CS5800: Algorithms — Spring '21 — Virgil Pavlu

Homework 1
Due : Wednesday, January 27 at 11:59pm via Gradescope

Name: Shri Datta Madhira (NUID: 001557772)

Instructions:

- Make sure to put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- Please review the grading policy outlined in the course information page.
- You must also write down with whom you worked on the assignment. If this changes from problem to problem, then you should write down this information separately with each problem.
- Problem numbers (like Exercise 3.1-1) are corresponding to CLRS 3^{rd} edition. While the 2^{nd} edition has similar problems with similar numbers, the actual exercises and their solutions are different, so make sure you are using the 3^{rd} edition.

1. ***(20 points)***

Two linked lists (simple link, not double link) heads are given:headA, and headB;it is also given that the two lists intersect, thus after the intersection they have the same elements to the end. Find the first common element, without modifying the lists elements or using additional datastructures.

(a) A linear algorithm is discussed in the lecture: count the lists first, then use the count difference as an offset in the longer list, before traversing the lists together. Write a formal pseudocode (the pseudocode in the lecture is vague), using "next" as a method/pointer to advance to the next element in a list.
Solution:

Algorithm 1: Algorithm to find the first common element

```
Function findIntersection(Node* headA, Node* headB):
    Node* ptr1 = headA, Node* ptr2 = headB, m = 0, n = 0
    While ptr1 != NULL :
        m++
        ptr1 = ptr1 → next
    While ptr2 != NULL :
        n++
        ptr2 = ptr2 → next
    If m > n :
        ptr1 = headA, ptr2 = headB
    ElseIf n > m :
        ptr1 = headB, ptr2 = headA
    For i ← 0 to abs(m − n)
        ptr1 = ptr1 → next
    While ptr1 and ptr2 != NULL :
        If ptr1 == ptr2 :
            Return ptr1 → data
        ptr1 = ptr1 → next
        ptr2 = ptr2 → next
```

(b) Write the actual code in a programming language (C/C++, Java, Python etc) of your choice and run it on a made-up test pair of two lists. A good idea is to use pointers to represent the list linkage.
Solution:

```
#include <iostream>
#include <cmath>
using namespace std;

class Node {
public:
```

```
    int data;
    Node* next;
};

int intersect(Node* currA, Node* currB)
{
    while(currA != NULL && currB != NULL)
    {
        if(currA == currB)
            return currA->data;
        currA = currA->next;
        currB = currB->next;
    }
    return -1;
}

int count(Node* headA, Node* headB)
{
    Node* currA = headA;
    Node* currB = headB;
    int m = 0, n = 0;

    while(currA != NULL)
    {
        m++;
        currA = currA->next;
    }
    while(currB != NULL)
    {
        n++;
        currB = currB->next;
    }

    if(m >= n)
    {
        currA = headA, currB = headB;
    }
    else if(m < n)
    {
        currA = headB, currB = headA;
    }
    for(int i = 0; i < abs(m-n); i++)
    {
        if(currA == NULL) return -1;
        currA = currA->next;
```

```
    }

    return intersect(currA, currB);
}

int main()
{

    Node* headA = new Node();
    headA->data = 1;

    Node* headB = new Node();
    headB->data = 0;

    Node* temp = new Node();
    temp->data = 2;
    headB->next = temp;

    temp = new Node();
    temp->data = 3;
    headA->next = temp;

    temp = new Node();
    temp->data = 5;
    headA->next->next = temp;

    temp = new Node();
    temp->data = 4;
    headB->next->next = temp;

    temp = new Node();
    temp->data = 7;
    headA->next->next->next = temp;

    temp = new Node();
    temp->data = 6;
    headB->next->next->next = temp;
    headB->next->next->next->next =  headA->next->next->next;

    temp = new Node();
    temp->data = 9;
    headA->next->next->next->next = temp;

    temp = new Node();
    temp->data = 9;
```

```
        headA->next->next->next->next->next = temp;

        if(count(headA, headB) == -1)
            cout << "Something is wrong";
        else
            cout << "The intersection point is " << count(headA, headB)
                ;
        return 1;
    }
```

2. ***(10 points)*** *Exercise 3.1-1*

Solution:

$max(f(n), g(n)) = \Theta(f(n) + g(n)) \quad \Rightarrow c_1 * (f(n) + g(n)) \leq max(f(n), g(n)) \leq c_2 * (f(n) + g(n))$

