Christine Chong

05 23 17

Homework One - Problem Set

Chapter 1 - Schneider & Gersting

10a) Input: 2∅ & 32

Step One

I=32 J=2∅

Step Two
32/2∅ = 1 r 12
R = 12
Step Three
I=2∅ J=12

→ Step Two
2∅/12 = 1 r 8
R=8
Step Three
I=12 J=8

→ Step Two
12/8 = 1 r 4
R = 4
Step Three
I=8 J=4

→ Step Two
8/4 = 2 r ∅
Step Four
4

Output: 4

10b) Input: ∅,32

Step One
I=32 J=∅
Step Two
32/∅ = undefined

→ Because there is no remainder defined, the algorithm will not work as intended. There is no numerical value, so it will not be able to continue to the next step, as it checks for and requires a numerical value for R.

Fixed Algorithm:

Step One:

Get two positive integers as the input. If one of these values is ∅, print an error. ("∅ is not a valid input"). Else call the larger value I and smaller value J.

Step Two - Five would be the same.

11) The formula for all possible paths is $\frac{n!}{2}$ (all the paths) (eliminating repeating distances)

For 25 cities: $\frac{25!}{2}$ = 7,755,605,021,665,492,992,000,000
÷ 10,000,000 computations / sec
775,560,502,166,549,299 seconds
÷ 60
12,926,008,369,442,488 minutes
÷ 60
215,433,472,824,041 hours
÷ 24
8,976,394,701,001 days
÷ 365
24,592,862,194 years → Feasible

Due to the amount of possible combinations and lack of restrictions, this problem is not easily solved. A more reasonable approach would be to take the shortest path from the initial point to an unvisited point. It does not guarantee the most optimal path, as there can be overlaps.

Chapter 2 - Schneider & Gersting

19) Because k is the value of the location for the attempted matches, it will never move past the initial location. Also, the while loop could be infinite if the value of $(n \leq m+1)$ is either 1 or greater than 1 because there will never be a case where k is greater than $(n \leq m+1)$ because k is stuck at the value of 1.

20) Algorithm using trial division (up to $\sqrt{n}$). (If both factors are greater than $\sqrt{n}$, then they would multiply out to be a number greater than n.)

```
function isprime(n) {
    if n!== 0 && n!== 1 {
        for (i=2 ; i< √n ; i++){
            if (n % i == 0) {
                return false + "The smallest factor is" + i }}
                return true + "The number is prime"}
```

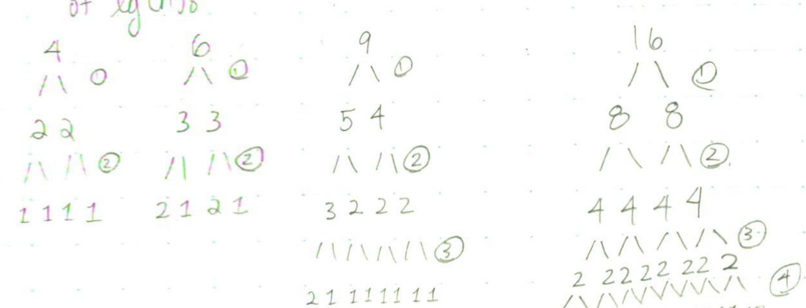
23)

```
function findsum(list, sum){
    let max= list.length
    let start = 0
    while (start< max) {
        if (i=1 & i< max){
            c = list[start] + list[i]
            if ( c == sum) {
                return list[start] + "," + list[i] }}
                start ++ }}
    if (start == max) return "There are no pairs that add to" + sum
```

We can add the first item of the list with another item on the list. Then we can compare sums. If there is no sum we can continue by iterating over the list.

Chapter 3 - Schneider & Gersting

17a) The levels at which the tree merges can occur would be at the floor of lg(n).



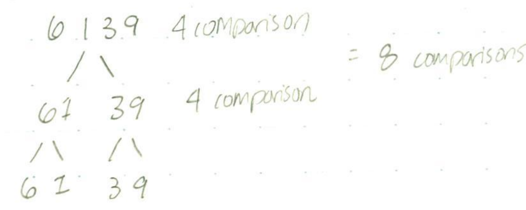
$lg(4)=2$ $lg(6)=2$ $lg(9)=3$ $lg(16)=4$

17b) The number of comparisons should be at $n \times \text{floor}(lgn)$.
$\underset{\text{\# of items}}{n}$, $\underset{\text{\# of levels}}{\text{floor}(lgn)}$

17c) The order of magnitude of the mergesort should be $O(n\ lgn)$ because this is the number of comparisons that need to be made.

