Algorithms Assignment 1

Ketaki Nitin Kolhatkar

September 2021

Collaborator : Madhusudan Malhar Deshpande

1 Answers

Ans 1. (a)

```
1. Make a class to initialize the nodes - data and their addresses.
2. Make a class - 'List' with functions for printing the list, getting the length of the
3. def length():
    total = 0
    while the last element is not null, i.e. the length of the list:
        iterate total by one
        go to the next node now
        return the total
4. Now find the offset to make calculations easier - for lesser number of comparisons.
Intersection:
if the length of a > length of b:
    offset = length(a) - length(b)
    Extend the list a head by the offset found
if the length of a < length of b:
    offset = length(b) - length(a)
    Extend the list b head by the offset found
5. Finding the intersection:
    After finding the offset we compare the elements of list a to list b.
    while list a and list b exist:
        if the address at a == address at b
        return value
```

(b)

```
    class Node:
  def __init__(self, data): #to initialize the nodes
    self.data = data
    self.next = None
```

```
class List:
  def __init__ (self,head):
    self.head = head

  def printlist(self):
    value = self.head
    while value != None:
      print(value.data)
      value = value.next

  def length(self):
    cur = self.head
    total = 0
    while cur != None:
        total += 1
        cur = cur.next
    return total

def inter(head1, head2):
    a = head1.length() #list length
    b = head2.length() #list length

    cur1 = head1.head
    cur2 = head2.head

    if a > b:
        offset = a - b #length()
        for i in range(offset):
            cur1 = cur1.next
    elif b > a:
        offset = b - a
        for i in range(offset):
            cur2 = cur2.next

    while cur1 != None and cur2 != None:
        if cur1 == cur2:
            return cur1.data
        cur1 = cur1.next #node assignment
        cur2 = cur2.next #node assignment
        print("The intersection : ", inter(list1, list2))
```

Ans 2. max f(n), g(n) = θ(f(n) + g(n))

This would mean,

c1$(f(n)+g(n)) \leq max(f(n)+g(n) \leq c2$(f(n)+g(n))

Taking LHS,
$f(n)\leq$ max(f(n), g(n))g(n) $\leq$ max(f(n), g(n))
Then adding the equations we get,
f(n) + g(n) $\leq$ 2[max(f(n), g(n)]
1/2 [f(n) + g(n)] $\leq$ max (f(n), g(n))
c1 $\leq$ 1/2

Now taking RHS,
max $(f(n), g(n)) \leq c2(f(n)+g(n))$

Since we're adding the functions, it will always be greater than the $max(f(n), g(n))$. We can assume c1 = 1

Ans 3. (a) 2^{n+1} = O(2^n)
2^n.2 $\leq$c1.2^n

Hence, c1 = 2

(b) $2^n = O(2^n)$
2^n.2^n $\leq$c1. 2^n
$2^n \leq c1$

Ans 4. First we sort the terms with an approximation of the increasing order of time complexities. Log functions are the slowest functions while exponential have the fastest. Polynomials lie in the middle of these two. Faster than all of them is factorial.
$lnlnn^2$ - log
n^2lnn - polynomial
2^{ln^2n} - exponential
2^{2lnn} - exponential
$(lnn)!$ - factorial
$n!$ - factorial
$n^{0.001}$ - exponential
We would that $lnlnn^2$ would be the lowest and n would be the highest according to this approximation.
Using the L'Hospital Rule we compare all the terms with one another:
First case:
$\lim_{x\to\infty} f(n)/ \lim_{x\to\infty} g(n) = \lim_{x\to\infty} n^{0.001}$ / $\lim_{x\to\infty} 2^{2lnn}$
$\lim_{x\to\infty} f'(n)/ \lim_{x\to\infty} g'(n) = 0.001n^{-0.999}/\ log2.2^{2logn}.(logn+1)$
$= 1/\infty = 0$
Hence, f = o(g), which means f(n) is the upper bound of g(n)

Similarly, we compare all the functions one by one with each other and get the order, with the help of the approximation.