Intuitively thinking, f(n) and g(n) individually, will always be less than max(f(n), g(n)). We can rewrite this mathematically as,

$$f(n) \leq max(f(n), g(n)) \text{ and } g(n) \leq max(f(n), g(n))$$

Adding the above inequalities, we get,

$$f(n) + g(n) \leq 2(max(f(n), g(n))) \quad \Rightarrow 1/2(f(n)+g(n)) \leq max(f(n), g(n)) \quad \Rightarrow c_1 = 1/2(or)0.5$$

Coming to the second inequality, $max(f(n), g(n)) \leq c_2 * (f(n) + g(n)) \cdots (a)$

As, f(n) and g(n) are asymptotically non-negative functions,

$$f(n) \geq 0,$$
$$g(n) \geq 0$$

That implies, $max(f(n), g(n) \geq 0 \cdots (b)$

Plugging in $c_2 = 0$, in equation (a), we get $max(f(n), g(n)) \leq 0$, which is not possible according to equation (b).

Plugging in $c_2 = 1$ satisfies the asymptotic condition and the inequality.

Hence, for $c_1 = 0.5$ and $c_2 = 1$ we have proved that $max(f(n), g(n)) = \Theta(f(n) + g(n))$

3. ***(5 points)*** *Exercise 3.1-4*

Solution:

(i) $2^{n+1} = O(2^n)$ - True.

$2^{n+1} = 2 * 2^n$

Ignoring '2' as a constant, we will get the asymptotic notation as $O(2^n)$

(ii) $2^{2n} = O(2^n)$ - False.

$2^{2n} = (2^2)^n = 4^n$

4^n is not equal to $O(2^n)$ because 2^n and 2^{2n} are two different terms.

4. ***(15 points)***

Rank the following functions in terms of asymptotic growth. In other words, find an arrangement of the functions $f_1, f_2, \ldots$ such that for all i, $f_i = \Omega(f_{i+1})$.

$$\sqrt{n}\ln n \quad \ln\ln n^2 \quad 2^{\ln^2 n} \quad n! \quad n^{0.001} \quad 2^{2\ln n} \quad (\ln n)!$$

Solution: **Ordered from Fastest to Slowest**

$$n^{0.001} \quad \ln\ln n^2 \quad \sqrt{n}\ln n \quad 2^{2\ln n} \quad (\ln n)! \quad 2^{\ln^2 n} \quad n!$$

Looking at the terms and basing on the order of growth table discussed in class, we can see that,

1. $\boldsymbol{n^{0.001}}$ is the fastest of the bunch because it's power is thousandth of 1.
2. $\boldsymbol{n!}$ is the slowest of them all. It is thousands of centuries slower than 2^n itself. So, definitely the slowest of the list.
3. $\boldsymbol{\ln\ln n^2}$ is second fastest. It is slower than $n^{0.001}$ because of the n^2 term. The nested logarithms is the reason why this term is faster than most others in the list.
4. $\boldsymbol{\sqrt{n}\ln n}$ is slower than $\ln\ln n^2$ because of the two logarithms in the previous term. Log terms are the fastest overall and that is the reason why $\sqrt{n}\ln n$ is the third fastest.
5. $\boldsymbol{2^{2\ln n}}$ is the fourth fastest because of the logarithm. 2^n is usually the second slowest run time we can get. Though it is so much slower than the terms before this, the logarithm in the power reduces the impact of the base.
6. $\boldsymbol{(\ln n)!}$ is the fifth fastest or third slowest on the list. Although logarithm is the fastest achievable run time, the factorial slows it down massively because of how slow it normally is.
7. $\boldsymbol{2^{\ln^2 n}}$ is the second slowest of the list because this is the only term remaining.

We can also write a simple python code using *matplotlib* and *numpy* to see how the terms behave.

```
import matplotlib.pyplot as plt
import numpy as np
import math
from scipy.special import gamma

n = np.linspace(1, 10000000)

plt.plot(n, np.sqrt(n)*np.log(n), label = 'first term')
plt.plot(n, np.log(np.log(n**2)), label = 'second term')
plt.plot(n, 2**(np.log(n)**2), label = 'third term')
plt.plot(n, gamma(n), label = 'fourth term')
plt.plot(n, n**0.001, label = 'fifth term')
plt.plot(n, 2**(2*np.log(n)), label = 'sixth term')
plt.plot(n, gamma(np.log(n)), label = 'seventh term')
```

5. ***(40 points)*** *Problem 4-1 (page 107)*

Give asymptotic upper and lower bounds for T(n) in each of the following recurrences. Assume that T(n) is constant for $n \leq 2$. Make your bounds as tight as possible, and justify your answers.

