Problem Set #3 CSC236 Fall 2018

Si Tong Liu, shuo Yang, Jing Huang

31/10/2018

We declare that this assignment is solely our own work, and is in accordance with the University of Toronto Code of Behaviour on Academic Matters.

This submission has been prepared using LaTeX.

Problem 1.

(Warmup - this problem will NOT be marked).

Consider the following recurrence that results from some unspecified divide-and-conquer algorithm, where k is a positive constant and $1 \leq z \leq m-1$:

$$T(m,n) = \begin{cases} km, & n \leq 2 \\ kn, & m \leq 2 \\ kmn + T(z, n/2) + T(m-z, n/2), & m, n > 2 \end{cases}$$

Use **induction** to prove a Theta bound for $T(m,n)$. Do **NOT** use the substitution method. To help you guess the Theta bound, here are two possibilities; perhaps one of these is correct: $T(m,n) = \Theta(mn), T(m,n) = \Theta(m^2n^2)$.

Solution:

Firstly, we guess the Theta bound of $T(m,n)$ is $\Theta(mn)$.

Secondly, we choose to use complete induction to prove.

Predicate P(n): the Theta bound of $T(m,n)$ is $\Theta(mn)$, for k is a positive constant, $1 \leq z \leq m-1$ and $m, n \geq 0$

Base case: When $m = 2$ and $n = 2$, we have $z = m - 1 = 2 - 1 = 1$.

T(2,2) = 4k + T(1,1) + T(1,1)

= 4k + k*1*1 + k*1*1 = 6k

6k $\geq$ 1*mn(where m = n = 2, and c = 1) , big O prove completes

6k $\leq$ 2*mn(where m = n = 2, and c = 2) , big Omega prove completes

T(mn) is the bound of T(2,2) so base case holds.

Induction hypothesis: assume P(n) is true

Induction steps:

(1) prove Big O

assume $T(z, n/2) \leq c_1(zn/2)$ and $T(m-z, n/2) \leq c_2[(m-z)n/2]$

T(n) = kmn + T(z, n/2) + T(m-z, n/2)

$\leq$ kmn + $c_1(zn/2)$ + $c_2[(m-z)n/2]$

= kmn + $c_1zn/2$ + $c_2mn/2$ + $c_2Zn/2$

= kmn + $n/2(c_1z + c_2m - c_2z)$

$\leq$ kmn + $n/2(c_1 z + c_2 m)$

$\leq$ kmn + $n/2(c_1 m + c_2 m)$(since $z \leq m - 1$, so $z \leq m$)

= kmn + $(c_1 + c_2)mn/2$

= $(2k + c_1 + c_2)mn/2$

This completes the Big O proof, for $c = (2k + c_1 + c_2)$ and $m_0, n_0 = 1$

(2) prove Big Omega

assume $T(z, n/2) \geq c_1(zn/2)$ and $T(m - z, n/2) \geq c_2[(m - z)n/2]$

T(n) = kmn + T(z, n/2) + T(m-z, n/2)

$\geq$ kmn + $c_1(zn/2)$ + $c_2[(m - z)n/2]$

= kmn + $c_1 zn/2$ + $c_2 mn/2$ + $c_2 Zn/2$

= kmn + $n/2(c_1 z + c_2 m - c_2 z)$

= kmn

This completes the Big Omega proof, for $c_0 = k$ and $m_0 and n_0 = 1$

In conclusion, the Big Theta bound for $T(m, n)$ is $\Theta(mn)$

Problem 2.

(6 Marks) Consider the following three methods of solving a particular problem (input size n):

1. You divide the problem into three subproblems, each $\frac{1}{5}$ the size of the original problem, solve each recursively, then combine the results in time linear in the original problem size.

2. You divide the problem into 16 subproblems, each $\frac{1}{4}$ of size of the original problem, solve each recursively, then combine the results in time quadratic in the original problem size.

3. You reduce the problem size by 1, solve the smaller problem recursively, then perform an extra "computation step" that requires linear time.

Assume the base case has size 1 for all three methods.

For each method, write a recurrence capturing its worst-case runtime. Which of the three methods yields the fastest asymptotic runtime?

In your solution, you should use the Master Theorem wherever possible. In the case where the Master Theorem doesn't apply, *clearly state why not* based on your recurrence, and show your work solving the recurrence using another method (no proofs required).

