

~~P2~~

(Pms 2 u)

tetraMultiple(n)

if (n == 0 V n == 1 V n == 2)

- return 0

if (n == 3)

return 1

~~else~~

return (tetraMultiple(n-1) + tetraMultiple(n-2)
+ tetraMultiple(n-3) + tetraMultiple(n-4))

43 2 1 6 7 8 5

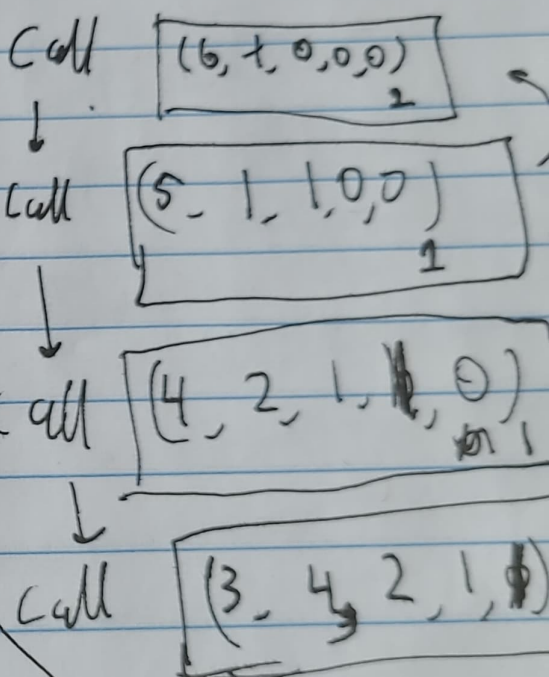
L: 5
F: 54

F(3,2,1): 12354
L(6,7,8) 12354678

remove

Q.enqueue(D.remove)
Q.enqueue(D.remove)
D.enqueue(
D.addFirst(Q.dequeue())
D.offerFirst(Q.dequeue())
D.offerFirst(Q.dequeue())

n = 7



Part 2 a)

testRecursive(n, a, b, c, d)

```

if (n == 0)
  return a
if (n == 1)
  return b
if (n == 2)
  return c
if (n == 3)
  return a + b + c + d

```

return (n-1, a+b+c+d, a, b, c)

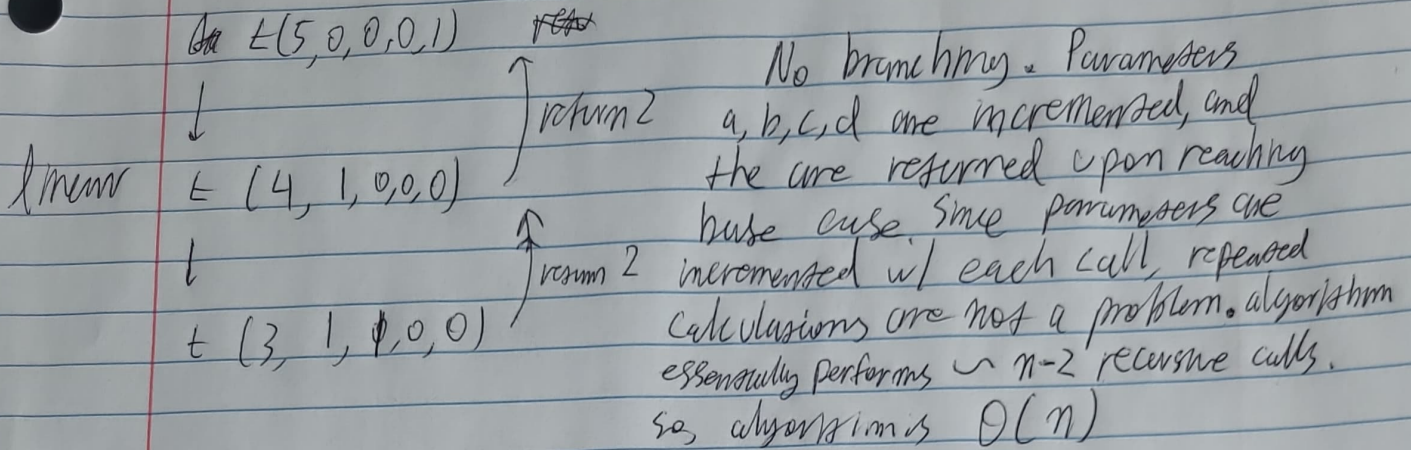
test(n) {

return testRecursive(n, 0, 0, 0, 1)