17d)

16)



$(4 * lg 4) = 8$ comparisons

Yes it does. Because there needs to be a n number of comparisons done per level (lgn).

31a) $\lfloor 1.2 \rfloor = 1$, $\lfloor 2.3 \rfloor = 2$, $\lfloor 8.9 \rfloor = 8$, $\lfloor -4.6 \rfloor = -5$

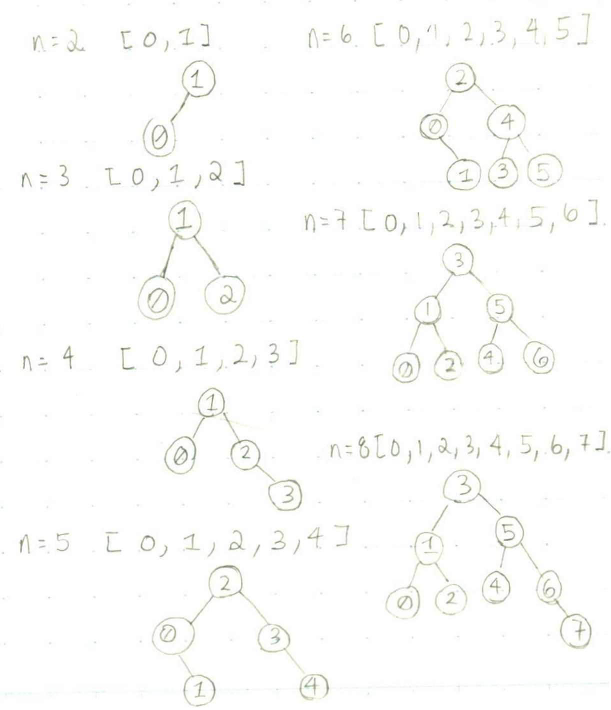
31b)

n	$\lfloor lg n \rfloor$
2	1
3	1
4	2
5	2
6	2
7	2
8	3

31c)

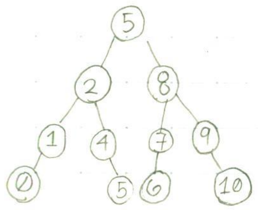
n	Number of compares (Worst case)
2	2
3	2
4	3
5	3
6	3
7	3
8	4

Binary Search Trees



31d) Number of comparisons on worst case = floor(lgn) +1

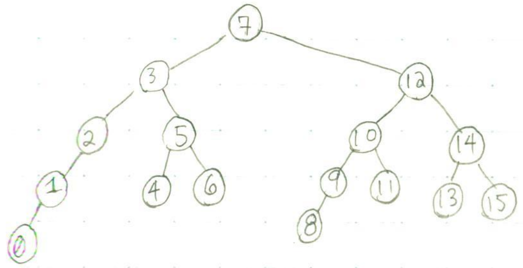
n=11 [0,1,2,3,4,5,6,7,8,9,10]



4 comparisons

floor(lg11) +1
3+1 = 4

n=16 [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]



5 comparisons

floor(lg16) + 1
4 + 1 = 5

36a) There is a Euler path for the first graph.

① C-A ② A-B ③ B-C ④ C-D ⑤ D-B

36b)

There is a Euler path for the first graph (0 odd nodes)

① A-C ② C-B ③ B-D ④ D-A ⑤ A-E ⑥ E-B ⑦ B-F ⑧ F-A

There is no Euler path for the second graph (4 odd nodes)

There is a Euler path for the third graph (2 odd nodes)

① C-A ② A-F ③ F-B ④ B-E ⑤ E-A ⑥ A-D ⑦ D-C ⑧ C-B
⑨ B-D

			# of times it runs
36c)	Step 4	While (i ≤ n) do Steps 5 through 13	n times
	Step 5	Set value of node counter j to 1	
	Step 6	set value of Degree to 0	-> j starts @ 1
	Step 7	while (j ≤ n) do steps 8 - 10	n - 1 times
	Step 8	if an edge i-j exists then	+1
	Step 9	Increase Degree by 1	
	Step 10	Increase j by 1	
	Step 11	If Degree is an odd number then	+1
	Step 12	Increase Odds by 1	
	Step 13	Increase i by 1	
	Step 14	If Odds > 2 then	+1
	Step 15	Print "No Euler Path"	
	Step 16	Else	
	Step 17	Print " Euler path exists	

number of steps

$n \cdot n-1 + 1 + 1 + 1$

$\downarrow$

$n^2 - 1n + 3$

$\downarrow$

$O(n^2)$

36d) This problem is not intractable because n^2 is a polynomial function. The function is also deterministic. It will output consistently.