Solution:

1. assume $T(n) = 3T(n/5) + O(n)$

Then use Master Theorem:

$a = 3, b = 5, f(n) = n$, where $k = 1$

$log_b a = log_5 3 < k$ (since $log_5 3 < 0$)

so $T(n) = O(n)$

2. assume $T(n) = 16T(n/4) + O(n^2)$

Then we use Master Theorem:

$a = 16, b = 4, f(n) = n^2$, where $k = 2$

$log_b a = log_4 16 = 2 = k$

so $T(n) = O(n^2 log n)$

3. assume $T(n) = T(n-1) + c$ (where c is the constant runtime)

We can not use Master Theorem here, since the $(n-1)/n$ is not a constant.

Then we will use repeated substitution here.

when k = 1: $T(n) = T(n-1) + c$

when k = 2: $T(n) = T(n-2) + c + c$

when k = 3: $T(n) = T(n-3) + c + c + c$

...

$T(n) = T(n-k) + kc$

Since the base case is size of 1,we have $n - k = 1 => k = n - 1$

so $T(n) = 1 + (n-1)k$

$= nk + 1 - k$

$\leq nk + 1$

$\leq nk + n = (k+1)n$

So the runtime is O(n) for $c_0 = k + 1$ and $n_0 = 1$

Problem 3.

(8 MARKS) (Modelled after Exercise 14 from lecture notes, p.48).

Recall the recurrence for the worst case runtime of quicksort:

$$T(n) = \begin{cases} c, & \text{if } n \leq 1 \\ T(|L|) + T(|G|) + dn, & \text{if } n > 1 \end{cases}$$

where L and G are the partitions of the list. Clearly, how the list is partitioned matters a great deal for the runtime of quicksort.

1. Suppose the lists are split as follows: $|L| = \frac{n}{4}, |G| = \frac{3n}{4}$ at each recursive call.

   Find a tight asymptotic bound on the runtime of quicksort using this assumption.

2. Now suppose that the lists are always very unevenly split: $|L| = n - 4$ and $|G| = 3$ at each recursive call for $n > 4$. Find a tight asymptotic bound on the runtime of quicksort using this assumption.

Solution:

1. Since $|L| = \frac{n}{4}$ and $|G| = \frac{3n}{4}$

we can have $T(n) = T(\frac{n}{4}) + T(\frac{3n}{4}) + dn$

To find the tight asymptotic bound, we try to get the Big O bound.

Since $|L| < |G|$, we have:

$T(n) = T(\frac{n}{4}) + T(\frac{3n}{4}) + dn$

$\leq T(\frac{3n}{4}) + T(\frac{3n}{4}) + dn$ (since n$\geq$0)

$= 2T(\frac{3n}{4}) + dn$

then we use Master Theorem:

a = 2 b = $\frac{4}{3}$, $f(n) = n$, where k = 1

$\log_b a = \log_{\frac{4}{3}} 2 \approx 2.4 \geq 1$

So the Big O bound for T(n) is $O(n^{2.4})$

In order to find the tightest bound, we make a guess of $O(n)$.

then we have $T(n) \leq c_0 n$, where c_0 is a positive number and $n_0 \geq 0$

we use complete induction to prove this

Firstly, the predicate P(n): $T(n) \in O(n)$, for $c_0 positive and n_0 \geq 0$

Secondly, the base case: when n = 0, $T(0) = T(0) + T(0) + 0 = 2c$

$\leq 2n$ (where $n \geq c$)

so the base case holds for $c_0 = 2$ and $n_0 = c$

Thirdly, the induction hypothesis: we assume P(n) holds for $P(1), P(2), p(3)....and P(n)$

Fourthly, the induction step:

$T(n) = T(\frac{n}{4}) + T(\frac{3n}{4}) + dn$

$\leq T(\frac{3n}{4}) + T(\frac{3n}{4}) + dn$ (since $\frac{3n}{4} < \frac{3n}{4}$)

$= 2T(\frac{3n}{4}) + dn$

suppose $T(\frac{3n}{4}) \leq c_0 \frac{3n}{4}$ (where $T(n) \leq c_0 n$)

show $T(n) = 2T(\frac{3n}{4}) + dn$

$= 2c\frac{3n}{4} + dn = (\frac{3c}{2} + d)n$

Therefore, we choose $c_0 = \frac{3c}{2} + d$ and $n_0 = 2$ to complete the proof

The tightest Big O bound is O(n)

2. Since $|L| = n - 4$ and $|G| = 3$, we can assume the run time of $T(|G|)$ is p, where p is a constant number

Then we have $T(n) = T(n-4) + p + dn$

We use repeated substitution to find the closed form

when k = 1 : $T(n) = T(n-4) + p + dn$

when k = 2 : $T(n) = (T(n-4-4) + p + d(n-4)) + p + dn$

$= T(n-8) + 2p + 2dn - 4d$

when k = 3 : $T(n) = T(n-8-4) + p + d(n-8) + 2p + 2dn - 4d$

$= T(n-12) + 3p + 3dn - 12d$

when k = 4 : $T(n) = T(n-12-4) + p + d(n-12) + 3p + 3dn - 12d$

$= T(n-16) + 4p + 4dn - 24d$

...