(a) $T(n) = 2T(n/2) + n^4$
Solution: Using Master Theorem, a = 2, b = 2, and c = 4. From this we can see that $a/b^c < 1$.
So, the answer will $\boldsymbol{\Theta(n^4)}$.

(b) $T(n) = T(7n/10) + n$
Solution: Using Master Theorem, a = 1, b = 10/7, and c = 4. From this we can see that $a/b^c < 1$.
So, the answer will $\boldsymbol{\Theta(n)}$.

(c) $T(n) = 16T(n/4) + n^2$
Solution: Using Master Theorem, a = 16, b = 4, and c = 2. From this we can see that $a/b^c = 1$.
So, the answer will $\boldsymbol{\Theta(n^2 \lg n)}$.

(d) $T(n) = 7T(n/3) + n^2$
Solution: Using Master Theorem, a = 7, b = 3, and c = 2. From this we can see that $a/b^c < 1$.
So, the answer will $\boldsymbol{\Theta(n^2)}$.

(e) $T(n) = 7T(n/2) + n^2$
Solution: Using Master Theorem, a = 7, b = 2, and c = 2. From this we can see that $a/b^c > 1$.
So, the answer will $\boldsymbol{\Theta(n^{\lg 7})}$.

(f) $T(n) = 2T(n/4) + \sqrt{n}$
Solution: Using Master Theorem, a = 2, b = 4, and c = 0.5. From this we can see that $a/b^c = 1$.
So, the answer will $\boldsymbol{\Theta(\sqrt{n} \lg n)}$.

(g) $T(n) = T(n-2) + n^2$
Solution:

$$\begin{aligned} T(n) &= T(n-2) + n^2 \\ \Rightarrow T(n) &= T(n-4) + (n-2)^2 + n^2 \\ \Rightarrow T(n) &= T(n-6) + (n-4)^2 + (n-2)^2 + n^2 \\ \Rightarrow T(n) &= T(n-8) + (n-6)^2 + (n-4)^2 + (n-2)^2 + n^2 \end{aligned}$$

The pattern of the recurrence relation would look something like,

$$\Rightarrow T(n) = T(n-2k) + (n-2k+2)^2 + (n-2k+4)^2 + \cdots + (n-2)^2 + n^2$$

Let k = n/2, we will have,

$$\begin{aligned} \Rightarrow T(n) &= T(0) + (2^2 + 4^2 + 6^2 + \cdots + n^2) \\ \Rightarrow T(n) &= T(0) + 4[\tfrac{(\frac{n}{2})(\frac{n}{2}+1)(\frac{2n}{2}+1)}{6}] \\ \Rightarrow T(n) &= T(0) + \tfrac{n(n+1)(n+2)}{6} \end{aligned}$$

The leading term on the right hand side in the above equation is n^3.
Therefore, the answer is $\boldsymbol{\Theta(n^3)}$

6. ***(30 points)*** *Problem 4-3 from (a) to (f) (page 108)*

Give asymptotic upper and lower bounds for T(n) in each of the following recurrences. Assume that T(n) is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers.

(a) $T(n) = 4T(n/3) + n\lg n$
Solution:

$$\begin{aligned} T(n) &= 4T(n/3) + n\lg n \\ \Rightarrow 4[4T(n/9) + \tfrac{n}{3}\lg(n/3)] + n\lg n &= 16T(n/9) + \tfrac{4n}{3}\lg(n/3) + n\lg n \\ \Rightarrow 16[4T(n/27) + \tfrac{n}{9}\lg(n/9)] + \tfrac{4n}{3}\lg(n/3) + n\lg n &= 64T(n/27) + \tfrac{16n}{9}\lg(n/9) + \tfrac{4n}{3}\lg(n/3) + n\lg n \end{aligned}$$

