Homework 1 | Last Updated: September 14, 2020
CSCI-SHU 220 | Due Date: September 28, 2020, 11:00pm

---

You are allowed to discuss with others but not allow to use references other than the course notes and reference books. Please list your collaborators for each questions. Write your own solutions and make sure you understand them.

There are 60 marks in total (including the bonus). The full mark of this homework is 50.

Enjoy :).

*Please specify the following information before submission*:

- Your Name: Zhenming Wang
- Your NetID: zw1806
- Collaborators:

## Problem 1: Asymptotics [5 marks]

Arrange the following functions in order of increasing growth rate, with $g(n)$ following $f(n)$ in your list if and only if $f(n) = O(g(n))$.

- $\log_2 \log_2 n$
- $n^3$
- $n^{\log_2 n}$
- $\sqrt{\log_2 n}$
- $n^{1/\log_2 n}$

*Solution.*

- $n^{1/\log_2 n}$
- $\log_2 \log_2 n$
- $\sqrt{\log_2 n}$
- $n^3$
- $n^{\log_2 n}$

(a) we would like to show that $n^{1/\log_2 n} = O(\log\log n)$
according to the definition of asymptotic upper bound, this means that there exists two constant $k_0$, $n_0$,such that $n^{1/\log_2 n} \leq k_0 \cdot \log\log n$ for $n \geq n_0$.

$$\begin{aligned} n^{1/\log_2 n} &= n^{\log_2 n^{-1}} \\ &= n^{-\log_2 n} \end{aligned}$$

When $n \geq 2$,

$$\begin{aligned} \log_2 n \geq 1 \\ \Rightarrow -\log_2 n \leq -1 \end{aligned}$$

When $n \geq 4$

$$\log\log n \geq 1$$

So, when $n \geq 4$, we would have

$$\begin{aligned} -\log_2 n < \log\log n \\ n^{-\log_2 n} < n^{\log\log n} \end{aligned}$$

To make our property $n^{1/\log_2 n} \leq \log\log n$ be true, we choose $k_0 = 1$, $n_0 = 4$.

(b) we would then like to show that $\log\log n$ is $O(\sqrt{\log_2 n})$, which means we need to find constant $k_1$ and $n_1$ such that for $n \geq n_1$, $\log\log n \leq k_1 \cdot \sqrt{\log_2 n}$.

$$\begin{aligned} \sqrt{\log_2 n} &= (\log_2 n)^{\frac{1}{2}} \\ &= \log_2 n^{\frac{1}{2}} \end{aligned}$$

For large enough $n$

$$\begin{aligned} \log_2 n < \sqrt{n} \\ \log logn < log\sqrt{n} \end{aligned}$$

Thus, we could choose $k_1 = 1$,$n_1 = 2^9$, so that for $n \geq n_1$, $\log\log n < \log\sqrt{n}$.
$\log\log n$=$O(\log\sqrt{n})$.

(c) we would then like to show that $\sqrt{\log_2 n} = O(n^3)$, which means there exists constants $k_2$,$n_2$, so that for $n \geq n_2$, $\sqrt{\log_2 n} \leq k_2 \cdot (n^3)$.

$$\sqrt{\log_2 n} = \frac{1}{2}\log_2 n$$

It is well known that for large enough n,

$$\log n < n^3$$

So, we could choose $k_2 = 1$,$n_2 = 2$, such that for $n \geq n_2$, $\sqrt{\log_2 n} < n^3$.
$\sqrt{\log_2 n} = O(n^3)$

(d) We now would like to show that $n^3$ is $O(n^{\log_2 n})$.
For $n > 8$, $\log_2 n > 3$
We could easily satisfy the property by picking $k_3 = 1$,$n_3 = 8$, so that $n^3 = O(n^{\log n})$

## Problem 2: Solving recurrences [15 marks]

(a) (10 marks) Find an asymptotically tight bound of the following recurrence relations. Justify your answers by naming a particular case of the Master method, or by iterating the recurrence, or by using the substitution method. Assume that the base cases can be solved in constant time.

(i) $T(n) = 2T(n/4) + 2n$

(ii) $T(n) = T(n-2) + n^2$

(iii) $T(n) = 2T(2n/3) + T(n/3) + n^2$

(b) (5 marks) Consider the recurrence relation $C_0 = 0$ and

$$C_n = n + 1 + \frac{2}{n}\sum_{k=0}^{n-1} C_k.$$

Find an explicit formula for $C_n$.