$T(n) = T(n-4k) + kp + kdn - 4\frac{(k-1)k}{2}$

$= T(n-4k) + kp + kdn - 2(k-1)k$

when $n - 4k = 0 => k = \frac{n}{4}$ (Base case)

So we get $T(n) = c + \frac{n}{4}p + \frac{n}{4}dn - \frac{n^2}{8} + \frac{n}{2}$

(1) Big O prove:

$T(n) = c + \frac{n}{4}p + \frac{n}{4}dn - \frac{n^2}{8} + \frac{n}{2}$

$\leq c + \frac{n}{4}p + \frac{n^2}{4}d + \frac{n}{2}$

$\leq cn^2 + pn^2 + dn^2 + n^2$, where $n \geq 0$

$=(c + p + d + 1)n^2$

This completes the Big O proof with $c_0 = c + p + d + 1$ and $n_0 = 0$

(2) Big Omega prove:

$T(n) = c + \frac{n}{4}p + \frac{n}{4}dn - \frac{n^2}{8} + \frac{n}{2}$

$\geq \frac{n}{4}dn - \frac{n^2}{8}$ (since p,d are positive numbers and $n_0 > 0$)

$= \frac{2d-1}{8}n^2$ (where $2d - 1$ needs to positvie $=> d > \frac{1}{2}$)

This completes the proof with $2d - 18$, and $n_0 = 0$

In conclusion, the tight asymptotic bound is $\Theta(n^2)$

Problem 4.

(8 MARKS)

Video rankings. The InstaVid social network collects user preferences by asking them to rank their favorite videos. One of the features of InstaVid is offering friendship suggestions for users with similar tastes, using the following metric.

If user *One* ranks the videos using the sequence $1, 2, \ldots, n$, and user *Two* ranks same videos using the sequence $v_1, v_2, \ldots, v_n$ of numbers $1, 2, \ldots, n$, then their social distance is computed by counting all pairs (v_i, v_j) from the ranking of *Two* that satisfy the condition $v_i > v_j$ for $i < j$. For example, if the user *One* ranks four videos as $1, 2, 3, 4$ and *Two* ranks them as $3, 1, 2, 4$ then their social distance is 2 because of the pairs $(3, 1)$ and $(3, 2)$ in the rankings of *Two*.

(i) Design an algorithm `social_distance(prefa, prefb)` with time complexity in $\Theta(n^2)$ that computes the social distance for users with video rankings `prefa` and `prefb`. Please write your solution in the form of a Python function. Justify the run time of your code.

(ii) Improve your algorithm using divide and conquer approach. Fully analyse your algorithm; you may use the Master Theorem. Write your solution in the form of a Python function. (You may also write pseudocode if desired; we will not run your code anyway, but it is aceptable to actually implement your code in Python and copy it in your soluton.)

Solution:

1.

```
def social_distance(prefa, prefb):
    count = 0

    list_tuple = []
    for num in prefa:
        for num_p in prefa:
            if num < num_p:
                list_tuple += [(num,num_p)]

    for t in list_tuple:
        if prefb.index(t[0]) > prefb.index(t[1]):
            count += 1
```

```
    return count
```

According to the python code above, the runtime complexity can be written as:

T(n) = 1 + 1 + $2n^2$ + 2n + 1 = $2n^2 + 2n + 3$

Then we are going to prove the Big Theta bound of T(n) is $\Theta(n^2)$ (1) Big O proof: $T(n) = 2n^2 + 2n + 3$

$\leq 2n^2 + 2n^2 + 3n^2$ (if $n \geq 0$)

=$7n^2$

So this completes the Big O proof with c = 7 and $n_0 = 0$

(2) Big Omega proof: $T(n) = 2n^2 + 2n + 3$

$\geq 2n^2 + 2n$

$\geq 2n^2$

So this completes the Big Omega proof with c = 2 and $n_0 = 0$

Therefore, the Big Theta bound for the python function is $\Theta(n^2)$

2.

```
def social_distance(prefa, prefb):
    count = 0
    if prefb[0] greater than any rest of elements in prefb:
        count += 1
    return prefb[:len(prefb)/2]+ prefb[len(prefb)/2:] + count
```

so the runtime complexity of the function above is $T(n) = 2T(n/2) + n$

then we apply Master Theorem here

a = 2 b = 2, $f(n) = n$, where k = 1

$\log_b a = \log_2 2 = 1 = k$

Therefore the runtime of this function is O(nlogn).