First Name: Zhihong
Last Name: Wang
Student ID: 1002095207

First Name: Fenglun
Last Name: Wu
Student ID: 1002596684

We declare that this assignment is solely our own work, and is in accordance with the University of Toronto Code of Behaviour on Academic Matters.

This submission has been prepared using LaTeX.

1. Consider the Fibonacci function f:

$$f(n) = \begin{cases} 0, & \text{if } n = 0 \\ 1, & \text{if } n = 1 \\ f(n-2) + f(n-1) & \text{if } n > 1 \end{cases}$$

Prove $f(n-1) * f(m-1) + f(n+1) * f(m+1) + f(n)f(m) > f(n+m)$ for $n, m \geq 1$.

Proof:
$\forall m \in \mathbb{N},\ m \geq 1, P(m) : \forall n \in \mathbb{N},\ n \geq 1,\ f(n-1)f(m-1) + f(n+1)f(m+1) + f(n)f(m) > f(n+m)$
Prove by complete induction:

Base Case:

1.
$m = 1, f(0) = 0, f(1) = 1, f(2) = f(1) + f(0) = 1$
$\quad f(n-1)f(m-1) + f(n+1)f(m+1) + f(n)f(m)$
$= f(n-1)f(0) + f(n+1)f(2) + f(n)f(1)$
$= f(n+1) + f(n)$
$> f(n{+}1)$ $\sharp$ f(n) is fibonacci function and n is greater or equal to 1, so f(n) is greater than 0
then $P(1)$

2.
$m = 2, f(1) = 1, f(2) = 1, f(3) = f(2) + f(1) = 2$
$\quad f(n-1)f(m-1) + f(n+1)f(m+1) + f(n)f(m)$
$= f(n-1)f(1) + f(n+1)f(3) + f(n)f(2)$
$= f(n-1) + 2f(n+1) + f(n)$
$> f(n+1) + f(n)$
$= f(n+2)$
then $P(2)$

3.
$m = 3, f(2) = 1, f(3) = 2, f(4) = f(3) + f(2) = 3$
$\quad f(n-1)f(m-1) + f(n+1)f(m+1) + f(n)f(m)$
$= f(n-1)f(2) + f(n+1)f(4) + f(n)f(3)$
$= f(n-1) + 3f(n+1) + 2f(n)$
$> f(n+1) + f(n) + f(n+1)$
$= f(n+2) + f(n+1)$
$= f(n+3)$
then $P(3)$

Inductive Step:
Let $m \in \mathbb{N}$

I.H: Assume $\forall 1 \leq i < m, i \in \mathbb{N}, P(i)$
Want To Prove: $P(m)$

Case 1
When $m = 1, 2, 3, P(m)$ $\sharp$ By Base Cases

Case 2
When $m > 3, m-1 \geq 1, m-2 \geq 1$
$f(n+m) = f(n+m-1) + f(n+m-2)$
$f(n+m-1) < f(n-1)f(m-2) + f(n+1)f(m) + f(n)f(m-1)$ $\sharp$ By I.H.
$f(n+m-2) < f(n-1)f(m-3) + f(n+1)f(m-1) + f(n)f(m-2)$ $\sharp$ By I.H.
then $f(n+m-1) + f(n+m-2)$
$< f(n-1)[f(m-2) + f(m-3)] + f(n+1)[f(m) + f(m-1)] + f(n)[f(m-1) + f(m-2)]$
$= f(n-1)f(m-1) + f(n+1)f(m+1) + f(n)f(m)$
$= f(n+m)$
then $P(m)$

Therefore $f(n-1) * f(m-1) + f(n+1) * f(m+1) + f(n)f(m) > f(n+m)$ for $n, m \geq 1$

2. (a) Find a recurrence relation, $T(n)$, for number of distinct full binary trees with n nodes. Show how you find the relation.

