CS5800: Algorithms — Virgil Pavlu

Homework 1

Name: Naren Mohan

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 head B;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 Intersection of two linked lists

Ensure: head1 is a linked list
Ensure: head2 is a linked list
  m = 0
  n = 0
  head1x = head1
  head2x = head2
  while head1 or head2 is not null **do**
    if head1 is not null **then**
      m = m + 1
    end if
    if head2 is not null **then**
      n = n + 1
    end if
  end while
  if m greater than or equal to n **then**
    head1 = head1x
    head2 = head2x
  else
    head1 = head2x
    head2 = head1x
  end if
  offset = abs(m - n)
  for i in 1 to offset **do**
    head1 = head1.next
  end for
  while head1 is not null **do**
    if head1 == head2 **then**
      return head1
    end if
    head1 = head1.next
    head2 = head2.next
  end while
  return ”No intersection found”
  =0

(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:

```
class Node:
    def __init__(self, val=None):
```

```
        self.data = val
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def printlist(self):
        printval = self.head
        while printval is not None:
            print(printval.data)
            printval = printval.next

head1 = LinkedList()
head2 = LinkedList()

node_a = Node("a")
node_b = Node("b")
node_c = Node("c")
node_d = Node("d")
node_e = Node("e")
node_f = Node("f")
node_1 = Node("1")
node_2 = Node("2")

head1.head = node_a
node_a.next = node_b
node_b.next = node_c
node_c.next = node_d
node_d.next = node_e
node_e.next = node_f

head2.head = node_1
node_1.next = node_2
node_2.next = node_d  # Intersecting node

print("List 1")
head1.printlist()

print("\n\nList 2")
head2.printlist()

def intersectFinder(h1, h2):
    m = n = 0
```

```
    h1_x = h1
    h2_x = h2

    # Count the length of both the lists
    while not(h1 == None and h2 == None):
        if h1 != None:
            m += 1
            h1 = h1.next

        if h2 != None:
            n += 1
            h2 = h2.next
    print("Length of List 1: {} \tList 2: {}".format(m, n))


    # Offsetting the list
    if m >=n:
        h1, h2 = h1_x, h2_x
    else:
        h1, h2 = h2_x, h1_x
    offset = abs(m - n)
    for i in range(offset):
        h1 = h1.next

    # Return the intersecting node
    while h1 != None:
        if h1 == h2:
            return h1
        h1 = h1.next
        h2 = h2.next
    return None

res_node = intersectFinder(head1.head, head2.head)

if res_node:
    print(res_node.data)
else:
    print("No intersection found!")
```

Solution can be found in the attached ”ques1.py” Python file.

2. (10 points) Exercise 3.1-1

Solution:
To prove
$max(f(n), g(n)) = \Theta(f(n) + g(n))$

Given f, g are asymptotically non-negative functions.
Then for $n \geq n_0$,
$f(n) + g(n) \geq f(n) \geq 0$ and
$f(n) + g(n) \geq g(n) \geq 0$

$max(f(n), g(n)) \leq f(n)$
$max(f(n), g(n)) \leq g(n)$
Therefore,
$max(f(n), g(n)) \leq c_1(f(n) + g(n))$
As per the definition of Θ,
$max(f(n), g(n)) = \Theta(f(n) + g(n)$ with $c = 1$
Hence proved.

3. (5 points) Exercise 3.1-4

Solution:
Is $2^{n+1} = O(2^n)$?
$2^{n+1} = 2.2^n$ for all $n \geq 0$ and $c = 2$.
Hence $2^{n+1} = O(2^n)$

Is $2^{2n} = O(2^n)$?
$2^{2n} = 2^n.2^n \leq c.2^n$ with $c = 2^n$
But 2^n cannot be lesser than c
Hence $2^{2n} \neq O(2^n)$

4. (15 points)

Rank the following functions in terms of asymptotic growth. In other words, find an arrangement of the functions $f_1, f_2, ...$ 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:
It is ordered in the following way - $2^{\ln^2 n} \quad 2^{2\ln n} \quad n! \quad (\ln n)! \quad \sqrt{n}\ln n \quad \ln\ln n^2 \quad n^{0.001}$

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.

