the value of R, and go back to Step 2

   R is not 0, it is 8, therefore I is set to the value of J – which is 22, and J is set to the value of R – which is 8.

   I = 22, J = 8, R = 8

Going back to Step 2.

   I/J = 22/8 = 2 with a remainder of 6

   R = 6

Reset I to value of J, and J to value of R and repeat.

   I/J = 8/6 = 1 with a remainder of 2. R = 2

Repeat.

   I/J = 6/2 = 3.

I = 6, J = 2, R = 3.

4. Print out the answer, which is the value of J.

   I = 6, J = 2, R = 3. The answer is 2.

The final output of this algorithm with the given values of 22 and 30, is 2.

Chapter 1: 10b – *Use values 12 and 0*

Does this algorithm work correctly when the two inputs are 12 and 0?

It does not work out correctly because 12 cannot be divided by 0. The quotient is undetermined. This occurs during step 2, when the algorithm says to divide I by J, which are in this case, 12 and 0, respectively. When this is attempted, no mathematical solution can be found, which therefore means no actual remainder can be found either.

In order for this algorithm to give the appropriate error message, the following modification has been made to Step 2 (shown in underline).

2 If J is equal to 0, then print out the words "No common divisor. An integer cannot be divided by 0. The quotient is undetermined."

If J is greater than 0, then divide I by J, and call the remainder R.

This modification to the algorithm should allow it to give the proper error message whenever it encounters 0 as one of the chosen integers.

Chapter 1: 11 - *Instead of 25 cities use 20*

By starting out in the first city, this means we calculate the possibilities for the paths to the remaining 19 cities. There are 19! possible paths. If the computer can analyze 10,000,000 paths per second, then it should take the computer 19!/10000000 = 12164510041 seconds to determine the optimal route for visiting these 20 cities (which is approximately 140,793 days).

On the basis of this answer, this is not a feasible algorithm. Instead, I think we could use an algorithm that might not find THE MOST optimal solution to the problem, but that can estimate a fairly good solution within a much more efficient span of time.

One way of doing this might be to start at a city, and then to select the next unvisited city closest to the currently occupied city. This might give us a reasonable solution that is fairly good, but most likely will also not be the most absolutely optimal way of traveling between the 20 cities. However, this algorithm will certainly be much faster to execute.

Chapter 2: 18

a. The output of the algorithm in Figure 2.16 as it currently stands is:

There is a match at position 10
There is a match at position 23
There is a match at position 28

b. We can modify the algorithm so that it finds only the complete word "and", not the character sequence contained within a word like "band", by doing the following modification to the second line:

…
Get values for the text T_1, T_2, etc… T_n
Set the value for P_1 to be the value of a space.
Get values for the pattern starting at position P_2, then on to P_3, P_4, etc… P_{m+1}
Set the value for P_{m+2} to be the value of a space.

Set k…. (the rest of the code remains the same)

This way, it will only find the complete word "and", and consider it a mismatch if contained within any other word, like "band", "Andrew", etc…

However, because we need to also accept the word "and" at the end of a sentence or phrase, where there are punctuations. As a result, we should create another loop that envelops the first two loops and contains the code

Set the value for P_1 to be the value of a [insert punctuation character].
Set the value for P_1 to be the value of a [insert punctuation character].
Set k to 1.

and let it continue looping until it has exhausted all potential punctuation characters that would still allow "and" to be a complete word on its own. The loop should be limited by the number of punctuation characters.

Chapter 2: 25

Get value for n, the size of the list.

Get values for list V_1, V_2… V_{n-1}, -1

Set k, the starting location for the checking process, to 1.

Set value of Mismatch to YES.

While both (k <= n-1) and (Mismatch = YES) do

if V_k does not equal V_{k+1} then

Increment k by 1

else

Mismatch = NO

End of loop

If Mismatch = NO then

Print the message "Yes"

else

Print the message "No"

End of program

NOTICE: I put in the condition that "k" needed to remain less than the value of "n", the size of the list, MINUS one. This is because there is a special value V = -1 at the end of the list. Because that is merely a marker for the ending of the list, it should not be taken into account in the actual testing of adjacent pairs.

Chapter 3: 31

a. |1.2| = 1

|2.3| = 2

|8.9| = 8

|-4.6| = -5

b.

n	|lg n|
2	1
3	1
4	2
5	2
6	2
7	2
8	3

c.

n	Number of Compares, Worst Case
2	2
3	2
4	3
5	3
6	3
7	3
8	4

See last attached page for drawn tree structures.

d. Number of comparison binary search requires in worst case on n-element list = |lg n| + 1

See last attached page for the testing of this formula with other values of n.

Chapter 3: 36

a. The first graph shown has an Euler path, but not the one underneath.
   Just picking a point such as B, we can then go to A, C, along the diagonal back to B, then to D and lastly C.

b. The graphs that have Euler paths are (i) and (iii).
   Euler path for Graph (i): A-C-B-D-A-E-B-F-A
   Euler path for Graph (iii): D-C-A-F-B-E-A-D-B-C

c. The order of magnitude of this algorithm is 2. (The Big-Oh notation is $O(n^2)$).

d. No, the Euler path problem is not intractable because it has polynomial-time algorithms.