Q1.

Let $P(n)$ denote $f(n-1)f(m-1)+f(n+1)f(m+1)+f(n)f(m)>f(n+m)$. Show $P(n)$ holds $\forall n, m \geq 1 \in \mathbb{N}$.

As $P(n)$ is symmetric with respect to n and m, WLOG, we prove it by strong induction on n.

Basis step.

$P(1)$ holds since

$$f(0)f(m-1)+f(2)f(m+1)+f(1)f(m)>f(1+m)$$

$$f(m+1)+f(m)>f(1+m)$$

$f(m)>0$ # obvious, by definition of f when $m \geq 1$

$P(2)$ holds since

$$f(1)f(m-1)+f(3)f(m+1)+f(2)f(m)>f(2+m)$$

$$f(m-1)+2f(m+1)+f(m)>f(2+m)$$

$$f(m-1)+2f(m+1)+f(m)>f(1+m)+f(m)$$

$f(m-1)+f(m+1)>0$ # obvious, by definition of f when $m \geq 1$

Inductive step. Assume $P(i)$ holds for $1 \leq i \leq k$, for an arbitrary fixed k where $2 \leq k \in \mathbb{N}$, *i.e.*,

$f(i-1)f(m-1)+f(i+1)f(m+1)+f(i)f(m)>f(k+m)$ IH

We must show $P(k+1)$ holds too, *i.e.*, $f(k)f(m-1)+f(k+2)f(m+1)+f(k+1)f(m) > f(k+1+m)$.

Since $2 \leq k$, $1 \leq k\text{-}1$, by IH, $P(k)$ and $P(k-1)$ hold: *i.e.*,

$f(k-1)f(m-1)+f(k+1)f(m+1)+f(k)f(m)>f(k+m)$ $> f(k+m)$

$f(k-2)f(m-1)+f(k)f(m+1)+f(k-1)f(m)$ $> f(k+m-1)$

adding both sides:

$$(f(k-1)+f(k-2))f(m-1)+(f(k+1)+f(k))f(m+1)+(f(k)+f(k-1))f(m) > f(k+m) + f(k+m-1)$$

$f(k)f(m-1)+f(k+2)f(m+1)+f(k+1)f(m) > f(k+m+1)$ # by definition of f

This completes the inductive step. Hence, by strong induction,

$f(n-1)f(m-1)+f(n+1)f(m+1)+f(n)f(m)>f(n+m)$ $\forall n, m \geq 1 \in \mathbb{N}$ □

Q2.

a)

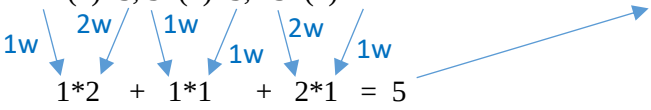
In Tutorial 3, we proved that any non-empty FBT has an odd number of nodes. We would like to see how we can make all distinct FBTs with n nodes, using previously made FBTs (the ones that have less than n nodes) in a ***systematic*** approach. Structurally, it is important what left subtree an FBT has and what at its right if the left and right have different structures. Considering this fact, after some scratch work, we notice the problem boils down in how many different ways we can make an odd number using two smaller odd numbers plus one (for the root) if order matters.

1=(1) (just the root) — 1 way

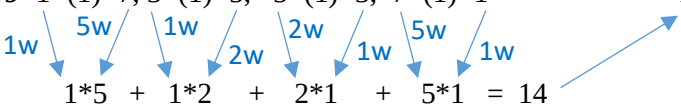
3=1+(1)+1 — 1 way

5=1+(1)+3, 3+(1)+1 — 2 ways — # note the order is important

7=1+(1)+5, 3+(1)+3, 5+(1)+1 — 5 ways



9=1+(1)+7, 3+(1)+5, 5+(1)+3, 7+(1)+1 — 14 ways



Let $T(n)$ denote number of FBTs with n nodes where $n = 2m + 1, m \geq 2 \in \mathbb{N}$.

As an example, the pattern for $T(9)$ looks like $T(9) = T(1) * T(7) + T(3) * T(5) + T(5) * T(3) + T(1) * T(5)$

In general, the pattern looks as follows:

$$T(2m+1) = T(1) * T(2m-1) + T(3) * T(2m-3) + \cdots + T(3) * T(2m-3) + T(1) * T(2m-1)$$

or, $T(n) = T(1) * T(n-2) + T(3) * T(n-4) + \cdots + T(3) * T(n-4) + T(1) * T(n-2)$

We use summation for a shorter notation:

$$T(n) = \begin{cases} 1 & where\ n = 1\ or\ n = 3 \\ \sum_{j=1}^{\frac{n-1}{2}} T(n-2j).T(2j-1) & where\ n = 2m+1, m \geq 2 \in \mathbb{N} \end{cases}$$

b)

We want to prove $T(n) \geq 2^{\frac{n-1}{2}}$.