*Solution.* **Please write down your solution to Problem 2 here.**

(a) Problem a

(i) $T(n) = 2T(n/4) + 2n$

i. We start by guessing that $T(n) \leq kn$, where $k$ is some constant. We now prove our guess using induction.
**Base case**: $T(1) \leq k * 1$, which is true since base case $T(1)$ is some constant.
**Inductive case**: we assume that our property is true for $n/4$, that $T(n/4) \leq k(n/4)$. We need to show it also holds true for n.

$$\begin{aligned} T(n) &= 2T(n/4) + 2n \\ &\leq 2k(n/4) + 2n \\ &= (k/2)n + 2n \\ &= (k/2 + 2)n \\ &\leq kn \end{aligned}$$

Our property is true as long as $k \geq 4$.To finish our proof, we let $k = 4$, so that $T(n) \leq kn$ for $n \geq 1$.

ii. Now we would try to show that $T(n) \geq c * n$, where c is some constant greater than 0.
**Base case**: $T(1) \geq c(1)$, which is true since s is some constant and $T(1)$ is some constant. We only need to make $T(1) \geq c$ to make the base case work.
**Inductive case**:we assume that our property is true for $n/4$, that $T(n/4) \geq c(n/4)$,

we need to show it also holds true for n.

$$\begin{aligned} T(n) &= 2T(n/4) + 2n \\ &\geq 2c(n/4) + 2n \\ &= (c/2 + 2)n \\ &\geq cn \end{aligned}$$

This is true as long as $c \leq 4$, hence we have shown that $T(n) \geq cn$ for some constant $c$, to finish up, we pick $c = 4$.

So, $T(n)$ has an asymptotic upper bound $O(n)$ and a lower bound of $\Omega(n)$, so $T(n)$ has a tight bound of $\Theta(n)$.

(ii) $T(n) = T(n-2) + n^2$

i. We start by guessing that $T(n) \leq kn^3$, where k is some constant $\geq 0$
**Base Case**:$T(1) \leq k * (1)^3$, which holds true as long as $k \geq T(1)$, since $T(1)$ is some constant.
**Inductive Case**: We assume our property is true for $n-2$, we need to show that it also holds for $n$.

$$\begin{aligned} T(n) &= T(n-2) + n^2 \\ &\leq k(n-2)^3 + n^2 \\ &= k(n^3 - 6n^2 + 12n - 8) + n^2 \\ &= kn^3 - ((6k-1)n^2 - 12kn + 8k) \end{aligned}$$

if

$$T(n) \leq kn^3$$

then

$$(6k-1)n^2 - 12kn + 8k \geq 0$$

according to the property of quadratic function

$$\begin{aligned} \text{let} \qquad \Delta &= ((-12)k)^2 - 4(6k-1) * 8k \leq 0 \\ \Delta &= 32k - 48k^2 \leq 0 \\ &\Rightarrow k \geq \frac{2}{3} \end{aligned}$$

Hence, we have shown that our property holds true for $k \geq \frac{2}{3}$, to finish our proof, we choose $k = \frac{2}{3}$.
We have shown that when $k = \frac{2}{3}$,$n \geq 1$,$T(n) \leq kn^3$. so $T(n)$ has an upper bound of $O(n^3)$.

ii. Now we want to show that $T(n) \geq c * n^3$, where c is another constant $\geq 0$.
**Base Case**: $T(1) \geq c(1)^3$, which is true since both sides are constants.
**Inductive Case**: We assume our property is true for $n-2$, we would like to show it also holds for $n$.

$$\begin{aligned} T(n) &= T(n-2) + n^2 \\ &\geq c(n-2)^3 + n^2 \\ &= kn^3 - ((6k-1)n^2 - 12kn + 8k) \\ &\geq kn^3 \end{aligned}$$

Our property holds true when the equation $(6k-1)n^2 - 12kn + 8k \leq 0$

$$\begin{aligned} \Delta = ((-12)k)^2 - 4(6k-1)8k \geq 0 \\ 6k - 1 < 0 \end{aligned}$$

Solve the above two equation we could get

$$\begin{aligned} k < \frac{1}{6} \\ 0 \leq k \leq \frac{2}{3} \end{aligned}$$

So we our property is true as long as $0 < k < \frac{1}{6}$. Now we have proved that $T(n) \geq kn^3$, for some constant $0 < k < \frac{1}{6}$. So, T(n) has a lower bound of $\Omega(n^3)$.

Hence, we have shown that $T(n) = \Theta(n^3)$.