Because $T(n)$ is the number of binary trees with n nodes.
Assume when n is very big, then $T(n)$ can be divide into
$T(1)T(n-2) + T(2)T(n-3) + ... + T(n-2)T(1)$
$T(1)T(n-2)$ is the case that left-subtree only has 1 node and right-subtree has n-2 nodes, so we have $T(1)T(n-2)$ kinds of binary tree.
$T(2)T(n-3)$ is the case that left-subtree has 2 node and right-subtree has n-3 nodes, so we have $T(2)T(n-3)$ kinds of binary tree. etc.
Therefore,

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 0 & \text{if } n = 2 \\ \sum_{i=1}^{n-2} T(i)T(n-1-i) & \text{if } n > 2 \end{cases}$$

(b) Without using a closed form, prove $T(n) \geq 2^{(n-1)/2}$ for most odd numbers.

Before Proof:
Because when $n = 2, T(2) = 0$, we know that when n is even, because of $T(2) = 0$, all $T(evennumber)$ are 0.
Proof:
$\forall n \in$ Odd numbers, $n \geq 11, P(n) : T(n) \geq 2^{\frac{n-1}{2}}$
prove by complete induction

Before Base Case:
$T(1) = 1 \leq 2^0$
$T(3) = T(1)T(1) = 1 \leq 2^1$
$T(5) = T(1)T(3) + T(3)T(1) = 2 \leq 2^2$
$T(7) = T(1)T(5) + T(3)T(3) + T(5)T(1) = 5 \leq 2^3$
$T(9) = T(1)T(7) + T(3)T(5) + T(5)T(3) + T(7)T(1) = 14 \leq 2^4$

Base Case:
$n = 11$
$T(11) = T(1)T(9) + T(3)T(7) + T(5)T(5) + T(7)T(3) + T(9)T(1)$
$= 42 \geq 32 = 2^5$
then $P(11)$

Inductive Step:
Let n be odd number
I.H: Assume $\forall 11 \leq i < n, i$ is odd numbers, $P(i)$
WTP: $P(n)$

Case 1:
when $n = 11, P(n)$ $\sharp$ By Base Case

Case 2:
when $n \geq 13, n-2 \geq 11$
$T(n) = \sum_{i=1}^{n-2} T(i)T(n-1-i)$
$= T(1)T(n-2) + ... + T(n-2)T(1)$
$\geq 2T(1)T(n-2)$
$= 2T(n-2)$ ♯ $T(1) = 1$
$\geq 2 \times 2^{\frac{n-3}{2}}$ ♯ By I.H.
$= 2^{\frac{n-1}{2}}$
then $T(n) \geq 2^{\frac{n-1}{2}}$
then $P(n)$

Therefore $T(n) \geq 2^{(n-1)/2}$ for $n \geq 11$ odd numbers

3. (a) Find a recurrence relation, $T(n)$, for number of microorganisms in a microbial culture in which every 2 hours the number of microorganisms is quadrupled and also three times as many of the microorganisms die 4 hours after creation. There are initially 4 microorganisms in the culture.

$\forall n \in \mathbb{N}, n$ is even number,

$$T(n) = \begin{cases} 4 & \text{if } n = 0 \\ 16 & \text{if } n = 2 \\ 4T(n-2) - 3T(n-4) & \text{if } n \geq 4 \end{cases}$$

(b) Without using a closed form, prove $T(n)$ is strictly increasing.

Proof:

$\forall n \in \mathbb{N}, n$ is even number,

$P(n) : T(n) < T(n+2)$

Prove by complete induction.

Base Case:

1.

$n = 0$

$T(0) = 4 < T(2) = 16$

then $P(0)$

2.

$n = 2$

$T(2) = 16$

$T(4) = 4T(2) - 3T(0) = 4 \times 16 - 3 \times 4 = 52$

then $T(2) < T(4)$

then $P(2)$

Inductive Step:

Let n be even number

I.H: Assume $\forall 0 \leq i < n,\ i$ is even numbers, $P(i)$

WTP: $P(n)$

Case 1:

when $0 \leq n < 4, P(n)$ $\sharp$ By Base Case

Case 2:

when $n \geq 4, n - 4 \geq 0$

$$\begin{aligned} T(n+2) &= 4T(n) - 3T(n-2) \\ &= 4[4T(n-2) - 3T(n-4)] - 3T(n-2) \\ &= 16T(n-2) - 12T(n-4) - 3T(n-2) \\ &= 13T(n-2) - 12T(n-4) \end{aligned}$$

$T(n) = 4T(n-2) - 3T(n-4)$
then $T(n+2) - T(n) = 13T(n-2) - 12T(n-4) - 4T(n-2) + 3T(n-4)$
$= 9T(n-2) - 9T(n-4)$
$= 9[T(n-2) - T(n-4)]$
> 0 ♯ By I.H. $T(n-2) > T(n-4)$
then $T(n+2) > T(n)$
then $P(n)$

Therefore $T(n)$ is strictly increasing

(c) Compute the closed form of $T(n)$ using unwinding (repeated substitution).