This can be generalised as, $4^i T(n/3^i) + n\sum_{j=0}^{i-1}(\frac{4}{3})^j \lg(n/3^j)\cdots(1)$
$\frac{n}{(3^i)} \simeq 1 \Rightarrow n \simeq 3^i \Rightarrow i = \lg_3 n$. Substituting this in equation (1), we get,

$$\begin{aligned} &4^{\lg_3 n} T(n/3^{\lg_3 n}) + n\sum_{j=0}^{\lg_3 n-1}(\tfrac{4}{3})^j \lg(n/3^j) \\ &= n^{\lg_3 4} T(1) + n\sum_{j=0}^{\lg_3 n-1}(\tfrac{4}{3})^j \lg(n/3^j)\cdots(2) \end{aligned}$$

Expanding the summation,

$$\begin{aligned} &(\tfrac{4}{3})^0 \lg n + (\tfrac{4}{3})^1 \lg(n/3) + \cdots + (\tfrac{4}{3})^{\lg_3 n-1} \lg(n/3^{\lg_3 n-1}) \\ &= \lg n + \tfrac{4}{3}\lg(n/3) + \cdots + \tfrac{4^{\lg_3 n}.4^{-1}}{3^{\lg_3 n}.3^{-1}} \lg 3 \\ &= \lg n + \tfrac{4}{3}\lg(n/3) + \cdots + \tfrac{3}{4}.\tfrac{n^{\lg_3 4}}{n} \lg 3 \\ &= \lg n + \tfrac{4}{3}\lg(n/3) + \cdots + \tfrac{3}{4}.n^{\lg_3 4 - 1} \lg 3 \\ &\Rightarrow T(n) = \Theta(n^{\lg_3 4} + \tfrac{3}{4}.n^{\lg_3 4 - 1} \lg 3) \end{aligned}$$

From this we can see that T(n) is asymptotically bound by $\boldsymbol{\Theta(n^{\lg_3 4})}$

(b) $T(n) = 3T(n/3) + n/\lg n$
Solution:

$$\begin{aligned} T(n) &= 3T(n/3) + n/\lg n \\ \Rightarrow 3(3T(n/9) + (n/3)/\lg(n/3)) + n/\lg n \\ \Rightarrow 9T(n/9) + n/\lg(n/3) + n/\lg n \end{aligned}$$

This can be generalised as, $3^i T(n/3^i) + n\sum_{j=1}^{i} 1/\lg(n/3^{j-1})$.
$\frac{n}{(3^i)} \simeq 1 \Rightarrow n \simeq 3^i \Rightarrow i = \lg_3 n$. Substituting this in the above equation, we get,

$$nT(1) + n \sum_{j=1}^{\log_3 n} 1/\lg(n/3^{j-1})$$
$$\Rightarrow T(n) + n \sum_{j=1}^{\log_3 n} 1/(\lg(n) - (j-1)\lg 3)$$
$$\Rightarrow T(n) + (n/\lg 3) \sum_{j=1}^{\log_3 n} [1/\log_3 n - (j-1)]$$

In the above equation, $\sum_{j=1}^{\log_3 n} [1/\log_3 n - (j-1)]$ expands to a Harmonic sequence when $\lg_3 n$ is assumed as some variable X. We get $\lg X \Rightarrow \lg\lg n$.

So, from that we can say, $T(n) = 3T(n/3) + n/\lg n$ is asymptotically bound by $\boldsymbol{\Theta(n \lg\lg n)}$.

(c) $T(n) = 4T(n/2) + n^2\sqrt{n}$
Solution:

Using Master Theorem, a = 4, b = 2, and c = 2.5. From this we can see that $a/b^c < 1$. So, the answer will $\boldsymbol{\Theta(n^2\sqrt{n})}$.

(d) $T(n) = 3T(n/3 - 2) + n/2$
Solution:

$$T(n) = 3T(n/3 - 2) + n/2 \;=\; 3T(\tfrac{n-6}{3}) + \tfrac{n}{2}$$
$$\Rightarrow T(n) = 3[3T(\tfrac{\frac{n-6}{3}-6}{3}) + \tfrac{\frac{n-6}{3}}{2}] + \tfrac{n}{2} \Rightarrow T(n) = 9T(\tfrac{n-24}{9}) + \tfrac{n-6}{2} + \tfrac{n}{2}$$
$$\Rightarrow T(n) = 9[3T(\tfrac{\frac{n-24}{9}-6}{3}) + \tfrac{\frac{n-24}{9}}{2}] + \tfrac{n-6}{2} + \tfrac{n}{2} \Rightarrow T(n) = 27T(\tfrac{n-78}{27}) + \tfrac{n-24}{2} + \tfrac{n-6}{2} + \tfrac{n}{2}$$

