CSC236 Fall 2016
Assignment #2
due November 4th, 10 p.m.

These problems are to give you some practice proving facts about recurrences, and the time complexity of algorithms that are expressed as recurrences.

Your assignment must be typed to produce a PDF document **a2.pdf** (hand-written submissions are not acceptable). You may work on the assignment in groups of **1 or 2** and submit a single assignment for the entire group on MarkUs. Your teammates must be **different** than all teammamtes you had in Assignment 1.

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 $n, m \geq 1$.
Let P(n) denotes: for all m $\geq$ 1, $f(n-1) * f(m-1) + f(n+1) * f(m+1) + f(n)f(m) > f(n+m)$
Proof By Complete Induction.
Inductive Step:
Inductive Hypothesis: Assume P(i) for 1 $\leq$ i < n.
i.e: $f(i-1) * f(m-1) + f(i+1) * f(m+1) + f(i)f(m) > f(i+m)$
Want to Show: P(n)
i.e $f(n-1) * f(m-1) + f(n+1) * f(m+1) + f(n)f(m) > f(n+m)$
Case1(Base case): n = 1 or n = 2:
since f(0)*f(m - 1) + f(2)*f(m + 1) + f(1)f(m)
= f(m + 1) + f(m) = f(m + 2) # By definition.
$\geq$ f(m + 1) # Since fibonacci function is non-decreasing.
So, P(1) holds.
since f(1)*f(m - 1) + f(3)*f(m + 1) + f(2)f(m)
= f(m - 1) + 2f(m + 1) + f(m).
= 3f(m + 1)
$\geq$ f(m + 1) + f(m) + f(m + 1) # Since fibonacci function is non-decreasing.
$\geq$ f(m + 2) # Since each term of fibonacci function is non-negative.
So, P(2) holds.
Case2: n $\geq$ 3
then 1 $\leq$ n - 2 < n and 1 $\leq$ n - 1 < n.
P(n - 2) holds and P(n - 1) holds.
we have two expression.
f(n - 3)*f(m - 1) + f(n - 1)*f(m + 1) + f(n - 2)f(m) $\geq$ f(n - 2 + m) (1) # By I.H

f(n - 2)*f(m - 1) + f(n)*f(m + 1) + f(n - 1)f(m) ≥ f(n - 1 + m) (2) # By I.H
Sum up (1) and (2) on both right hand side and left hand side.
we get:
f(n - 3)*f(m - 1) + f(n - 1)*f(m + 1) + f(n - 2)f(m) + f(n - 2)*f(m - 1) + f(n)*f(m + 1) + f(n - 1)f(m) ≥ f(n - 2 + m) + f(n - 1 + m)
Recombination left hand side.
(f(n - 3)*f(m - 1) + f(n - 2)*f(m - 1)) + (f(n - 1)f(m + 1) + f(n)f(m + 1)) + (f(n)f(m) + f(n - 1)f(m).
= f(n - 1)f(m - 1) + f(n)f(m + 1) + f(n + 1)f(m) # By Definition.
Simplify right hand side.
f(n - 2 + m) + f(n - 1 + m) = f(n + m) # By Definition.
Since left hand side is greater than or equal to left hand side.
So, f(n - 1)f(m - 1) + f(n)f(m + 1) + f(n + 1)f(m) ≥ f(n + m)
P(n) hold in this case.
We have proven P(n) holds in all the case n ≥ 1. So, $f(n-1)*f(m-1)+f(n+1)*f(m+1)+f(n)f(m) > f(n+m)$ 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.
We notice that for full binary trees, the number of nodes must be odd.
So for even number, we cannot form full binary tree.
Consider n is odd:
The full binary tree has right subtree and left subtree.
In the first case right subtree has 1 nodes, has T(1) distinct FBT left subtree has n - 1 - 1 nodes. has T(n-2) distinct FBT.
We have T(1) * T(n-2) distinct FBT in this case
In the second case right subtree has 2 nodes,has T(2) distinct FBT, left subtree has n - 1 - 2 nodes. has T(n-3) distinct FBT
We have T(2) * T(n-3) distinct FBT in this case but since 2 is even and n - 3 is even , this case sum up to 0
In the third case right subtree has 3 nodes, has T(3) distinct FBT, left subtree has n - 1 - 3 nodes. has T(n-4) distinct FBT.
We have T(3) * T(n-4) distinct FBT in this case
and so on.
The total number of distinct number of FBT is just the sum of all above.
T(n) = $\sum_{i=1}^{n-2} T(i) * T(n-i-1)$
In summary:

$$T(n) = \begin{cases} 0 & \text{n is even} \\ 1 & \text{n} = 1 \\ \sum_{i=1}^{n-2} T(i) * T(n-i-1) & \text{n is odd and n} \geq 3 \end{cases}$$