$T(n) = 4T(n-2) - 3T(n-4)$
$= 4[4T(n-4) - 3T(n-6)] - 3T(n-4)$
$= 16T(n-4) - 12T(n-6) - 3T(n-4)$
$= 13T(n-4) - 12T(n-6)$
$= 13[4T(n-6) - 3T(n-8)] - 12T(n-6)$
$= 52T(n-6) - 39T(n-8) - 12T(n-6)$
$= 40T(n-6) - 39T(n-8)$
$= 40[4T(n-8) - 3T(n-10)] - 39T(n-8)$
$= 160T(n-8) - 120T(n-10) - 39T(n-8)$
$= 121T(n-8) - 120T(n-10)$
...
$= \frac{1}{2}(-1 + 3^{k+1})T(n-2k)) - \frac{3}{2}(-1 + 3^k)T(n-2k-2)$
Substitute $n - 2k = 2$ Then $k = (n-2)/2$
$= \frac{1}{2}(-1 + 3^{\frac{n-2}{2}+1})T(2) - \frac{3}{2}(-1 + 3^{\frac{n-2}{2}})T(0)$
$= \frac{1}{2}(-1 + 3^{\frac{n-2}{2}+1})16 - \frac{3}{2}(-1 + 3^{\frac{n-2}{2}})4$
$= 8(-1 + 3 * 3^{\frac{n-2}{2}}) - 6(-1 + 3^{\frac{n-2}{2}})$
$= -8 + 24 * 3^{\frac{n-2}{2}} + 6 - 6 * 3^{\frac{n-2}{2}}$
$= 18 * 3^{\frac{n-2}{2}} - 2$
$= 2 * 3^{\frac{n-2}{2}+2} - 2$
$= 2 * 3^{\frac{n+2}{2}} - 2$

(d) Prove the closed form, computed in part (c), is correct.

Proof:
Prove $\forall n \in \mathbb{N}$, n is even, $P(n)$: $T(n) = 2 * 3^{\frac{n+2}{2}} - 2$
Prove by complete induction

Basis step:

1.
$n = 0$
$T(0) = 4 = 2 * 3^{\frac{0+2}{2}} - 2$
Then $P(0)$

2.
$n = 2$
$T(2) = 16 = 2 * 3^{\frac{2+2}{2}} - 2$
Then $P(2)$

Inductive step:
Let n be even number
Inductive Hypothesis: Assume $\forall 0 \leq i < n, i$ is even, $P(i)$
WTP: $P(n)$

Case 1:
when $n = 0, 2, P(n)$ $\sharp$ By Base Cases

Case 2:
when $n > 2, n - 2 \geq 0, n - 4 \geq 0$
Because $T(n) = 4T(n-2) - 3T(n-4)$

$$
\begin{aligned}
&= 4(2 * 3^{\frac{n}{2}} - 2) - 3(2 * 3^{\frac{n-2}{2}} - 2) \text{ \# By I.H} \\
&= 8 * 3^{\frac{n}{2}} - 8 - 6 * 3^{\frac{n-2}{2}} + 6 \\
&= 24 * 3^{\frac{n-2}{2}} - 6 * 3^{\frac{n-2}{2}} - 2 \\
&= 18 * 3^{\frac{n-2}{2}} - 2 \\
&= 2 * 3^{\frac{n-2}{2}+2} - 2 \\
&= 2 * 3^{\frac{n+2}{2}} - 2
\end{aligned}
$$

Then $P(n)$ holds

Therefore, $\forall n \in \mathbb{N}$, n is even, $T(n) = 2 * 3^{\frac{n+2}{2}} - 2$

4. (a) Find a recurrence relation, $T(n)$, for number of distinct ways that a postage of n cents can be made by 4-cent, 6-cent, and 10-cent stamps for most even numbers.