Here, if we substitute n = 1000, the reduction in T(n) mostly comes from the division in $T(\frac{n-78}{27})$ than the subtraction. From this observation, I am making an assumption that asymptotically, subtraction will not contribute to any considerable change for it do be included in the equation. Thus, by excluding the subtraction, we will get,

$$\Rightarrow T(n) = 27T(\tfrac{n}{27}) + \tfrac{n}{2} + \tfrac{n}{2} + \tfrac{n}{2}$$

The above equation can now be generalised as follows,

$$\Rightarrow T(n) = 3^i T(\tfrac{n}{3^i}) + \sum_{j=1} i\tfrac{n}{2} \cdots (1)$$

Now, $\frac{n}{(3^i)} \simeq 1 \Rightarrow n \simeq 3^i \Rightarrow i = \lg_3 n$. Substituting this in equation (1),

$$\Rightarrow T(n) = nT(1) + (\tfrac{n}{2} + \tfrac{2n}{2} + \tfrac{3n}{2} + \cdots + \tfrac{n\lg_3 n}{2})$$
$$\Rightarrow T(n) = \Theta(n + \tfrac{n\lg_3 n}{2})$$

Therefore, we can say that $T(n) = 3T(n/3 - 2) + n/2$ is asymptotically bound by $\boldsymbol{\Theta(n \lg_3 n)}$

(e) $T(n) = 2T(n/2) + n/\lg n$

Solution:

$$T(n) = 2T(n/2) + n/\lg n$$
$$\Rightarrow 2(2T(n/4) + (n/2)/\lg(n/2)) + n/\lg n$$
$$\Rightarrow 4T(n/4) + n/\lg(n/2) + n/\lg n$$

This can be generalised as, $2^i T(n/2^i) + n\sum_{j=1}^{i} 1/\lg(n/2^{j-1})$

$\frac{n}{(2^i)} \simeq 1 \Rightarrow n \simeq 2^i \Rightarrow i = \lg n$. Substituting this in the above equation, we get,

$$nT(1) + n\sum_{j=1}^{\lg n} 1/\lg(n/2^{j-1})$$
$$\Rightarrow T(n) + n\sum_{j=1}^{\lg n} 1/(\lg(n) - (j-1)\lg 2)$$
$$\Rightarrow T(n) + n\sum_{j=1}^{\lg n} [1/\lg n - (j-1)]$$

In the above equation, $\sum_{j=1}^{\lg n} [1/\lg n - (j-1)]$ expands to a Harmonic sequence when $\lg n$ is assumed as some variable X. We get $\lg X \Rightarrow \lg\lg n$.

. So, from that we can say, $T(n) = 2T(n/2) + n/\lg n$ is asymptotically bound by $\boldsymbol{\Theta(n \lg\lg n)}$.

(f) $T(n) = T(n/2) + T(n/4) + T(n/8) + n$

Solution:

Guessing $T(n) = T(n/2) + T(n/4) + T(n/8) + n$ is bound by $\Theta(n)$.

$$\Rightarrow c_1 n \leq T(n) \leq c_2 n \cdots\cdots (1)$$

Solving $T(n) \leq c_2 n$,

$$\Rightarrow c_2(\tfrac{n}{2}) + c_2(\tfrac{n}{4}) + c_2(\tfrac{n}{8}) + n \leq c_2 n$$
$$\Rightarrow c_2 n(\tfrac{4+2+1}{8}) + n \leq c_2 n$$
$$\Rightarrow \tfrac{7c_2 n}{8} + n \leq c_2 n$$
$$\Rightarrow n(\tfrac{7c_2}{8} + 1) \leq c_2 n$$

For $c_2 = 8$, we get $8n \leq 8n$, which satisfies the condition. Therefore, we proved that $T(n) \leq c_2 n$.
Next, solving $T(n) \geq c_1 n$,

$$\Rightarrow n(\tfrac{7c_1}{8} + 1) \geq c_1 n$$

For $c_1 = 1$, we get $\frac{15}{8} n (or) 1.8n \geq n$, which satisfies the condition. Therefore, it is proved that $T(n) \geq c_1 n$.
Since the equation (1) is proved, we can say that T(n) is asymptotically bound by $\boldsymbol{\Theta(n)}$.