Recursively Defined Functions (Continued)

Example 1. What is the running time of MergeSort algorithm as defined below:

```
1: function MergeSort(A, s, f)
2:   if s == f then
3:     return
4:   else
5:     m = (s + f)/2
6:     MergeSort(A, s, m)
7:     MergeSort(A, m+1, f)
8:     # merge sorted A[s..m] and A[m+1..f] back into A[s..f]
9:     for i = s, ···, f do
10:        B[i] = A[i]
11:     end for
12:     c = s
13:     d = m + 1
14:     for i = s, ···, f do
15:        if d > f or (c ≤ m and B[c] < B[d]) then
16:           A[i] = B[c]
17:           c = c + 1
18:        else
19:           A[i] = B[d]
20:           d = d + 1
21:        end if
22:     end for
23:   end if
24: end function
```

Ans: Recall that the worst-case running time of a for loop
```
for i = s, ···, f do
  BODY
end for
```
is \((f - s + 1) \times T_B\) where \(T_B\) is the worst-case running time of \(BODY\).

Hence, worst-case runtime \(T(n)\) of MergeSort satisfies:

\[
T(1) = 2 \quad \text{(lines 2-3)}
\]
\[
T(n) = k \quad \text{(all constant time work outside recursive calls)}
\]
\[
+ T(\lfloor \frac{n}{2} \rfloor) \quad \text{(line 6)}
\]
\[
+ T(\lceil \frac{n}{2} \rceil) \quad \text{(line 7)}
\]
\[
+n \quad \text{(lines 9-11)}
\]
\[
+3n \quad \text{(lines 14-21)}
\]
Repeated substitution:

\[ T(n) = 2T\left(\frac{n}{2}\right) + 4n + k \]
\[ = 2(2T\left(\frac{n}{4}\right) + 4\left(\frac{n}{2}\right) + k) + 4n + k \]
\[ = 4T\left(\frac{n}{4}\right) + 2 \times 4n + 3k \]
\[ = 4(2T\left(\frac{n}{8}\right) + 4\left(\frac{n}{4}\right) + k) + 2 \times 4n + 3k \]
\[ = 8T\left(\frac{n}{8}\right) + 3 \times 4n + 7k \]

Educated guess: After \(i\) steps

\[ T(n) = 2^i T\left(\frac{n}{2^i}\right) + 4n \times i + (2^i - 1)k \]

the base case is reached when \(\frac{n}{2^i} = 1\) or \(i = \log_2 n\).

Hence we expect \(T(n) \in \Theta(n \log n)\). In other words, we have to prove \(T(n) \in O(n \log n)\) and \(T(n) \in \Omega(n \log n)\).

\([T(n) \in \Omega(n \log n)]\): We need a \(B, c > 0\) such that \(T(n) \geq cn \log n\). Since we are working with powers of 2 in this problem and \(\log n = \log_2 n / \log_2 e\), we can change the base and try to prove \(T(n) \geq cn \log_2 n\) instead. The value of \(c\) in the new statement is just a multiple of the \(c\) in the original statement and we only need the existene of “a” \(c\) value. Now, Let’s look at the induction step.

\[ T(n) = T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + 4n + k \]
\[ \geq c(\lceil \frac{n}{2} \rceil \log_2 \lceil \frac{n}{2} \rceil) + c(\lfloor \frac{n}{2} \rfloor \log_2 \lfloor \frac{n}{2} \rfloor) + 4n + k \]
\[ \geq c(\lceil \frac{n}{2} \rceil + \lfloor \frac{n}{2} \rfloor) \log_2 \frac{n}{2} + 4n + k \]
\[ = cn \log_2 \frac{n}{2} + 4n + k \]
\[ \geq cn \log_2 \frac{n-1}{2} + 4n + k \]  \([\lceil \frac{n}{2} \rceil \geq \frac{n-1}{2}\])
\[ \geq cn \log_2 (n-1) - cn \log_2 2 + 4n + k \]
\[ \geq cn \log_2 \frac{n}{2} + (4 - c)n + k \]  \((x \geq 2 \Rightarrow x - 1 \geq \frac{x}{2}\))
\[ \geq cn \log_2 n + (4 - 2c)n + k \]

but to conclude that the last line is \(\geq cn \log_2 n\), we need:

\[(4 - 2c)n + k \geq 0 \Rightarrow c \leq 2 + \frac{k}{2n}\]

Let’s choose \(c = 2\). Are we done? No! we have to check the base case which in this case will be \(n = 2\) since we exploit the fact that \(n \geq 2\) in our analysis of the induction step.

\[ T(2) = k + 2 \times T(1) + 8 = k + 2 \times 2 + 8 = k + 12 > 4 = 2 \times 2 \times \log_2 2 \]

Good. Let’s proof \(T(n) \geq 2n \log_2 n\) for all \(n \geq 2\).
Define Proof.

Induction step: For \( i > 2 \), suppose \( T(j) \geq 2j \log_2 j \) for all \( 2 \leq j < i \). For \( P(i) \) we have

\[
T(i) = T\left(\left\lfloor \frac{i}{2} \right\rfloor \right) + T\left(\left\lceil \frac{i}{2} \right\rceil \right) + 4i + k
\]

\[
\geq 2\left(\left\lfloor \frac{i}{2} \right\rfloor \log_2 \left\lfloor \frac{i}{2} \right\rfloor \right) + 2\left(\left\lceil \frac{i}{2} \right\rceil \log_2 \left\lceil \frac{i}{2} \right\rceil \right) + 4i + k \quad \text{(by III)}
\]

\[
\geq 2\left(\left\lfloor \frac{i}{2} \right\rfloor + \left\lceil \frac{i}{2} \right\rceil \right) \log_2 \left\lceil \frac{i}{2} \right\rceil + 4i + k
\]

\[
= 2n \log_2 \left\lceil \frac{i}{2} \right\rceil + 4i + k
\]

\[
\geq 2i \log_2 \frac{i-1}{2} + 4i + k \quad \left( \left\lceil \frac{i}{2} \right\rceil \geq \frac{i-1}{2} \right)
\]

\[
\geq 2i \log_2 (i-1) - 2i \log_2 2 + 4i + k
\]

\[
\geq 2i \log_2 \frac{i}{2} + 2i + k \quad (x \geq 2 \Rightarrow x - 1 \geq \frac{x}{2})
\]

\[
\geq 2i \log_2 i - 2i + 2i + k
\]

\[
\geq 2i \log_2 i 
\]

\[
(k > 0)
\]