Answer - $ln(lnn) < n^{0.001} < \sqrt{n}lnn < 2^lnn < 2^{2lnn} < n!$

Ans 5. (a)

$$T(n) =$$

2T(n/2) + n^4 (1)

Comparing this to the standard equation of $aT(n/b) + n^c$, we get:
a= 2
b= 2
c=4

We now find - a/b^c for understanding which condition is satisfied in the Master Theorem
Here, $a/b^c = 2/2^4$ which is less than 1.
According to the Master Theorem, if $a/b^c < 1$, then T(n) = θ(n^c)
$Hence, T(n) = \theta(n^4)$

(b)

$$T(n) = T(7n/10) + n \tag{2}$$

Comparing this to $aT(n/b) + n^c$, we get:
a = 1
b=10/7
c= 1

Here, a/b^c = 7/10. Hence we get - a/b^c ¡ 1. According to Master Theorem, if a/b^c is less than 1, T(n) = θ(n^c)

Here we get, T(n) = $\theta(n)$

(c)

$$T(n) = 16T(n/4) + n^2 \tag{3}$$

Comparing this to the equation $aT(n/b) + n^c$,
a = 16
b = 4
c= 2
Hence, $a/b^c = 1$
According to the Master Theorem, if $a/b^c = 1$,
then we the T(n) = θ(n^c$logn$).

(d)

$$T(n) = 7T(n/3) + n^2 \tag{4}$$

Comparing this to - $aT(n/b) + n^c$, we get
a= 7
b= 3
c=2

We get $a/b^c < 1$
According to the Master Theorem, if- $a/b^c < 1$ then,
T(n) = $\theta(n^c)$
$Hence, T(n) = \theta(n^2)$

(e)

$$T(n) = 7T(n/2) + n^2 \tag{5}$$

Here,
a = 7
b = 2
c = 2

Also, we get - $a/b^c < 1$. According to the Master Theorem, if- $a/b^c < 1$ then,
T(n) = $\theta(\mathrm{n}^{log_b a})$

(f)

$$T(n) = 7T(n/2) + n^2 \tag{6}$$

Here a = 7
b = 2
c = 2
Also, we get - $a/b^c > 1$
According to Master theorem,
if $a/b^c > 1$, then
T(n) = $\theta(\mathrm{n}^{log_b a})$

(g)

$$T(n) = T(n-2) + n^2 \tag{7}$$

Using the recurssive method, $= T(n-4) + (n-2)^2 + n^2$
$= T(n-6) + (n-4)^4 + (n-2)^2 + n^2$
$= T(n-8) + T(n-6) + (n-4)^2 + (n-2)^2 + n^2$
$... = T(1) + 2^2 + 4^2 + 6^2 + ... + (n-2)^2 + n^2$
$= T(1) + 2^2(1 + 2^2 + 3^2 + ... + (n/2)^2)$
$= T(1) + 4.(n/2)(n/2 + 1)(2(n/2) + 1)/6$
Therefore, T(n) = $\theta(n^3)$
Hence, taking $T(n) \leq ck^3$
$T(n) \leq c(n-3)^3 + n^2$
$= c(n^3 - 6n^2 + 12n - 8) + n^2$
$= cn^3 - ((6c-1)n^2 - 12n + 8c)$
this is true for- $1/6 \leq c$

$Ans6.(a)T(n) = 4T(n/3) + nlgn$
(8)

We assume, T(n) $\leq$n$^{log4/log3}$
$T(n) = 4T(n/3) + nlgn$
$4.c(n/3)^{1.25} + lgn$ $\leq$c.n$^{1.25}$

Now we assume, T(n) $\leq$c.n$^{1.25} - dnlgn$
T(n) $= 4T(n/3) + nlgn$
$= 4[c(n/3)^{1.25} - (dn/3)lg(n/3)] + nlgn$
$= [c.n^{1.25} - 4dn/3.lg(n/3)]$ + nlgn
= cn$^{1.25} - 4dn/3(lgn) - 4dn/3(lg3) + nlgn$
$= cn^{1.25} - dnlgn - 0.3dnlgn - 0.63dn + nlgn$
We want, $0.3dnlgn + 0.6dn + nlgn > 0$

Hence proved, T(n) = θ(n$^{log4/log3}$)

