CSC236H, Winter 2019

Assignment 2

Jiacheng Ge 1003795814

Yulin WANG 1003942326

1. Step1: Find the closed-form expression for $T(x,k)$

   Assume that k is large enough and k > 1.
   Let i be the number of iterations and $i \in \mathbb{N}$. Then

$$\begin{aligned}
T(x,k) &= 2T(\frac{x}{2}, k-2) + \frac{1}{2}x && \text{\#by definition, since k > 1} \\
&= 2[2T(\frac{x}{2^2}, k-4) + \frac{x}{2^2}] + \frac{1}{2}x \\
&= 2^2T(\frac{x}{2^2}, k-4) + \frac{1}{2}x + \frac{1}{2}x && \text{\#i=2} \\
&= 2^2[2T(\frac{x}{2^3}, k-6) + \frac{x}{2^3}] + \frac{1}{2}x + \frac{1}{2}x \\
&= 2^3T(\frac{x}{2^3}, k-6) + \frac{1}{2}x + \frac{1}{2}x + \frac{1}{2}x && \text{\#i=3} \\
&\dots \\
&= 2^iT(\frac{x}{2^i}, k-2i) + \frac{i}{2}x
\end{aligned}$$

Case1: When k is even, let $i = \frac{k}{2}$. Then

$$\begin{aligned}
T(x,k) &= 2^{\frac{k}{2}}T(\frac{x}{2^{\frac{k}{2}}}, 0) + \frac{\frac{k}{2}}{2}x \\
&= 2^{\frac{k}{2}} \cdot \frac{x}{2^{\frac{k}{2}}} + \frac{k}{4}x \\
&= x + \frac{k}{4}x \\
&= (\frac{k+4}{4})x
\end{aligned}$$

Case2: When k is odd, let $i = \frac{k-1}{2}$. Then

$$\begin{aligned}
T(x,k) &= 2^{\frac{k-1}{2}}T(\frac{x}{2^{\frac{k-1}{2}}}, 1) + \frac{\frac{k-1}{2}}{2}x \\
&= 2^{\frac{k-1}{2}} \cdot \frac{x}{2^{\frac{k-1}{2}}} + \frac{k-1}{4}x \\
&= x + \frac{k-1}{4}x \\
&= (\frac{k+3}{4})x
\end{aligned}$$

   Therefore, The closed-form expression for $T(x,k)$ is:

$$T(x,k) = \begin{cases} (\frac{k+4}{4})x & \text{k is even} \\ (\frac{k+3}{4})x & \text{k is odd} \end{cases}$$

Step2: Prove the correctness of the expression.

Let the predicate $P(k)$ denote that $T_c(x,k) = T_R(x,k)$, where $x \in \mathbb{R}$

Prove that $\forall k \in \mathbb{N}, P(k)$ holds by complete induction.

Let $k \in \mathbb{N}$ and $x \in \mathbb{R}$
Base Case:
(1) When $k = 0, T(x,k) = T(x,0) = (\frac{0+4}{4})x = x$
Then $P(0)$ holds.
(2) When $k = 1, T(x,k) = T(x,1) = (\frac{1+3}{4})x = x$
Then $P(1)$ holds.

Induction Step:
Let $k \in \mathbb{N}$. Assume for all $j \in \mathbb{N}, 0 \leq j \leq k, P(j)$ holds. [IH]
WTP: $P(k+1)$ holds.
Assume $k > 1$
Then by definition of $T(x,k)$, we have
$T(x,k+1) = 2T(\frac{x}{2}, k+1-2) + \frac{1}{2}x = 2T(\frac{x}{2}, k-1) + \frac{1}{2}x$

Case1: When (k+1) is even , then (k-1) is also even.

$$\begin{aligned} T(x,k+1) &= 2[(\frac{k-1+4}{4}) \cdot \frac{1}{2}x] + \frac{1}{2} \qquad \text{\#by IH, since (k-1) is even and } 0 \leq k-1 < k+1\text{,so } P(k-1)\text{holds} \\ &= (\frac{k+3}{4})x + \frac{1}{2} \\ &= (\frac{k+5}{4})x \\ &= [\frac{(k+1)+4}{4}]x \end{aligned}$$

Thus, $P(k+1)$ holds when (k+1) is even.

Case2: When (k+1) is odd , then (k-1) is also odd.

$$\begin{aligned} T(x,k+1) &= 2[(\frac{k-1+3}{4}) \cdot \frac{1}{2}x] + \frac{1}{2}x \qquad \text{\#by IH, since (k-1) is odd and } 0 \leq k-1 < k+1\text{,so } P(k-1)\text{holds} \\ &= (\frac{k+2}{4})x + \frac{1}{2}x \\ &= (\frac{k+4}{4})x \\ &= [\frac{(k+1)+3}{4}]x \end{aligned}$$

Thus, $P(k+1)$ holds when (k+1) is odd.

Therefore, by the principle of complete induction, $\forall k \in \mathbb{N}, P(k)$ holds.

