Comp 160: Algorithms — Jiawei Wang
Tufts University, Fall 2019 — Homework 02

Question 1:

1.1 Recursion tree

- In order to find the upper and lower bounds using Recursion Tree, First we need to visualize $H(n) = H(\frac{n}{2}) + logn$ by drawing the levels:

$$\begin{tikzcd} logn \arrow[r] & log\frac{n}{2} \arrow[r] & \cdots\cdots \arrow[r] & H(1) \\ \text{Top level} & \text{2nd level} & & \text{leaf level} \end{tikzcd}$$

  Top level: $logn$ times 1 node;
  2nd level: $log\frac{n}{2}$ times 1 node;
  .....
  leaf level: H(1) times 1 node

- Lemma: The recursion tree has $logn$ height

  Proof. The problem size is n at the top level, $\frac{n}{2}$ at the second level and $\frac{n}{2^{i-1}}$ at i-th level. The recursion will stop when the problem size comes to a constant number, for example 2. So let h be the number of recursive calls we need to do before we reach the constant-sized problem then it holds that:

$$\frac{n}{2^{i-1}} = 2 \tag{1}$$
$$n = 2^i \tag{2}$$
$$logn = i \tag{3}$$

  So this tree has $logn$ levels □

- Let us sum up the total work:

$$T(n) = logn + log(\frac{n}{2}) + log(\frac{n}{4}) + ... + \frac{n}{2^{logn}} \tag{4}$$
$$T(n) = logn + logn - 1 + logn - 2 + ... + logn - logn \tag{5}$$

  This sums up to $\frac{(logn)(logn+1)}{2} = \theta((logn)^2)$.

- So the matching bound for $H(n)$ is: $\theta((logn)^2)$

1.2 Substitution

- Guess $H(n) = \theta((logn)^2)$
- Fist let us make $n = 2^k$ so we get:

$$H(2^k) = H(2^{k-1}) + log(2^k) \tag{6}$$

$$H(2^k) = H(2^{k-1}) + k \tag{7}$$

- Then we make $m(k) = H(2^k)$ and get:

$$m(k) = m(k-1) + k \tag{8}$$

$$m(k) = m(0) + 1 + 2 + ... + k \tag{9}$$

$$m(k) = m(0) + \theta(k^2) \tag{10}$$

- Assume that m(0)=1 then this recurrence solves to $\theta(k^2)$
- Since $m(k) = H(2^k) = H(n)$, we get that $H(n) = \theta(k^2) = \theta((logn)^2)$

1.3 Master method

- $H(n) = H(\frac{n}{2}) + logn$
  So $a = 1$,$b = 2$,$f(n) = logn$
- Number of leaves $= n^{{log_2}^1} = 1$
- $H(n) = logn$
- Case 2 applies (leaves and $H(n)$ are asymptotically equivalent), and thus: $H(n) = \theta((logn)^2)$

Question 2:

2.1

- First method is to cut the photos into the right size and leave another piece each time so the total work should be $f(n) = n - 1$ and the time complexity is $\theta(n)$
- We want to prove that $f(n) = \theta(n)$:

$$f(n) = n - 1 \tag{11}$$

$$f(n-1) = (n-1) - 1 \tag{12}$$

$$f(n) = f(n-1) + 1 \tag{13}$$

$$f(n) = f(1) + 1 + 1 + ... + 1 \tag{14}$$

- So we can see that this work takes n rounds and in each round the workload is 1, So the time complexity is $\theta(n)$
- Second method is to cut the photos in half and take the half-photo and cut it again for each one so the workload doubles each round and the total is $f(n) = 2f(\frac{n}{2}) + c$ and the time complexity is $\theta(n)$
- We want to prove that $f(n) = \theta(n)$:

$$f(n) = 2f(\frac{n}{2}) + c \tag{15}$$

- If we draw the recursion tree, we can see that the top level is c with 1 node, the second level is $\frac{c}{2}$ with 2 nodes and the i-th level is $\frac{c}{2^(i-1)}$ with 2^{i-1} nodes. So $f(n) = c + 2c + 4c + ... + nc$ and the total nodes should be $n + n - 1 = 2n - 1$
- Therefore $T(n) = 2n - 1 = \theta(n)$
  (please turn to next page)

- Thirf method is to cut the photos in half but rather take the two pieces together and cut together so each round the workload remains the same and the total should be $f(n) = f(\frac{n}{2}) + c$ and the time complexity is $\theta(logn)$
- We want to prove that $f(n) = \theta(logn)$:
- By using master method: Case 2 applies (leaves and $f(n)$ are asymptotically equivalent), and thus: $f(n) = \theta(logn)$

2.2

Third method is the best because the time complexity is the smallest. First and second method are both the worst because the time complexity is the biggest.

