236A3

siweitang

August 2020

1 MysterySort

1.1 a

$$T(n) = \begin{cases} a, n = 1 \\ T(n-1) + b(n-1) + c,\ \ n > 1 \end{cases} \tag{1}$$

For any natural number n, let T(n) denotes the maximum number of steps executed by a call to MySterySort(A,s,e). where n = e - s + 1.
If n =1, that means e - s = 0 , so e = s, then it does nothing except evaluating the if-condition, which take constant time, represented by a constant value a.
Otherwise, n > 1, e - s + 1 > 1, so e > s, line 2-12 executes.The recursive call in line 3 increments s by one, and therefore decrements n = e - s+ 1. Therefore, line 3 executes T(n - 1).The while loop in line 7 executes n -1 times.Therefore lines 7-9 takes b_1*(n-1), where b_1 is a constant value. Line 11 appends A[S], so it takes constant time. For line 12, it is list concatenation and there are total n - 1 elements, so it takes b_2 (n - 1) for concatenation. Also, line 4-6 and line 11 takes constant time, denotes as c
So in total, T(n)= T(n-1)+b_1(n- 1) +b_2(n- 1)+c. Simplify equation as T(n) = T(n-1)+2b(n- 1)+c where b = max$\{b_1, b_2\}$

1.2 b

Step 1 (Formulate a loop invariant)
Let LI(k) denote the assertion that if the loop is executed at least k times, then
(a) $C_k = B[0 : i_k]$
$(b) \forall j \in N, 0 \leq j \leq i_{k-1}, C_k[j] > A[s]$
$(c) C_k$ is sorted in non-increasing order.
$(d) 0 \leq i_k \leq len(B)$

Step 2(Prove the LI):
Basis: On entering the loop, C_0 is empty list, C_0 = [ ] and $i_0 = 0$
Therefore $0 \leq i_0 \leq$ len(B), also, since there are no elements in C_0,
so all the elements in C_0 are greater than A[s] and thus sorted in non-increasing order

Induction Step: Let k be an arbitrary number and assume that LI(k) holds.
That is, if the loop is executed at least k times, then (i) $C_k = B[0 : i_k]$, (ii) $\forall j \in N, 0 \leq j \leq i_{k-1}, C_k[j] > A[s]$ (iii) C_k is sorted in non-increasing order (iv) $0 \leq i_k \leq len(B)$
WTP: LI(k+1) holds
Assume that k+1 iterations exist. Then we have
$C_{k+1} = C_k.append(B[i_k])$ $[Line\ 8]$
$= B[0 : i_k].append(B[i_k])$ $[IH(i)]$
$= B[0 : i_{k+1}]$

as wanted for part (a) in LI(k+1)

By IH(ii), we know that after kth iteration, $\forall j \in N, 0 \leq j \leq i_{k-1}, C_k[j] > A[s]$
For the next iteration, (k+1)th iteration, by line 9, we know that $i_k = i_{k+1} - 1$ and $i_{k-1} = i_k - 1$
By line 7, since (k+1)th iteration exists, so $B[i_k] > A[s]$
By line 8 ,we know that $C_{k+1} = C_k.append(B[i_k])$
$\Rightarrow \forall j \in N, 0 \leq j \leq i_k, C_{k+1}[j] > A[s]$
$\Rightarrow \forall j \in N, 0 \leq j \leq i_{k+1} - 1, C_{k+1}[j] > A[s]$
as wanted for par(b) in LI(k+1)

By IH(iii), we know that after kth iteration, C_k is sorted in non-increasing order
By line 9, we know that $i_{k+1} = i_k + 1 \Rightarrow i_{k+1} > i_k$
By line 8, we know that $C_{k+1} = C_k.append(B[i_k])$
By line 3, B = MysterySort(A,s+1, e), so B is in non-increasing order
Since $i_{k+1} > i_k \Rightarrow B[i_k] \geq B[i_{k+1}$.Similarly, we can say that $B[i_{k-1}] \geq B[i_k]$
Since $C_{k+1} = C_k.append(B[i_k])$
By line 7, $B[i_k] = B[i_{k+1} - 1] > l = A[s]$
Since C_k is sorted in non-increasing order, also, $B[i_{k-1}]$ is the last element of C_k
$\Rightarrow B[i_{k-1}]$ is the smallest element in C_k
So for C_{k+1}, the new added element is $B[i_k]$ which is smaller than $B[i_{k-1}]$, the smallest element in C_k
$\Rightarrow$ We can say that the new added element $B[i_k]$ is the smallest element in C_{k+1} and it is at the end of list
So elements from index 0 to i_k is in non-increasing order $\Rightarrow C_{k+1}$ is in non-increasing order
as wanted for part (c) in LI(k+ 1)