□

2. (a) Let n be an arbitrary natural number. Let $T(n)$ denote the maximum number of steps executed by a call to RecClosestValue(root, x). Let n be the height of the non-empty binary search tree T.

   If $n = 0$, tree T contains only one single node, that is $root.left == null$ and $root.right == null$, so line 2-4 execute, which take constant time, represented by a constant value a.

   Otherwise, when $n > 0$, line 5-15 execute.

   Case1: if $x < root.value$ and $root.left \neq null$, then line 6-8 execute. Since the height of left subtree is at most $(n-1)$ in worst case scenario. So the recursive call in line 7 takes at most $T(n-1)$ time.

   Case2: if $x > root.value$ and $root.right \neq null$, then line 9-11 execute. Since the height of right subtree is at most $(n-1)$ in worst case scenario. So the recursive call in line 10 takes at most $T(n-1)$ time.

   And since all other instructions in line 12-15(if $x == root.value$, line 12-15 also execute), as well as line 1 and line 5, take constant time, represented by a constant value b. Then, putting all together, we get the following definition of $T(n)$:

$$T(n) = \begin{cases} a & n = 0 \\ T(n-1) + b & n > 0 \end{cases}$$

   (b)Let predicate $P(n)$ denote that if input root is the root of a non-empty binary search tree T storing numbers and x is some number, and n is the height of T, then RecClosestValue(root, x) terminates and returns a node T with the closest value to x.

   Prove that $\forall n \in \mathbb{N}, P(n)$ holds by complete induction.

   Base case: when $n = 0$, T has only one single node, that is the root. So, the node in T with the closest value to x is the root itself. Line 2-4 execute, then it returns the root as wanted in line 3, and since there is no recursive call, so RecClosestValue(root, x) terminates. Thus, $P(0)$ holds.

   Induction Step: Let $n \in \mathbb{N}$, assume for all $j \in \mathbb{N}, 0 \leq j < n, P(j)$ holds. [I.H]
   WTP:$P(n)$ holds

   Assume $n \geq 1$, then line 5-15 execute.
   Let n_L be the height of left subtree and n_R be the height of right subtree.

   Case1: if $x < root.value$ and $root.left \neq null$, line 6-8 execute. Since $0 \leq n_L < n$, by I.H. $P(n_L)$ holds, that is RecClosestValue(root.left, x) terminates and it returns a node(which is closest) in left subtree with the closest value to x. And then line 12-15 execute, if | root.val-x | $\leq$ | closest.val-x |, then change the node closest to root, otherwise node closest in left subtree is the node closest in T. And by line 15, RecClosestValue(root,x) terminates and returns a node in T with the closest value to x as wanted. Thus, $P(n)$ holds for all $n \geq 1$ in this case.

   Case2: if $x > root.value$ and $root.right \neq null$, line 9-11 execute. Since $0 \leq n_R < n$, by I.H. $P(n_R)$ holds, that is RecClosestValue(root.right, x) terminates and it returns a node(which is closest) in right subtree with the closest value to x. And then line 12-15 execute, if | root.val-x | $\leq$ | closest.val-x |, then change the node closest to root, otherwise node closest in right subtree is the node closest in T. And by line 15, RecClosestValue(root, x) terminates and returns a node in T with the closest value to x as wanted. Thus, $P(n)$ holds for all $n \geq 1$ in this case.

   Case3: if $x == root.value$, then the node in T with the closest value to x is the root itself. Line 12-15 execute, since | root.val-x | $= 0 \leq$ | closest.val-x |, then change the node closest to root, and by line 15, RecClosestValue(root, x) terminates and returns the root as wanted. Thus, $P(n)$ holds for all $n \geq 1$ in this case.

   Therefore, by the principle of complete induction, $\forall n \in \mathbb{N}, P(n)$ holds.

$\square$

3. (a)Step1: Formulate a loop invariant.

   Let LI(k) denote the assertion that if the while loop is executed at least k times, then
   (1) $0 \leq lo_k \leq hi_k - 1 \leq len(A) - 1$
   (2) $sum_k = A[lo_k] + A[hi_k]$
   (3) $sum_{k-1} \neq x$

   Step2: Prove the LI.

   Prove LI(k) by simple induction.

   Base Case: On entering the while loop (when k=0)
   (1)By precondition, $len(A) \geq 2$, then $hi_0 = len(A) - 1 \geq 1$
   Since $lo_0 = 0 \leq 1$, so $0 \leq lo_0 \leq hi_0 - 1 \leq len(A) - 1$
   (2) By line 4, $sum_0 = A[lo_0] + A[hi_0]$
   (3) Since we do not enter the loop, so no previous iteration exists, then $sum_{k-1} \neq x$ is vacuously true.
   Thus, LI(0) holds.