Question 3:

3.1 Recursion tree

- First let us check the tree levels:
- Lemma 2: The recursion tree has $loglogn$ height

$$\begin{tikzcd} n \arrow[r] & \sqrt{n} \arrow[r] & \cdots\cdots \arrow[r] & T(1) \\ \text{Top level} & \text{2nd level} & & \text{leaf level} \\ T(n) & T\left(\sqrt{n}\right) & & T\left(n^{\frac{1}{2^i}}\right) \end{tikzcd}$$

Proof. The problem size is n at the top level, $n^{\frac{1}{2}}$ at the second level and $n^{\frac{1}{2^i}}$ at i-th level. The recursion will stop when the problem size comes to a constant number, for example 2. So let h be the number of recursive calls we need to do before we reach the constant-sized problem then it holds that:

$$n^{\frac{1}{2^h}} = 2 \tag{16}$$

$$n = 2^{2^h} \tag{17}$$

$$logn = log2^{2^h} \tag{18}$$

$$logn = 2^h \tag{19}$$

$$loglogn = h \tag{20}$$

□

- Now that we know the exact number of times that we recurse, we can express the total work as:

(please turn to next page)

$$T(n) = \sum_{x=0}^{loglogn} \frac{1}{2^x} n \tag{21}$$

$$\leq \sum_{x=0}^{\infty} \frac{1}{2^x} n \tag{22}$$

$$= \frac{1}{1 - \frac{1}{2}} n \tag{23}$$

$$= 2n \tag{24}$$

- So $T(n) = O(n)$

3.2 Substitution

- Guess: $T(n) \leq O(n)$
- We prove this statement by substitution. Assume that $T(k) \geq ck$ for all $k < n$ and some $c > 0$. We want to show that $T(n) \geq cn$. Then by substitution we have:
- Fist let us make $n = 2^k$ so we get:

$$T(2^k) = T(2^{\frac{k}{2}}) + 2^k \tag{25}$$

$$\leq c2^{\frac{k}{2}} + 2^k \tag{26}$$

$$= c2^k 2^{-\frac{k}{2}} + c2^k \frac{1}{c} \tag{27}$$

$$= c2^k (2^{-\frac{k}{2}} + \frac{1}{c}) \tag{28}$$

- So as long as $2^{-\frac{k}{2}} + \frac{1}{c} \geq 1$, we will have $T(n) \geq cn$. We set $k \geq 1$ This holds as long as $c \geq 2 - \sqrt{2}$. Therefore $T(n) \leq O(n)$

3.3 Master method

- First observation is that we cannot directly apply the master method to $T(n)$ so we do a change of variables. We define $m = logn$ then $n = 2^m$ and:

$$T(n) = T(2^m) = T((2^m)^{\frac{1}{2}}) + 2^m \tag{29}$$

$$= T(2^{\frac{m}{2}}) + 2^m \tag{30}$$

- We now define a new recurrence $S(m)$ where$T(n) = T(2^m) = S(m)$ then

$$S(m) = S(\frac{m}{2}) + 2^m \tag{31}$$

- Now we have a recurrence in the correct form to apply the master method

  1. The number of leaves is $m^{log_b^a} = m^{log_2^1} = m^0 = 1$ and
  2. $f(m) = 2^m$
  3. Case 3 applies (the root level asymptotically dominates). Thus, we have $S(m) = \theta(f(m)) = \theta(2^m)$
  4. Now that we have a bound on $S(m)$ the only thing we have left is to undo the change of variables. Then:

$$\theta(2^m) = \theta(2^{logn}) = \theta(n) = T(n) \tag{32}$$