(iii) $T(n) = 2T(2n/3) + T(n/3) + n^2$

i. We guess that $T(n) \leq kn^2 \log n$, where $k$ is some constant $\geq 0$. Now we try to prove this by induction.
**Base Case**: $T(2) \leq k * 2^2 * log2 \Rightarrow T(2) \leq 4k$, which is true since our base case $T(2)$ is some constant.
**Inductive Case**: we assume our property is true for $2n/3$ and $n/3$, we need to show that it is also true for $n$.

$$\begin{aligned} T(n) &= 2T(2n/3) + T(n/3) + n^2 \\ &\leq 2[k(2n/3)^2 \log{(2n/3)}] + k(n/3)^2 log(n/3) + n^2 \\ &= \frac{8}{9}kn^2 log(2n/3) + \frac{k}{9}n^2 log(n/3) + n^2 \end{aligned}$$

using the property of log, we know that $logab = loga + logb$, $loga/b = loga - logb$, hence we arrange our $T(n)$ and get

$$\begin{aligned} T(n) &\leq kn^2 logn + \frac{8}{9}kn^2 - kn^2 log3 + n^2 \\ &\leq kn^2 logn \end{aligned}$$

The last relationship holds true if $(\frac{8}{9}k + 1 - k\log 3)n^2 \leq 0$, since $n^2 \geq 0$, we need $(\frac{8}{9}k + 1 - k\log 3) \leq 0$

$$(\frac{8}{9} - log3)k \leq -1$$

since $\frac{8}{9} - \log 3 < 0$, this relationship holds true as long as we choose a $k \geq 2$. So we have finished with proving our property, that $T(n) \leq kn^2 logn$. $T(n)$ has an asymptotic upper bound of $O(n^2 \log n)$.

ii. Now we would try to show that $T(n) \geq cn^2 \log n$, where $c$ is some constant, $c \geq 0$.
**Base Case**: $T(2) \geq k(2)^2 \log 2 \Rightarrow T(2) \geq 4c$, which holds true since both sides are constant.
**Inductive Case**: we assume our property is true for $T(2n/3)$ and $T(n/3)$, we need to show that it's also true for $n$.

$$\begin{aligned} T(n) &= 2T(2n/3) + T(n/3) + n^2 \\ &\geq 2[c(2n/3)^2 \log{(2n/3)}] + c(n/3)^2 log(n/3) + n^2 \\ &= cn^2 \log n + \frac{8}{9}cn^2 - cn^2 log3 + n^2 \\ &\geq cn^2 logn \end{aligned}$$

To make the last relationship be true, we need

$$\frac{8}{9}cn^2 - cn^2 log3 + n^2 \geq 0$$

which is

$$c(\frac{8}{9} - log3) \geq -1$$

since we know $0 \geq \frac{8}{9} - log3 \geq -1$, we could satisfy this relationship by choosing $c = 1$. So, we have $T(n) \geq cn^2 logn$, where $c = 1$. So, $T(n)$ has an asymptotic lower bound of $\Omega(n^2 logn)$.

Hence, we have shown that $T(n) = \Theta(n^2 logn)$.

(b) **Problem b**

According to the property given, we could have

$$C_n = n + 1 + \frac{2}{n}\sum_{k=0}^{n-1} C_k$$

$$C_{n-1} = (n-1) + 1 + \frac{2}{n-1}\sum_{k=0}^{n-2} C_k$$

By $C_n - C_{n-1}$, we have

$$C_n - C_{n-1} = 1 + \frac{2}{n}\sum_{k=0}^{n-1} C_k - \frac{2}{n-1}\sum_{k=0}^{n-2} C_k$$

To get rid of the denominator, we multiply $n(n-1)$ at both sides

$$n(n-1)(C_n - C_{n-1}) = n(n-1) + 2(n-1)\sum_{k=0}^{n-1} C_k - 2n\sum_{k=0}^{n-2} C_k$$

Simplify this, we get

$$n(n-1)(C_n - C_{n-1}) = n(n-1) + 2n(C_{n-1}) - 2\sum_{k=0}^{n-1} C_k$$

Since we know that

$$C_n = n + 1 + \frac{2}{n}\sum_{k=0}^{n-1} C_k$$
$$\Rightarrow 2\sum_{k=0}^{n-1} C_k = n(C_n - n - 1)$$

we could write the right hand side into

$$n(n-1)(C_n - C_{n-1}) = n(n-1) + 2n(C_{n-1}) - n(C_n - n - 1)$$

Finally we get a relationship between $C_n$ and $C_{n-1}$

$$C_n = 2 + \frac{n+1}{n}C_{n-1}$$

Using the above relationship, we iteratively write down the recurrence relations, we denote the time we do the iteration as k

$$\begin{aligned}
C_n &= 2 + \frac{n+1}{n}C_{n-1} \\
&= 2 + \frac{n+1}{n}(2 + \frac{n}{n-1}C_{n-2}) \\
&\dots \\
&= 2 + 2(n+1)(\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{n-(k-2)}) + \frac{n+1}{n-(k-1)}C_{n-k}
\end{aligned}$$