$\forall n \in \mathbb{N}, n \geq 4$ and n is even number,

$$T(n) = \begin{cases} 1 & \text{if } n = 4 \\ 1 & \text{if } n = 6 \\ 1 & \text{if } n = 8 \\ 2 & \text{if } n = 10 \\ 2 & \text{if } n = 12 \\ 2 & \text{if } n = 14 \\ 3 & \text{if } n = 16 \\ 3 & \text{if } n = 18 \\ 4 & \text{if } n = 20 \\ 4 & \text{if } n = 22 \\ T(n-4) + T(n-6) + T(n-20) - T(n-14) - T(n-16) & \text{if } n \geq 24 \end{cases}$$

(b) Without using a closed form, prove $T(n)$ is non-decreasing.

Proof:

$\forall n \in \mathbb{N}, n \geq 4$ and n is even number,

$P(n) : T(n) \leq T(n+2)$

This question cannot prove by induction.

5. (a) Devise a brute-force algorithm in Python[1] notation, **max-sum**, to find the largest sum of consecutive terms of a sequence of n positive and negative numbers.

```
def max_sum(Array):
1.  L = []
2.  for i in range(len(Array)):
3.      for j in range(len(Array)):
4.          L.append(sum(Array[i:j+1]))
5.  return max(L)
```

(b) Find the worst-case time complexity of **max-sum**.

Let n be the size of parameter Array.
Compute the worse-case time complexity $T(n)$.
By line 2, the worst-case time complexity is n because it is a for loop depends on size of Array, which is n. Similarly, line 3 is also n. On line 4, the built-in function sum is also n and the last line is 1. Therefore, $T(n) = (\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(j-i)) + 1 \in \theta(n^3)$.

(c) Devise a divide-and-conquer algorithm to find **max-sum**, by splitting the sequence to halves, and finding **max-sum** in each. Note that the maximum sum of consecutive terms can include terms in both halves.

```
def max_sum(Array, begin, end):
    if begin > end:
        return -2**31
    mid = (begin + end) // 2
    Leftmax = max_sum(Array, begin, mid-1)
    Rightmax = max_sum(Array, mid+1, end)
    result = max(Leftmax,Rightmax)
    summation = 0
    midleftmax = 0
    for i in range(begin, mid)[::-1]:
        summation += Array[i]
        if(summation > midleftmax):
            midleftmax = summation
    summation = 0
    midrightmax = 0
    for i in range(mid+1, end+1):
        summation += Array[i]
        if(summation > midrightmax):
            midrightmax = summation
    result = max(result, midleftmax + midrightmax + Array[mid])
    return result
```

[1] or any other common programming language

(d) Find a recurrence relation for the worst-case time complexity of the divide-and-conquer **max-sum**.

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n + 1 & \text{if } n \geq 2 \end{cases}$$

Justification:
When size n is equal to 1: $T(n)$ is clearly 1 since there is no recursive step and no any for loops.
When size n is greater or equal than 2: By line 4, it's a recursive step and the size of n becomes the floor function of $n/2$. Then it is $T(\lfloor \frac{n}{2} \rfloor)$. Similarly, line 5 is also a recursive step of the ceiling function of $n/2$, then it is $T(\lceil \frac{n}{2} \rceil)$. Line 9 is a for loop depends on half size of Array, then it is n/2. So is the for loop on line 15. Last, the return statement counts 1.

(e) How does the time complexity of the the divide-and-conquer algorithm compare with that of the brute-force one?

For divide-and-conquer algorithm:
By Master Theorem: $a_1 = 1, a_2 = 1, f(n) = n + 1, b = 2, d = 1$.
Then $a = a_1 + a_2 = 2, f(n) \in \theta(n^d) = \theta(n)$
Then $a = b^d = b = 2 \implies T(n) \in \theta(n \log n)$
Apparently $nlogn$ is less than n^3
Therefore we conclude that divide-and-conquer algorithm is a faster and more efficient algorithm compare to the brute-force algorithm.