For proof readability, we can change the variable. Let $T'(m)$ denote number of FBTs with $2m + 1$ nodes where $m \in \mathbb{N}$. Hence, $T'(m) = T(n)$ where $m = \frac{n-1}{2}$. Now, we need to prove $T'(m) \geq 2^m$ where

$$T'(m) = \begin{cases} 1 & m < 2 \in \mathbb{N} \\ \sum_{i=1}^{m} T'(m-i).T'(i-1) & 2 \leq m \in \mathbb{N} \end{cases}$$

You may want to see the scratch work, here.

Proof by simple induction.

Let $P(m)$ denote $T'(m) \geq 2^m$. We prove it $\forall\, m \geq 5 \in \mathbb{N}$.

Basis step. $P(5)$ holds since $T'(5) = 42 \geq 2^5 = 32$.

Inductive step. Assume $P(k)$ holds for an arbitrary k where $k \geq 5 \in \mathbb{N}$. *i.e.*,

$T'(k) = T'(k-1).T'(0) + T'(k-2).T'(1) + \cdots + T'(1).T'(k-2) + T'(0).T'(k-1) \geq 2^k$. IH

We must show $P(k+1)$ holds too, *i.e.*, $T'(k+1) \geq 2^{k+1}$.

$T'(k+1) = T'(k).T'(0) + T'(k-1).T'(1) + \cdots + T'(1).T'(k-1) + T'(0).T'(k)$

$= 2T'(k) + T'(k-1).T'(1) + \cdots + T'(1).T'(k-1)$

$\geq 2T'(k)$ # $T'(k-1).T'(1) + \cdots + T'(1).T'(k-1) \geq 0$ (since number of FBTs with more than 1 node is positive, proved in Tutorial 3)

$\geq 2.\, 2^k$ # by IH

$\geq 2^{k+1}$. □

Q3.

a)

We can model the program using different variables. You may want to see the scratch work, here. In the following model, n demonstrates steps rather than hours, and every step is 2 hours.

$$T(n) = \begin{cases} 0 & n = 0 \\ 4 & n = 1 \\ 4T(n-1) - 3T(n-2) & n > 1 \in \mathbb{N} \end{cases}$$

b)

Proof by simple induction.

Let $P(n)$ denote $T(n) < T(n+1)$.

Basis step. $P(0)$ holds since $T(0) = 0 < T(1) = 4$.

Inductive step. Assume $P(k)$ holds for an arbitrary k where $0 \leq k \in \mathbb{N}$, *i.e.*, $T(k) < T(k+1)$.

We must show $P(k+1)$ holds too; *i.e.*, $T(k+1) < T(k+2)$.

$T(k+1) > T(k)$ # by IH

$3T(k+1) > 3T(k)$

$4T(k+1) > 3T(k) + T(k+1)$

$4T(k+1) - 3T(k) > T(k+1)$

$T(k+2) > T(k+1)$

□

c)

$T(n) = \mathbf{4\,T(n-1) - 3\,T(n-2)}$

$= 4\big(4\,T(n-2) - 3\,T(n-3)\big) - 3\,T(n-2) = \mathbf{13\,T(n-2) - 12\,T(n-3)}$

$= 13\big(4\,T(n-3) - 3\,T(n-4)\big) - 12\,T(n-3) = \mathbf{40\,T(n-3) - 39\,T(n-4)}$

$= 40\big(4\,T(n-4) - 3\,T(n-5)\big) - 39T(n-4) = \mathbf{121\,T(n-4) - 120\,T(n-5)}$

$= 121\big(4\,T(n-5) - 3\,T(n-6)\big) - 120\,T(n-5) = \mathbf{364\,T(n-5) - 363\,T(n-6)}$

…

since $4 = \frac{1}{2}(3^2 - 1)$ $13 = \frac{1}{2}(3^3 - 1)$ $40 = \frac{1}{2}(3^4 - 1)$ $121 = \frac{1}{2}(3^5 - 1)$ ...