   Induction Step: Let $k \in \mathbb{N}$. Assume that LI(k) holds. [IH]
   That is, if the loop is executed at least k times, then
   (i) $0 \leq lo_k \leq hi_k - 1 \leq len(A) - 1$
   (ii) $sum_k = A[lo_k] + A[hi_k]$
   (iii) $sum_{k-1} \neq x$
   WTP: LI(k+1) holds.

   Assume that k+1 iterations exist.

   (1) Note that since there is an iteration, so condition in line 5 must hold before the iteration.
   Then $lo_k < hi_k - 1$, or equivalently $lo_k + 1 \leq hi_k - 1$

   Case1: if $sum_k < x$, line 6-7 are executed, then $lo_{k+1} = lo_k + 1$, $hi_{k+1} = hi_k$

$$
\begin{aligned}
0 &\leq lo_k && \text{\#by IH}(i) \\
&\leq lo_k + 1 = lo_{k+1} && \text{\#by line 7} \\
&\leq hi_k - 1 = hi_{k+1} - 1 && \text{\#by condition in line 5} \\
&\leq len(A) - 1 && \text{\#by IH}(i)
\end{aligned}
$$

   Case2: if $sum_k > x$, line 8-9 are executed, then $hi_{k+1} = hi_k - 1$, $lo_{k+1} = lo_k$

$$
\begin{aligned}
0 &\leq lo_k && \text{\#by IH}(i) \\
&= lo_{k+1} \\
&\leq (hi_k - 1) - 1 && \text{\#by condition in line 5} \\
&= hi_{k+1} - 1 && \text{\#by line 9} \\
&\leq len(A) - 1 && \text{\#by IH}(i)
\end{aligned}
$$

   Thus, $0 \leq lo_{k+1} \leq hi_{k+1} - 1 \leq len(A) - 1$, then part (1) in LI(k+1) holds.

   (2) By line 11, $sum_{k+1} = A[lo_{k+1}] + A[hi_{k+1}]$, then part (2) in LI(k+1) holds.

   (3) Since k+1 iterations exist, then the while loop condition holds, so $sum_k = sum_{(k+1)-1} \neq x$, then part (3) in LI(k+1) holds.

   Thus, LI(k+1) holds.

   Therefore, by the principle of complete induction, $\forall k \in \mathbb{N}$, LI(k) holds.

$\square$

3.(b)Suppose the program terminates and the precondition holds.
Since the program terminates, then the while loop is executed a finite number of times, say t.
By exit condition of the while loop in line 5, the loop exits when $lo_t \geq hi_t - 1$ or $sum_t = x$
And by part (1) in LI, $lo_t \leq hi_t - 1$, so the loop exits when $lo_t = hi_t - 1$ or $sum_t = x$
By line 13, if $sum_t = x$, then line 14 is executed, TwoSum(A, x) terminates and returns [lo, hi], which is the pair [i, j] in the post condition, where $sum_t = A[lo_t] + A[hi_t] = x = A[i] + A[j]$ as wanted.# by part (2) in LI
Otherwise, line 15-16 is executed, TwoSum(A, x) terminates and returns [-1, -1] as wanted.

□

3.(c)Step1: Find an appropriate loop measure.
Let $m_k = hi_k - lo_k$

Step2: Prove the termination of TwoSum.

Assume that k iterations exist.
Since $hi_k, lo_k \in \mathbb{N}$ and $lo_k \leq hi_k - 1$ by part (1) in LI, then $m_k = hi_k - lo_k \geq 1$, so $m_k \in \mathbb{N}$

Case1: When $sum_k < x$, line 7 is executed, then $lo_k = lo_{k-1} + 1$, $hi_k = hi_{k-1}$, so we have

$$\begin{aligned}
m_k &= hi_k - lo_k && \text{[definition of } m_k\text{]} \\
&= hi_{k-1} - (lo_{k-1} + 1) && \text{[line 7]} \\
&= (hi_{k-1} - lo_{k-1}) - 1 \\
&= m_{k-1} - 1 && \text{[definition of } m_{k-1}\text{]} \\
&\leq m_{k-1}
\end{aligned}$$

Case2: When $sum_k > x$, line 9 is executed, then $hi_k = hi_{k-1} - 1$, $lo_k = lo_{k-1}$, so we have

$$\begin{aligned}
m_k &= hi_k - lo_k && \text{[definition of } m_k\text{]} \\
&= hi_{k-1} - 1 - lo_{k-1} && \text{[line 9]} \\
&= (hi_{k-1} - lo_{k-1}) - 1 \\
&= m_{k-1} - 1 && \text{[definition of } m_{k-1}\text{]} \\
&\leq m_{k-1}
\end{aligned}$$

Thus, m is always decreasing. Therefore the values of m form a decreasing sequence of natural numbers, that is the while loop eventually terminates.
And then line 13-17 execute. If $sum_k == x$, line 14 is executed and then it returns [lo, hi], so TwoSum(A, x) terminates. Otherwise, line 16 is executed and then it returns [-1, -1], so TwoSum(A, x) terminates. In conclusion, the Function TwoSum(A, x) eventually terminates.

□