When we reach the base case, $C_{n-k} = C_0$

$$\begin{aligned}
k &= n \\
C_n &= 2 + 2(n+1)\sum_{i=2}^{n} \cdot \frac{1}{i} \\
&= 2(n+1)\frac{1}{n+1} + 2(n+1)\sum_{i=2}^{n} \frac{1}{i} \\
&= 2(n+1)\sum_{i=2}^{n+1} \frac{1}{i}
\end{aligned}$$

Now, we have found the explicit formula for $C_n$.

## Problem 3: Fibonacci-3 [10 marks]

Consider the recurrence relation $F_{n+3} = F_{n+2} + F_{n+1} + F_n$, with the initial state $F_0 = 0, F_1 = 0, F_2 = 1$.

(a) Prove that

$$\begin{pmatrix} F_n \\ F_{n+1} \\ F_{n+2} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}^n \cdot \begin{pmatrix} F_0 \\ F_1 \\ F_2 \end{pmatrix}.$$

(b) So, in order to compute $F_n$, it suffices to raise this $3 \times 3$ matrix, called $X$, to the $n$th power. Show that $O(\log n)$ matrix multiplications suffice for computing $X^n$.

*Solution.* **Please write down your solution to Problem 3 here.**

(a) We use prove by induction

**Base Case**: When $n = 1$,

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}^1 \cdot \begin{pmatrix} F_0 \\ F_1 \\ F-2 \end{pmatrix} = \begin{pmatrix} 0 \cdot F_0 + 1 \cdot F_1 + 0 \cdot F_2 \\ 0 \cdot F_0 + 0 \cdot F_1 + 1 \cdot F_2 \\ 1 \cdot F_0 + 1 \cdot F_1 + 1 \cdot F_2 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ F_0 + F_1 + F_2 \end{pmatrix}$$

According to the recurrence relation $F_{n+3} = F_{n+2} + F_{n+1} + F_n$,

$$\begin{pmatrix} F_1 \\ F_2 \\ F_0 + F_1 + F_2 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ F_3 \end{pmatrix}$$

**Inductive Case**: we assume that our property is true for $n-1$, we would like to show it's also holds true for $n$.

Knowing that

$$\begin{pmatrix} F_{n-1} \\ F_n \\ F_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}^{n-1} \cdot \begin{pmatrix} F_0 \\ F_1 \\ F_2 \end{pmatrix}$$

We could have

$$\begin{pmatrix} F_{n-1} \\ F_n \\ Fn+1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} F_n \\ F_{n+1} \\ F_{n-1} + F_n + F_{n+1} \end{pmatrix} = \begin{pmatrix} F_n \\ F_{n+1} \\ F_{n+2} \end{pmatrix}$$

We have shown that our property is true by using induction.

(b) To compute $X^n$, we could compute $(X^{n/2})^2$,

we write the time function of computing $X^n$ as $T(n)$,and we have

$$T(n) = T(\frac{n}{2}) + 1$$

The 1 represents the step of powering $T(n/2)$, as we iteratively substitute the equation for $k$ times, we get

$$T(n) = T(\frac{n}{2^k}) + k$$

When we reach the base case, that is when

$$\frac{n}{2^k} = 1$$
$$\Rightarrow k = \log n$$

So we could now write $T(n)$ in the form of

$$T(n) = T(1) + \log n$$

Since $T(1)$ is some constant, this is equal to saying $T(n)$ is $O(\log n)$

## Problem 4: Recurrences in programs [10 marks]

Consider the following two programs:

```
int F(int x) {
  assert(x>=1);

  if (x == 1 || x == 2)
    return 1;
  else
    return 2*F(x-1) - F(x-2);
}

void Hanoi(int disk, int source, int dest, int spare) {
  if (disk == 1) {
    return;
  }
  else {
    Hanoi(disk - 1, source, spare, dest);
    Hanoi(disk - 1, spare, dest, source);
  }
}
```

(a) How many times is the function `F` is called when invoking `F(n)` with $n \geq 1$?

(b) How many times is the function `Hanoi` called when invoking `Hanoi(n,0,0,0)` with $n \geq 1$?