$T(n) = \frac{1}{2}(3^n - 1)\,T(1) - \left(\frac{1}{2}(3^n - 1) - 1\right)T(0)$

Hence, $T(n) = \frac{1}{2}(3^n - 1)\,4 = \mathbf{2(3^n - 1)}$

d)

Let $P(n)$ denote $T_r(n) = T_c(n)$ where

$$T_r(n) = \begin{cases} 0 & n = 0 \\ 4 & n = 1 \\ 4T(n-1) - 3T(n-2) & n > 1 \in \mathbb{N} \end{cases}$$

$T_c(n) = 2(3^n - 1)$

Proof by strong induction.

Basis step:

$P(0)$ holds as $T_r(0) = 0 = T_c(0) = 2(3^0 - 1) = 0$

$P(1)$ holds as $T_r(1) = 4 = T_c(1) = 2(3^1 - 1) = 4$

Inductive step:

Assume $P(i)$ holds for $0 \le i \le k$, for an arbitrary $k > 1 \in \mathbb{N}$,

i.e., $T_r(i) = T_c(i)$, *i.e.*, $4T(i-1) - 3T(i-2) = 2(3^i - 1)$

We must show $P(k+1)$ holds too, *i.e.*, $T_r(k+1) = T_c(k+1)$

$T_r(k+1) = 4T(k) - 3T(k-1)$

$= 4.2\big(3^k - 1\big) - 3.2\big(3^{k-1} - 1\big)$ by IH, and since $0 \le k-1 \le k \quad \forall k > 1$

$= 4.2.3^k - 8 - 2.3^k - 6$

$= 6.3^k - 2$

$= 2(3^{k+1} - 1)$

This completes the inductive step. Hence, $T_r(n) = T_c(n) \ \ \forall n \in \mathbb{N}$. □

Q4.

a)

After writing a brute-force algorithm to generate all triples (i, j, k) that satisfy $4i + 6j + 10k = n$, it's clear that there are lots of opportunities to undercount (miss stamp configurations) and overcount (miss the same configuration being produced in two different ways). Here's a way to build up the algorithm step-by-step with confidence.

step 1: Let $f(n)$ represent the number of con gurations of stamps using only four-cent stamps. Since n is either divisible by 4 or not, this function is straightforward:

```
def f(n):
    if (n >= 0) and (n%4 == 0):
        return 1
    else:
        return 0
```

step two: Expand the scope to find stamp configurations allowing both six- and four-cent stamps for postage of n (no ten-cent stamps allowed). These break into two completely disjoint (non-overlapping) sets: configurations that include at least one six-cent stamp, and those that include only 4-cent stamps. The number of configurations that include at least one six-cent stamp is the same as the number of six- and four-cent configurations for postage $n - 6$, when we peel off one six-cent stamp:

```
def sf(n):
    if n >= 0:
        return f(n) + sf(n-6)
    else:
        return 0
```

step three: Expand the scope again, to find the number of stamp configurations allowing four-, six-, and ten-cent stamps for postage of n. These, again, break into two completely disjoint (non-overlapping) sets: configurations that include at least one ten-cent stamp, and those that include only six- and four-cent stamps. The number of configurations that include at least one ten-cent stamp is the same as the number of ten-, six-, and four-cent configurations for $n - 10$, when we peel off one ten-cent stamp:

```
def tsf(n):
    if n >= 0:
        return sf(n) + tsf(n-10)
    else:
        return 0
```

symmetry: As a lead-up to proving monotonicity, notice that we can switch the roles of six and four in the definition of sf(n) above. In other words, define $s(n)$ to be the number of ways to make postage n with only six-cent stamps, and then define $fs(n)$ to be the number of ways to make postage of n with four- and six- (but no ten-) cent stamps:

```
def s(n):
    if (n >= 0) and (n%6 == 0):
        return 1
    else:
        return 0
```

```
def fs(n):
    if n >= 0:
        return s(n) + fs(n-4)
    else:
        return 0
```

Now $fs(n) = sf(n)$: the number of ways to make postage n using only four- and six-cent stamps.

b)

monotonicity: Define $P(n) : tsf(n-2) \leq tsf(n)$. Claim: for all even numbers n greater than 2, $P(n)$

base cases: It is clear (from a printout of that brute-force algorithm) that $P(n)$ holds for $n = 4,6,\ldots,22$.

inductive step: Assume n is an even number 24 or greater and that $P(i)$ is true for all even numbers $4 \leq i < n$.

must prove $P(n)$: $tsf(n-2) \leq tsf(n)$:

$tsf(n) - tsf(n-2) = sf(n) + tsf(n-10) - (sf(n-2) + tsf(n-12))$

$= sf(n) + sf(n-10) + tsf(n-20) - (sf(n-2) + sf(n-12) + tsf(n-22))$

$\geq sf(n) + sf(n-10) - (sf(n-2) + sf(n-12))$

by IH, since $4 \leq n-20 < n$

$= f(n) + sf(n-6) + f(n-10) + sf(n-16) - (sf(n-2) + sf(n-12))$

by definition of $fs(n)$

$= f(n) + sf(n-6) + f(n-10) + sf(n-16)$

$-(s(n-2) + fs(n-6) + s(n-12) + fs(n-16))$

by definition of $fs(n) = sf(n)$

$\geq f(n) + f(n-10) - s(n-2) - s(n-12)$ # $fs(n) = sf(n)$

$= f(n) + f(n-2) - s(n-2) - s(n)$ # $-10 \equiv -2 \ mod\ 4, -12 \equiv 0 \ mod\ 6$

≥ 0

either $f(n) = 1$ or $f(n-2) = 1, n \in \{0,2\} \ mod\ 4$

not both $s(n) = 1$ and $s(n-2) = 1,\ n \in \{0,2,4\} \ mod\ 6$

Q5.

a)

```
def max_sum1(seq, frm, to)
    Returns the sum and indices of a sub-sequence in seq that has maximum sum
    brute force approach
    return: (max_sum, from_index, to_index)
    max_sum = seq[frm]
    for i in range(frm, to)
        for j in range(i+1,to+1)
            new_sum = sum(seq[i:j+1])
            if new_sum > max_sum:
                max_sum, m, n = new_sum, i, j
    return max_sum, m, n
```

b)

$n = to - frm + 1$

The body of the inner loop will execute $n + n - 1 + n - 2 + n - 3 + \cdots + 1$ times, which is $\frac{n(n+1)}{2}$ times. In each iteration, function sum is called that is in $O(n)$, i.e., another nested loops that iterates n times . Hence, `max_sum1` is in $O(n^3)$. Obviously, a brute-force algorithm in $O(n^2)$ can be developed too.

c)

```
max_sum2(seq, frm, to)
    Returns the sum and indices of a sub-sequence in seq that has maximum sum
    by divide and conquer approach
    return: (max_sum, from_index, to_index)
    if frm < to
         mid = (frm+to) div 2
         half1 = max_sum2(seq,frm, mid)
         half2 = max_sum2(seq,mid+1, to)

         frmLeft = i = mid
         combinedSum = leftSum = seq[i]
         while i > half1[frm_index]
             i -= 1
             leftSum += seq[i]
             if combinedSum < leftSum:
                 combinedSum, frmLeft = leftSum, i

         toRight = i = mid+1
         sumSoFar = combinedSum += seq[i]
         while i < half2[to_index]
             i += 1
             sumSoFar += seq[i]
             if combinedSum < sumSoFar:
                 combinedSum, toRight = sumSoFar, i

         return max(half1, half2, combinedSum)
     else: return seq[frm],frm, frm
```

d)

$n = to - frm + 1$

The base case needs constant time; the recursive case needs $T\left(\frac{n}{2}\right) time$ to find the max_sum in each half plus the time needed for calculating a max_sum that is involves both halves and some constant time for overhead.

$$T(n) = \begin{cases} c_1 & n = 1 \\ c_2 + 2T\left(\frac{n}{2}\right) + n & n > 1 \end{cases}$$

e)

The time complexity of `max_sum1` and `max_sum2` is $O(n^3)$ and $(n \log n)$, respectively. For the latter, use Master Theorem.

Appendix.

Q2b scratch work

m	n	$T(n)$	$2^{(n-1)/2}$	$P(n)$
0	1	1	1	False
1	3	1	2	False
2	5	2	4	False
3	7	5	8	False
4	9	14	16	False
5	11	42	32	True
6	13	132	64	True
7	15	429	128	True
8	17	1430	256	True
…				

Q3a scratch work

We can model the problem in different ways:

n	$T(n)$
0	2
2	4
4	16
6	=16*4-3*4=52
8	=4*52-3*16=160
…	

n	$T(n)$
2	4
4	16
6	=16*4-3*4=52
8	=4*52-3*16=160
…	

n	$T(n)$
0	0
1	4
2	16
3	=16*4-3*4=52
4	=4*52-3*16=160
…	

n	$T(n)$
1	4
2	16
3	=16*4-3*4=52
4	=4*52-3*16=160
…	