Hence by complete induction \( P(n) \) holds for all \( n \geq 2 \) which means that \( T(n) \in \Omega(n \log n) \).

\[\square\]

\([T(n) \in O(n \log n)]:\) A backward process similar to the one that we did to to prove \( T(n) \in \Omega(n \log n) \) results in statement \( T(n) \leq cn \log_2 n \) for \( c = \frac{6+k}{2-\log_2 3} \) and all \( n \geq 2 \).

Proof. Define \( P(n) : T(n) \leq cn \log_2 n \) for \( c = \frac{6+k}{2-\log_2 3} \). We want to prove by complete induction that \( \forall n \geq 2, P(n) \).

Base case:

\[
T(2) = T(1) + T(1) + 4 + k
\]

\[
= 12 + k
\]

\[
< 12 + 2k
\]

\[
< \frac{12+k}{\log_2 2} \quad (\log_2 \frac{4}{3} < \log_2 2 = 1)
\]

\[
= \frac{6+k}{2-\log_2 3} \times 2 \log_2 2
\]

Induction step: Let \( i > 3 \) and suppose \( T(j) \leq cj \log_2 j \) for \( 2 \leq j < i \) (III). We want to prove \( P(i) \) is true.

\[
T(i) = T\left(\left\lfloor \frac{i}{2} \right\rfloor \right) + T\left(\left\lceil \frac{i}{2} \right\rceil \right) + 4i + k
\]

\[
\leq c\left(\left\lfloor \frac{i}{2} \right\rfloor \log_2 \left\lfloor \frac{i}{2} \right\rfloor \right) + c\left(\left\lceil \frac{i}{2} \right\rceil \log_2 \left\lceil \frac{i}{2} \right\rceil \right) + 4i + k \quad \text{(by III)}
\]

\[
\leq ci \log_2 \left\lfloor \frac{i}{2} \right\rfloor + 4i + k
\]

\[
\leq ci \log_2 \frac{3i}{4} + 4i + k \quad (2 \leq x \Rightarrow \left\lfloor \frac{x}{2} \right\rfloor \leq \frac{x+1}{2} \leq \frac{3x}{4})
\]

\[
= ci \log_2 i - ci \log_2 \frac{4}{3} + 4i + k
\]

\[
\leq ci \log_2 i - (6+k)i + 4i + k \quad \text{(by choice of } c)\]

\[
= ci \log_2 i - (2+k)i + k
\]

\[
\leq ci \log_2 i - (2+k) + k \quad (2 \leq i)
\]

\[
\leq ci \log_2 i
\]
Therefore, by complete induction \( T(i) \leq ci \log_2 i \) and hence \( T(n) \in O(n \log n) \).

Now that we have proved that \( T(n) \in O(n \log n) \) and \( T(n) \in \Omega(n \log n) \), we can conclude that \( T(n) \in \Theta(n \log n) \).

**Divide and Conquer Algorithms**

MergeSort and RecBSearch functions are two examples of a general class of algorithms that use a problem-solving technique known as “divide and conquer”. To solve a large instance of a problem, a divide and conquer technique splits up the instance into smaller instances, often of the same size, solves subproblems recursively and combine solutions to the smaller instances to construct a solution for the large instance.

The worst-case running times of such algorithms satisfy recurrences of the form:

\[
T(n) = \begin{cases} 
K & \text{if } n \leq B \\
 a_1 T(\lceil \frac{n}{b} \rceil) + a_2 T(\lfloor \frac{n}{b} \rfloor) + f(n) & \text{if } n > B
\end{cases}
\]

for constants \( K > 0, a_1 \geq 0, a_2 \geq 0, b > 1, B > 0 \) from the algorithm. We expect a divide and conquer algorithm to make at least one call. Hence \( a_1 + a_2 > 0 \).

**Remark.** To simplify the analysis and understanding of worst-case running time of divide and conquer algorithms, in (1), the base cases are assumed to have similar running time. This assumption does not limit the application of (1) as one can always assume that \( T(n) \geq \min_{i \leq B} T(i) \) or \( T(n) \leq \max_{i \leq B} T(i) \) for \( n \leq B \). Both of these assumptions will not change the order of asymptotic bounds and can only affect the constants in these bounds.

In our discussion, so far, we have seen instances of (1) for which a closed-form formula was achievable. But can we find a closed-form formula for the general case?

**Theorem 1** (Master Theorem). If \( f(n) = \Theta(n^d) \) for constant \( d \geq 0 \), and \( a = a_1 + a_2 \), then the recurrence \( T(n) \) above has closed-form solution:

\[
T(n) = \begin{cases} 
\Theta(n^d) & \text{if } a < b^d \\
\Theta(n^d \log n) & \text{if } a = b^d \\
\Theta(n^{\log_b a}) & \text{if } a > b^d
\end{cases}
\]

**Reading.** Please see the textbook, pages 87–90, for a proof of this theorem.

**Remark.** You will see a more general form of this theorem in CSC263.