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

Let P(n) denotes T(n) ≥ $2^{(n-1)/2}$
T(1) = 1
T(3) = T(1)T(1) = 1 < $2^{(3-1)/2} = 2$
T(5) = T(1)T(3) + T(2)T(2) +T(3)T(1) = 1 + 0 + 1 = 2 < $2^{(5-1)/2} = 4$
T(7) = T(1)T(5) + T(2)T(4) + T(3)T(3) + T(4)T(2) + T(5)T(1) = 2*1 + 0 + 1*1 + 0 + 1*2 = 5 < $2^{(7-1)/2} = 8$

T(9) = T(1)T(7) + T(2)T(6) + T(3)T(5) + T(4)T(4) + T(5)T(3) + T(6)T(2) + T(7)T(1) = 5*1 + 0 + 1*2 + 0 + 1*2 + 0 + 1*5 = 14 $< 2^{(9-1)/2} = 16$
T(11) = T(1)T(9) + T(2)T(8) + T(3)T(7) + T(4)T(6) + T(5)T(5) + T(6)T(4) + T(7)T(3) + T(8)T(2) + T(9)T(1) = 14*1 + 0 + 1*5 + 0 + 2*2 + 0 + 1*5 + 0 + 14*1 = 42 $\geq 2^{(11-1)/2} = 32$
So we should start from n = 11 and n is odd.
Proof By Complete Induction.
Inductive Step:
Inductive Hypothesis:
Assume P(i) holds for $11 \leq$ i $<$ n and i is odd.
Want to Show: P(n) holds for n is odd.
Case1:(Base case) n = 11
P(11) holds since T(11) = 42 and $2^{(11-1)/2} = 32$
$T(11) \geq 2^{(11-1)/2}$
Case2: n $\geq$ 13.
$T(n) = \sum_{i=1}^{n-2} T(i) * T(n-i-1)$ #By Definition.
$\geq$ T(n-2)T(1) + T(1)T(n-2) # Since T(n-2)T(1) and T(1)T(n-2) are two terms of the sum and each term is greater than or equal to 0
= 2T(n-2) # T(1) = 1
Since $11 \leq$ n - 2 $<$ n, we may apply our I.H here.
$\geq$ $2*2^{(n-3)/2}$ # By I.H
$\geq 2^{(n-1)/2}$
So, P(n) holds this case.
We have proven P(n) holds for all case n $\geq$ 11 and n is odd.# Most odd number.

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.

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

(b) Without using a closed form, prove $T(n)$ is strictly increasing.
Proof By Simple Induction.
Let P(n) denotes T(n+2) - T(n) $>$ 0
Base case:
P(0) holds. Since T(2) = 4*4 - 3*0 = 16 and T(0) = 4 16-4 = 12$>$ 0
Inductive Step:
Inductive Hypothesis: Assume P(n) holds for n $>$ 0.
i.e: T(n+2) - T(n) $>$ 0.
Want to show: P(n+2) holds.
i.e: T(n+4) - T(n+2) $>$ 0.
T(n+4)-T(n+2) = (4T(n+2) - 3T(n)) - T(n+2) By Definition.
= 3(T(n+2) - T(n))
$>$ 0 By Inductive Hypothesis:T(n+2) - T(n) $>$ 0
then T(n+4) - T(n+2) $>$ 0.
then P(n+2) holds.