Using the following equation and the cases in Master's theorem as the base to solve the following sub-question - $T(n) = aT(n/b) + f(n)$

(a) $T(n) = 2T(n/2) + n^4$

Solution:
Here $a = 2$, $b = 2$, $f(n) = n^4$
$log_b a = 1$
$n^{log_b a + \epsilon} = n^{1+3} = f(n)$
Case 3 is to be applied -
$T(n) = \Theta(n^4)$

(b) $T(n) = T(7n/10) + n$

Solution:
Here $a = 1$, $b = 10/7$, $f(n) = n$
$log_b a = 0$
$n^{log_b a + \epsilon} = n^{0+1} = f(n)$
Case 3 is to be applied -
$T(n) = \Theta(n)$

(c) $T(n) = 16T(n/4) + n^2$

Solution:
Here $a = 16$, $b = 4$, $f(n) = n^2$
$log_b a = 2$
$n^{log_b a} = n^2 = f(n)$
Case 2 is to be applied -
$T(n) = \Theta(n^2 lg(n))$

(d) $T(n) = 7T(n/3) + n^2$

Solution:
Here $a = 7$, $b = 3$, $f(n) = n^2$
$log_b a = 1.77$
$n^{log_b a + \epsilon} = n^{1.77+0.23} = f(n)$
Case 3 is to be applied -
$T(n) = \Theta(n^2)$

(e) $T(n) = 7T(n/2) + n^2$

Solution:
Here $a = 7$, $b = 2$, $f(n) = n^2$
$log_b a = 2.81$
$n^{log_b a - \epsilon} = n^{2.81-0.81} = f(n)$
Case 1 is to be applied -
$T(n) = \Theta(n^{2.81})$

(f) $T(n) = 2T(n/4) + \sqrt{n}$

Solution:
Here $a = 2$, $b = 4$, $f(n) = n^{1/2}$

$log_b a = 1/2$
$n^{log_b a} = n^{1/2} = f(n)$
Case 2 is to be applied -
$T(n) = \Theta(n^{1/2} * lg(n))$

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

Solution:
Here Master theorem cannot be applied as the recurrence is not in its form.
Expanding the form -

$T(n) = T(n-2) + n^2$
$T(n) = T(n-4) + (n-2)^2 + n^2$
$T(n) = T(n-6) + (n-4)^2 + (n-2)^2 + n^2$
$T(n) = T(0) + 2^2 + 4^2 + ... + (n-4)^2 + (n-2)^2 + n^2$
$T(n) = T(0) + \sum_{i=0}^{n/2} (n-2i)^2$
$T(n) = T(0) + \sum_{i=0}^{n/2} n^2 - \sum_{i=0}^{n/2} 4ni + \sum_{i=0}^{n/2} 4i^2$
$T(n) = T(0) + \Theta(n^3) - \Theta(n^3) + \Theta(n^3)$
$T(n) = \Theta(n^3)$

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.

Using the following equation and the cases in Master's theorem as the base to solve the following sub-question - $T(n) = aT(n/b) + f(n)$

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

Solution:
Here $a = 4$, $b = 3$, $f(n) = n \lg n$
$log_b a = 1.26$
$n^{log_b a - \epsilon} = f(n)$
Case 1 is to be applied -
$T(n) = \Theta(n^{log_3 4})$

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

Solution:
By substitution -
$T(n) = 3T(n/3) + n/\lg n$
$T(n) \leq cn \lg n - cn \lg 3 + n/\lg n$
$= cn \lg n + n(\frac{1}{\lg n} - c \lg 3) \leq cn \lg n$
$T(n) \leq n \lg n$

$T(n) = 3T(n/3) + n/\lg n$

$T(n) \geq \frac{3c}{3^{1-\epsilon}n^{1-\epsilon}} + n/\lg n$
$= 3^{\epsilon}cn^{1-\epsilon} + n/\lg n$
$T(n) \geq cn^{1-\epsilon}$

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

Solution:
Here $a = 4$, $b = 2$, $f(n) = n^2\sqrt{n}$
$log_b a = 2$
$\Theta(n^{log_b a}) = f(n)$
Case 2 is to be applied -
$T(n) = \Theta(n^{5/2})$

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

Solution:
Ignoring the subtraction inside the argument as for large n it wouldn't matter
Then here $a = 3$, $b = 3$, $f(n) = n/2$
$log_b a = 1$
$\Theta(n^1) = f(n)$
Case 2 is to be applied -
$T(n) = \Theta(n \lg n)$

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

Solution:
Similar to part b, the function is $O(n \lg n)$ and $\Omega(n^{1-\epsilon})$ for all ϵ

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

Solution:
$T(n) = T(n/2) + T(n/4) + T(n/8) + n$
$T(n) \leq \frac{7}{8}cn + n \leq cn$ when $c \geq 8$