(b)
$$T(n) = 3T(n/3) + n/lgn \tag{9}$$
Assume $T(n) = \theta(n)$
$T(n) = c.n + n/lgn \leq c.n$
Hence we take,
$T(n) \leq c.n - dn/lgn$
Therefore, $T(n) = cn - d(n/3)/lg(n/3) + n/lgn \leq c.n$
= cn - d(n/lg (n/3)) + n/lgn
= cn - dn/(lgn - lg3) + n/lgn
= cn - [dn/(lgn - lg3) - n/lgn]$\leq$cn
$Hence, we want:$
[dn / (lgn - lg3) - n/lgn]$>$ 0
$Hence,$dn/(lgn-lg3)$should be >$n/lgn
$Hence proved, T(n) = \theta(n)$

(c)
$$T(n) = 4T(n/2) + n \tag{10}$$
Using Master Theorem,
a = 4
b = 2
c = 5/2
We find, a/b^c = 4/2$^{5/2}$ = 0.707 $a/b^c < 1$
Therefore, T(n) = θ(n^c)= $\theta(n^{2.5})$

(d)
$$T(n) = 3T(n/3 - 2) + n/2 \tag{11}$$
Using the recurssive method,
$T(n) = 3[3T(n/9 - 2) + n/6] + n/2$
$= 9T(n/9 - 2) + n/2 + n/2$
$= 3^3T(n/27 - 2) + n/2 + n/2 + n/2$

Following this pattern, we get $3^kT(n/3^k-2)+nk/2$
We equate $n/3^k-2=1$ for the case at T(1)
Therefore, $n=3^{k+1}$
$k=logn-1=log_3n-log_33$
$=log_3n/3$

Hence the equation would be $3^{log_3n}T(1)+n/2(log_3n/3)$
Therefore, T(n) = $\theta n(logn/3)$

(e)

$$T(n)=2T(n/2)+n/lgn \tag{12}$$

Assumsing, T(n) = $\theta(n^{log_ba})$
T(n) = $\theta(n^{log_22})$
$=\theta(n)$
$T(n)\leq cn$
$T(n)=2cn/2+n/lgn<cn$
This is not true, hence we subtract dn/lgn
We take $T(n)=\theta(n-n/lgn)$
$T(n)\leq cn-dn/lgn$ Therefore, $T(n)=2[cn/2-dn/2ln(n/2)+n/lgn]\leq cn$
$cn-dn/(lgn-lg2)+n/lgn\leq cn$
$cn-[dn(lgn-lg2)-n/lgn]\leq cn$

The term $dn/(lgn-lg2)-n/lgn$ should be greater to 0
$dn/(lgn-lg2)>n/lgn$
$d>(lgn-lg2)/lgn$
Therefore, T(n) = $\theta(n)$

(f)

$$T(n)=T(n/2)+T(n/4)+T(n/8)+n \tag{13}$$

Using the recursive method, $T(n)=T(n/4)+T(n/8)+T(n/16)+n/2+T(n/8)+T(n/16)+T(n/32)+n/4+T(n/16)+T(n/32)+T(n/64)+n/8$
which is equal to
$T(n)=T(n/2^2)$ + T(n/2^3) + T(n/2^4) + n/2 + T(n/2^3) + T(n/2^4) + T(n/2^5) + n/2^2 + T(n/2^4) + T(n/2^5) + T(n/2^6) + n/2^3

The sum of $n/2+n/4+n/8=7n/8$
This sum will keep growing in the order of $n[7/8+(7/8)^1+(7/8)^2+(7/8)^3+(7/8)^4+....+(7/8)^{k-1}]$
Adding the terms T(n/2^2) + T(n/2^3) + T(n/2^4) + T(n/2^3) + T(n/2^4) + T(n/2^5) gives us a max depth of log_2n and number of leaves as 3^{log_2n}.
The total sum would hence be $=n[7/8+(7/8)^1+(7/8)^2+(7/8)^3+(7/8)^4+....+(7/8)^{k-1}]+3^{log_2n}$

Therefore the sum $=n((7/8)/1-(7/8))+3^{log_n3}=7n+3^{log_23}$ Thus, $T(n)=\theta(n^{log_23})$