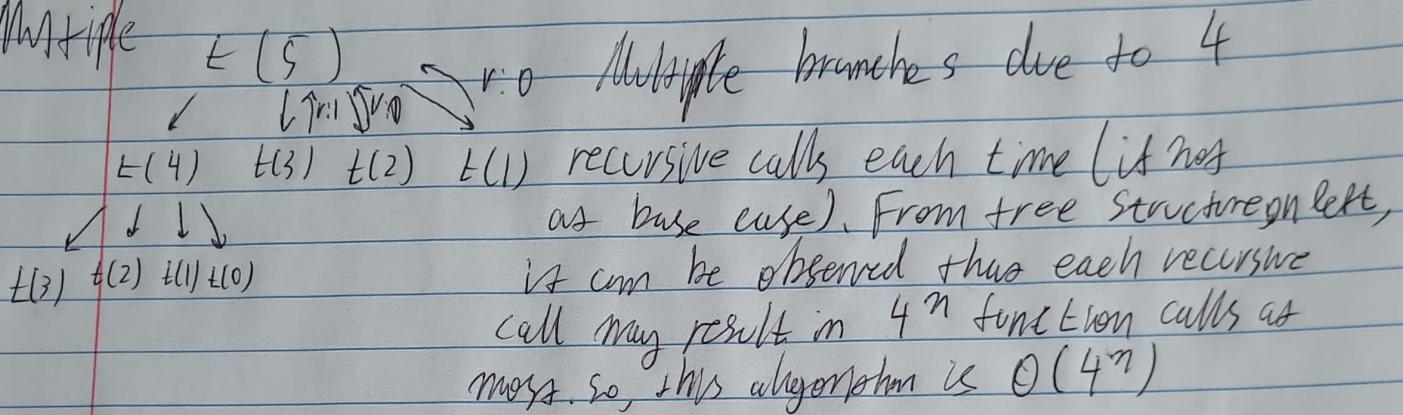
3

Part 2

(c)





Multiple



Potential Bottleneck: doing subproblem again. e.g. $T(n)$ will call $T(n-1)$, $T(n-2)$, $T(n-3)$, $T(n-4)$ and $T(n-1)$ will call $T(n-2)$, $T(n-3)$, $T(n-4)$, $T(n-5)$ and $T(n-2)$ will call $T(n-3)$, $T(n-4)$, $T(n-5)$, $T(n-6)$. As can be seen, same problem ^{are} repeated in each recursive call.

(d) Linear Recursive function uses tail recursion, as tail recursion is defined as a single recursive call at end of function, which matches structure of `Linear Recursive`.

3.3. n vs. Execution Time for Linear and Multiple Recursive Tetranacci Calculator

n	 t_1	 t_2	
0	600	1100	
5	300	700	
10	400	5700	
15	800	124400	
20	800	316900	
25	1400	5901300	
30	3300	173755200	

$$t_1 \sim mn + b$$

REGRESSION PARAMETERS

$$m = 76.42857$$
$$b = -60.71429$$

STATISTICS

$$R^2 = 0.6302$$
$$r = 0.7938$$

RESIDUALS

 e_1

plot

$$t_2 \sim a^* b^{(n)}$$

☐ Log Mode ?

REGRESSION PARAMETERS ?

$$a = 0.268942$$
$$b = 1.96642$$

STATISTICS

$$R^2 = 1$$

RESIDUALS

 e_2

plot