then 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)
= (4*4 - 3)T(n - 4) - (4*3)T(n - 6)
= (4*4 - 3)(4T(n - 6) - 3T(n - 8)) - (4*3)T(n - 6)
= (4*4*4 - 3*4 - 4*3)T(n - 6) - (4*4*3 - 3*3)T(n - 8)
= .....
= $\frac{3^{n/2}-1}{2}$ T(n - (n - 2)) + $\frac{3^{n/2}-3}{2}$ T(n - n)
= $\frac{3^{n/2}-1}{2}$ * 16 + $\frac{3^{n/2}-3}{2}$ * 4
= 6 * $3^{n/2}$ - 2

(d) Prove the closed form, computed in part (c), is correct.
Proof by Complete Induction.
Let P(n) denotes T(n) = 6 * $3^{n/2}$ - 2
Inductive Step:
Inductive Hypothesis: Assume P(i)
i.e : T(i) = 6 * $3^{i/2}$ - 2 for $0 \leq$ i < n.
Want to show P(n) i.e: T(n) = 6 * $3^{n/2}$ - 2
Base case n = 0:
P(0) holds since
6 * $3^{0/2}$ - 2 = 4 = T(0) # By Definition.
Base case n = 2:
P(2) holds since
6 * $3^{2/2}$ - 2 = 16 = T(2) # By Definition.
Case $n \geq 4$ and n is even:
Since $n \geq 4$ and n is even
then $0 \leq$ n - 2 < n and $0 \leq$ n - 4 < n
and n - 2 and n - 4 is even. # We may apply I.H when needed.
T(n) = 4T(n - 2) - 3T(n - 4) # By Definition
= 4 * (6 * $3^{(n-2)/2}$ - 2) - 3 * (6 * $3^{(n-4)/2}$ - 2) #By I.H
= 24 * $3^{(n-2)/2}$ - 18 * $3^{(n-4)/2}$ - (4 * 2 - 3 * 2)
= 8 * $3^{(n)/2}$ - 2 * $3^{(n-4)/2}$ - 2
= 6 * $3^{(n)/2}$ - 2
So, P(n) holds in this case.
We have proven P(n) holds in all case $0 \leq$ n # And Verify Base case before!

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. First we consider we use how many 10 cents at most to form this postage.
For example, if we have 20 cents, we can use 0 10-cent, 1 10-cents, or 2 10-cents.
then we consider use 4 and 6 to fill up.

$$T(n) = \begin{cases} 1 & \text{n = 4 or n = 6 or n = 8} \\ 2 & \text{n = 10 or n = 12 or n = 14} \\ T(n-10)+\lfloor (n+6)/12 \rfloor & \text{n = 4k + 2 for some k and n} \geq 14 \\ T(n-10)+\lfloor (n+12)/12 \rfloor & \text{n = 4k for some k and n} \geq 14 \end{cases}$$