Note that since another iterations exists, the condition on Line 7 must hold after k iterations.
Thus, $i_k < len(B)$, or equivalently $i_{k+1} \leq len(B)$. So we have
$0 \leq i_k$ $[IH(iv)]$
$< i_{k+1}$ [Since by Line 9, $i_{k+1} = i_k + 1$]
$\leq len(B)$
as wanted for part (d) in LI(k+ 1)

Step 3: (Prove partial correctness)
Suppose the precondition holds and the program terminates. Since the program terminates, the loop is executed a finite number of times, say t. Consider the values of C_t, i_t on exit.
By part (d) in LI, $i_t \leq len(B)$
By exit condition, $i_t \geq len(B)$ or the while loop ends on m iterations since $B[i_m] \leq l$
Case 1: $i_t \geq len(B)$
Hence $i_t = len(B)$. By a in LI, $C_t = B[0 : i_t]$
So C_t contains all elements in B and is sorted in non-increasing order
By line 7, all elements in C_t is greater than l = A[s], then by line 11, l is the last element of C_t
SO the return list is sorted in non-increasing order which contains all elements of A[s: e+1] and nothing else

Case 2: the while loop ends on m iterations since $B[i_m] \leq l$
By a in LI, $C_m = B[0 : i_m]$, by c in LI, all the elements of C_m are in non-increasing order.
By d, all the elements in C_m is greater than L= A[s]
So, on line 11, $C_m.append(l)$, C_m is still in non-increasing order
By exit condition, $B[i_m] < l$, also since B is sorted in non-increasing order
$\Rightarrow$ all the elements in $B[i_m : len(B)]$ are smaller than or equal to l and are sorted in non-increasing order
So by line 12, the list concatenation $C_m + B[i_m : len(B)]$ is sorted in non-increasing order
Since the return list contains all the element in B and A[s], so it contains all elements of A[s: e+1] and nothing else

Step 4: (Find an appropriate loop measure)

Let m_k = len(B) - i_k

Step 5: (Prove that the loop measure is a natural number on entering the loop and after every iteration, and decreases with every iteration):

Assume that k iterations exist.
By part (d) in LI, $i_k \leq len(B)$. So $m_k = len(B) - i_k \geq 0$. Thus m_k is always a natural number.
$m_k = len(B) - i_k$ [definition of m_k]
$= len(B) - (i_{k-1} + 1)$ [Line 9]
$= len(B) - i_{k-1} - 1$
$= m_{k-1} - 1$ [definition of m_{k-1}]
$< m_{k-1}$
Thus, m decreases after every iterations,and so the values of m form a strictly decreasing sequence of natural numbers, which must be finite sequence
The loop eventually terminates

2 closed-formed expression

$$
\begin{aligned}
T(x,n) &= 2^3 T(\frac{x}{2}, n-2) + 3x \\
&= 2^3(2^3 T(\frac{x}{4}, n-4) + 3 \times \frac{x}{2}) + 3x \\
&= 2^3(2^3 T(\frac{x}{4}, n-4)) + 3 \times 2^2 \times x + 3x \\
&= 2^6(2^3 T(\frac{x}{8}, n-6) + 3 \times \frac{x}{4}) + 3 \times 2^2 \times x + 3x \\
&= 2^9 T(\frac{x}{8}, n-6) + 3 \times 2^4 \times x + 3 \times 2^2 \times x + 3x \\
k=1, T(x,n) &= 2^3 T(\frac{x}{2}, n-2) + 3x \\
k=2, T(x,n) &= 2^3(2^3 T(\frac{x}{4}, n-4)) + 3 \times 2^2 \times x + 3x \\
k=3, T(x,n) &= 2^9 T(\frac{x}{8}, n-6) + 3 \times 2^4 \times x + 3 \times 2^2 \times x + 3x
\end{aligned}
$$

$$
\text{Let k} = \lfloor \frac{n}{2} \rfloor, T(x,n) = 3x(2^{3\lfloor \frac{n}{2} \rfloor} + \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor - 1} 2^{2i})
$$