*Solution.* **Please write your solution to Problem 4 here.**

(a) We write the recurrence relation in this program as

$$T(n) = T(n-1) + T(n-2) + 2$$

the 2 represents the operation cost each time we branching, including the multiplication and the subtraction.
We start by guessing that $T(n) \le k \cdot 2^n$, where $k \ge 0$ and we need to prove this by induction
**Base case**: $T(1) \le k \cdot 2^1 = 2k$, which holds true since $T(1)$ is a constant.
**Inductive case**: We assume our property to be true for $n-1$ and $n-2$, So we have

$$\begin{aligned} T(n-1) &\le k \cdot 2^{n-1} \\ T(n-2) &\le k \cdot 2^{n-2} \end{aligned}$$

So, we have

$$\begin{aligned} T(n) &\le k \cdot 2^{n-1} + k \cdot 2^{n-2} + 2 \\ &= 3k \cdot 2^{n-2} + 2 \\ &\le k \cdot 2^n \end{aligned}$$

To satisfy the last relationship,

$$\begin{aligned} 3k \cdot 2^{n-2} + 2 &\le k \cdot 2^n \\ \Rightarrow 3k \cdot 2^{n-2} + 2 &\le 4k \cdot 2^{n-2} \\ \Rightarrow 2 &\le k \cdot 2^{n-2} \\ \Rightarrow 2^{n-2} &\ge \frac{2}{k} \\ \Rightarrow n-2 &\ge \log \frac{2}{k} \\ \Rightarrow n &\ge 3 - \log k \end{aligned}$$

So, our property holds true for any $n \ge 3 - \log k$. To finish with our proof, we pick $k = 2$, $T(n) \le 2 \cdot 2^n$ for $n \ge 2$.
$T(n)$ has an asymptotically upper bound of $O(2^n)$.

(b) We write the recurrence relation in this program as

$$T(n) = 2T(n-1)$$

We start by guessing that $T(n) \le k \cdot 2^n$, for some constant $k \ge 0$.
**Base Case**: $T(1) \le 2k$, which is true since $T(1)$ is some constant.
**Inductive Case**: We assume our property true for $n-1$. We would like to show that it holds true for $n$.

$$\begin{aligned} T(n) &\le 2k \cdot 2^{n-1} \\ &= k \cdot 2^n \end{aligned}$$

We have shown that $T(n) \le k \cdot 2^n$, for any constant $k \ge 0$. We pick $k = 1$.
$T(n)$ has an asymptotically upper bound of $O(2^n)$.

## Problem 5: Summations [10 marks + 10 marks]

(a) (10 marks) Let us be given three sequences of integers, say $A, B$, and $C$, each of length $n$. Devise an algorithm to find whether there are three numbers $a \in A$ and $b \in B$ and $c \in C$ such that $a + b + c = 0$. You will get full marks if the algorithm is of $O(n^2)$ complexity and it is proven to be correct.

(b) **(Bonus: 10 marks)** Let us be given four sequences of integers, say $A, B, C$, and $D$, each of length $n$. Devise an algorithm to find whether there are four numbers $a \in A$ and $b \in B$ and $c \in C$ and $d \in D$ such that $a + b + c + d = 0$. You will get full marks if the algorithm is of $O(n^2 \log n)$ complexity and it is proven to be correct.

*Solution.* **Please write down your solution to Problem 5 here.**

(a)
```
summation3(A,B,C){
    sort(B);
    sort(C);
    ptr1 = 0;
    ptr2= len(B)-1;
    for (a in A):{
        while(ptr1<len(B)-1 and ptr2>0){
            2sum = B[ptr1]+C[ptr2];
            if(2sum > -a){
                ptr2 -= 1;
            }else if(2sum < -a){
                ptr2 ++;
            }else{
                return True;
            }
        }
    }
    return False;

}
```

**Explanation**: Since our goal is to find one number in each sequences such that $a+b+c=0$, the problem is equal to knowing $a$, find $b + c = -a$, b belongs to sequence B, c belongs to sequence C.
We sort the two sequence $B$ and $C$ first, which takes $O(n\dot{l}ogn)$ time.
Then, inside the for loop of sequence $A$, we search for two number in each sequence. We use two pointers to do the search synchronously, with ptr1 starts at the head of B, ptr2 starts at the tail of C. When the sum of the pointed 2 integers less or equal to $-a$, we increase ptr1 by 1; when the sum of the two integers greater or equal to $-a$, we decrease prt2 by 1. This process, which would be at most $2n$ operations, takes O(n) time.
Thus, the time complexity of our summation algorithm is $n \cdot n$, which is $O(n^2)$.

(b)
```
summation4(A,B,C,D){}
```