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

Proof By Complete Induction.
Let P(n) denotes T(n + 2) - T(n) $\geq$ 0.
Inductive Step:
Inductive Hypothesis:
Assume P(i) holds for 4 $\leq$ i < n and i is even.
i.e: T(i + 2) - T(i) $\geq$ 0
Want to show: P(n) holds for n is even.
i.e: T(n + 2) - T(n) $\geq$ 0
Base case: n < 24
I make some tuples here, the first,second and third element stand for number of 10-cent, 6-cent and 4-cent use respectively and the number behind them is the number of total distinct ways.
For Postage 4 : (0 , 0 , 1) 1
For Postage 6 : (0 , 1 , 0) 1
For Postage 8 : (0 , 0 , 2) 1
For Postage 10 : (0 , 1 , 1) (1 , 0 , 0) 2
For Postage 12 : (0 , 0 , 3) (0 , 2 , 0) 2
For Postage 14 : (0 , 1 , 2) (1 , 0 , 1) 2
For Postage 16 : (0 , 0 , 4) (0 , 2 , 1) (1 , 1 , 0) 3
For Postage 18 : (0 , 1 , 3) (0 , 3 , 0) (1 , 0 , 2) 3
For Postage 20 : (0 , 0 , 5) (0 , 2 , 2) (1 , 1 , 1) (2 , 0 , 0) 4
For Postage 22 : (0 , 1 , 4) (0 , 3 , 1) (1 , 0 , 3) (1 , 2 , 0) 4
For Postage 24 : (0 , 0 , 6) (0 , 2 , 3) (0 , 4 , 0) (1 , 1 , 2) (2 , 0 , 1) 5
Easily to check that P(n) holds for n < 24
Case:n $\geq$ 24
  Case:n = 4k + 2 for some k
  then n + 2 = 4k + 4 = 4(k + 1)
  T(n + 2) - T(n)
  $= T(n-8) + \lfloor (n+14)/12 \rfloor - T(n-10) - \lfloor (n+6)/12 \rfloor$ # By Definition.
  $= (T(n-8) - T(n-10)) + (\lfloor (n+14)/12 \rfloor - \lfloor (n+6)/12) \rfloor)$ # Regroup.
  Since n $\geq$ 24
  then 4 $\leq$ n - 8 < n and 4 $\leq$ n - 10 < n and both of (n-8) and (n-10) is even.
  then (T(n - 8) - T(n - 10)) $\geq$ 0 # By I.H
  then T(n + 2) - T(n) $\geq (\lfloor (n+14)/12 \rfloor - \lfloor (n+6)/12) \rfloor)$
  $\geq$ 0 # Since n + 14 > n + 6
  Case:n = 4k for some k
  then n + 2 = 4k + 2 for the same k
  T(n + 2) - T(n)
  $= T(n-8) + \lfloor (n+8)/12 \rfloor - T(n-10) - \lfloor (n+12)/12 \rfloor$ # By Definition.
  $= (T(n-8) - T(n-10)) + (\lfloor (n+8)/12 \rfloor - \lfloor (n+12)/12) \rfloor)$ # Regroup.
  $= (T(n-18) + \lfloor (n+4)/12 \rfloor - T(n-20) - \lfloor (n-4)/12 \rfloor) + (\lfloor (n+8)/12 \rfloor - \lfloor (n+12)/12) \rfloor)$ # By

Definition.

$= (T(n-18) - T(n-20)) + (\lfloor (n+4)/12 \rfloor - \lfloor (n-4)/12 \rfloor + \lfloor (n+8)/12 \rfloor - \lfloor (n+12)/12) \rfloor)$

$= (T(n-18) - T(n-20)) + (\lfloor (4k+4)/12 \rfloor - \lfloor (4k-4)/12 \rfloor + \lfloor (4k+8)/12 \rfloor - \lfloor (4k+12)/12) \rfloor)$ #

Plug in k.

SubCase1: k = 3j for some j #i.e : k is multiple of 3

$= (T(n-18) - T(n-20)) + (\lfloor (12j+4)/12 \rfloor - \lfloor (12j-4)/12 \rfloor + \lfloor (12j+8)/12 \rfloor - \lfloor (12j+12)/12) \rfloor)$

$= (T(n-18) - T(n-20)) + (\lfloor j + 4/12 \rfloor - \lfloor j - 4/12 \rfloor + \lfloor (j + 8/12 \rfloor - \lfloor (j+1) \rfloor)$

= (T(n - 18) - T(n - 20)) + (j - (j - 1) + j - (j + 1)

= (T(n - 18) - T(n - 20))

Since n $\geq$ 24

then 4 $\leq$ n - 18 < n and 4 $\leq$ n - 20 < n and both of (n-18) and (n-20) is even.

then (T(n - 18) - T(n - 20)) $\geq$ 0 # By I.H

then T(n + 2) - T(n) $\geq$ 0

SubCase2: k = 3j + 1 for some j i.e : k // 3 = 1

$= (T(n-18) - T(n-20)) + (\lfloor (12j+8)/12 \rfloor - \lfloor (12j)/12 \rfloor + \lfloor (12j+12)/12 \rfloor - \lfloor (12j+16)/12) \rfloor)$

= (T(n - 18) - T(n - 20)) + (j - j + (j + 1) - (j + 1))

= (T(n - 18) - T(n - 20)) Since n $\geq$ 24.

then 4 $\leq$ n - 18 < n and 4 $\leq$ n - 20 < n and both of (n-18) and (n-20) is even.

then (T(n - 18) - T(n - 20)) $\geq$ 0 # By I.H

then T(n + 2) - T(n) $\geq$ 0

SubCase3: k = 3j + 2 for some j i.e: k // 3 = 2

$= (T(n-18) - T(n-20)) + (\lfloor (12j+12)/12 \rfloor - \lfloor (12j+4)/12 \rfloor + \lfloor (12j+16)/12 \rfloor - \lfloor (12j+20)/12) \rfloor)$

= (T(n - 18) - T(n - 20)) + (j + 1) - j + (j + 1) - (j + 1))

= (T(n - 18) - T(n - 20)) + 1 Since n $\geq$ 24.

then 4 $\leq$ n - 18 < n and 4 $\leq$ n - 20 < n and both of (n-18) and (n-20) is even.

then (T(n - 18) - T(n - 20)) $\geq$ 0 # By I.H

then T(n + 2) - T(n) $\geq$ 0

In all case, we have proven T(n + 2) - T(n) $\geq$ 0

So, P(n) holds for all n $\geq$ 4 and n is even.

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(nums):
    if len(nums) == 0:
        return 0
    sum = nums[0]
    for i in range(len(nums)):
        sub_sum = 0
        # Consider all the possible subsequence start from i.
        for j in range(i, len(nums)):
            sub_sum += nums[j]
            if sub_sum > sum:
                sum = sub_sum
    return sum
```

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

[1] or any other common programming language

Let n denotes the length of list nums.
when n is 0, this function terminate in 1 time.
when n is greater than or equal to 1.
we go through 2 steps before go to for loop.
We have two for loops here.
the outer for loops have n iterations. each iterations has 1 step a inner loop.
the inner for loops have n - i iterations. each iterations has 3 step in worst case.
we have 1 step lastly for return.
So, we have 2 + 3 * $\sum_{i=1}^{n-1} i$ + n + 2 steps in total.
T(n) = 2 + 3n(n - 1)/ 2 + n + 2
T(n) = $\frac{3}{2}$ n^2 + $\frac{5}{2}$n + 4
So, the time complexity is T(n) $\in$ $\theta(n^2)$.

(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(nums):
    # Base case: empty list.
    if len(nums) == 0:
        return 0
    # Base case: list with only one element.
    if len(nums) == 1:
        return nums[0]
    # Find the middle index.
    m = len(nums) // 2
    left_sum = 0
    left_max = nums[m-1]
    right_sum = 0
    right_max = nums[m]
    for j in range(m-1,-1,-1):
        # we want to find left max sum here.
        left_sum += nums[j]
        if left_sum > left_max:
            left_max = left_sum
    for k in range(m,len(nums)):
        # we want to find right max sum here.
        right_sum += nums[k]
        if right_sum > right_max:
            right_max = right_sum
    # Return the max of the left part's max and right part's max and max through two parts.
    return max(left_max + right_max, max_sum(nums[:m]), max_sum(nums[m:]))
```

(d) Find a recurrence relation for the worst-case time complexity of the divide-and-conquer **max-sum**.
Let n denotes the length of list nums.
for base case n = 0 or n = 1, the program terminate in 1 step.
for case n is greater than or equal to 2.
we have 7 steps before go to for loops.

we have two for loops here.
first for loop has $\lfloor n/2 \rfloor$ iterations. each iteration have 3 steps for worst-case.
second for loop has $\lceil n/2 \rceil$ iterations. each iteration have 3 steps for worst-case.
max_sum of a list of length $\lfloor n/2 \rfloor$ runs in $T(\lfloor n/2 \rfloor)$ times.
max_sum of a list of length $\lceil n/2 \rceil$ runs in $T(\lceil n/2 \rceil)$ times.
return statement take 1 step.
In summary:

$$T(n) = \begin{cases} 1 & n = 0 \text{ or } n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 3n + 7 + 1 & n \geq 2 \end{cases}$$

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

when $n = 2^k$ for an natural number k.
we can get rid of floor and ceiling.

$$T(n) = \begin{cases} 1 & n = 0 \text{ or } n = 1 \\ T(n/2) + T(n/2) + 3n + 7 + 1 & n \geq 2 \end{cases}$$

Use master theorem to solve:

$$T(n) = \begin{cases} 1 & n = 0 \text{ or } n = 1 \\ T(n/2) + T(n/2) + 3n + 8 & n \geq 2 \end{cases}$$

$a = 2$ $b = 2$ and $f(n) = 3n + 8$
$T(n) \in \theta(n * log(n))$
Apparently, this algorithms has faster worst time complexity than the brute force one.