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:          $m = (s + f)/2$                                    ▷ Integer Division
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 $i = s, \cdots, f$ do
10:             B[i] = A[i]
11:         end for
12:         $c = s$
13:         $d = m + 1$
14:         for $i = s, \cdots, f$ do
15:             if $d > f$ or ($c \leq m$ and $B[c] < B[d]$) then
16:                 A[i] = B[c]
17:                 $c = c + 1$
18:             else                          ▷ $d \leq f$ and ($c > m$ or $B[c] \geq B[d]$)
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 $i = s, \cdots, f$ 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:

$$\begin{aligned}
T(1) &= 2 && \text{(lines 2-3)} \\
T(n) &= k && \text{(all constant time work outside recursive calls)} \\
&\quad + T(\lfloor \tfrac{n}{2} \rfloor) && \text{(line 6)} \\
&\quad + T(\lceil \tfrac{n}{2} \rceil) && \text{(line 7)} \\
&\quad + n && \text{(lines 9-11)} \\
&\quad + 3n && \text{(lines 14-21)}
\end{aligned}$$

Repeated substitution:

$$\begin{aligned}
T(n) &= 2T(\frac{n}{2}) + 4n + k \\
&= 2(2T(\frac{n}{4}) + 4(\frac{n}{2}) + k) + 4n + k \\
&= 4T(\frac{n}{4}) + 2 \times 4n + 3k \\
&= 4(2T(\frac{n}{8}) + 4(\frac{n}{4}) + k) + 2 \times 4n + 3k \\
&= 8T(\frac{n}{8}) + 3 \times 4n + 7k
\end{aligned}$$

Educated guess: After i steps

$$T(n) = 2^i T(\frac{n}{2^i}) + 4n \times i + (2^i - 1)k$$

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

$$\begin{aligned}
T(n) &= 2^{\log_2 n} T(n/n) + 4n \log_2 n + (2^{\log_2 n} - 1)k \\
&= T(1)n + 4n \log_2 n + kn - k \\
&= 4n \log_2 n + (2 + k)n - k
\end{aligned}$$

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.

$$\begin{aligned}
T(n) &= T(\lceil \tfrac{n}{2} \rceil) + T(\lfloor \tfrac{n}{2} \rfloor) + 4n + k \\
&\geq c(\lceil \tfrac{n}{2} \rceil \log_2 \lceil \tfrac{n}{2} \rceil) + c(\lfloor \tfrac{n}{2} \rfloor \log_2 \lfloor \tfrac{n}{2} \rfloor) + 4n + k \\
&\geq c(\lceil \tfrac{n}{2} \rceil + \lfloor \tfrac{n}{2} \rfloor) \log_2 \lfloor \tfrac{n}{2} \rfloor + 4n + k \\
&= cn \log_2 \lfloor \tfrac{n}{2} \rfloor + 4n + k \\
&\geq cn \log_2 \tfrac{n-1}{2} + 4n + k && (\lfloor \tfrac{n}{2} \rfloor \geq \tfrac{n-1}{2}) \\
&\geq cn \log_2 (n-1) - cn \log_2 2 + 4n + k \\
&\geq cn \log_2 \tfrac{n}{2} + (4 - c)n + k && (x \geq 2 \Rightarrow x - 1 \geq \tfrac{x}{2}) \\
&\geq cn \log_2 n + (4 - 2c)n + k
\end{aligned}$$

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 $n \geq 2$.

Proof. For $n \geq 2$ define $P(n) : T(n) \geq 2n\log_2(n)$.
Base case: $T(2) = 12 + k \geq 4 = 2 \times 2 \times \log_2 2$.
Induction step: For $i > 2$, suppose $T(j) \geq 2j\log_2 j$ for all $2 \leq j < i$. For $P(i)$ we have

$$\begin{aligned}
T(i) &= T(\lceil \tfrac{i}{2} \rceil) + T(\lfloor \tfrac{i}{2} \rfloor) + 4i + k \\
&\geq 2(\lceil \tfrac{i}{2} \rceil \log_2 \lceil \tfrac{i}{2} \rceil) + 2(\lfloor \tfrac{i}{2} \rfloor \log_2 \lfloor \tfrac{n}{2} \rfloor) + 4i + k && \text{(by IH)} \\
&\geq 2(\lceil \tfrac{i}{2} \rceil + \lfloor \tfrac{i}{2} \rfloor) \log_2 \lfloor \tfrac{i}{2} \rfloor + 4i + k \\
&= 2n \log_2 \lfloor \tfrac{i}{2} \rfloor + 4i + k \\
&\geq 2i \log_2 \tfrac{i-1}{2} + 4i + k && (\lfloor \tfrac{i}{2} \rfloor \geq \tfrac{i-1}{2}) \\
&\geq 2i \log_2 (i-1) - 2i \log_2 2 + 4i + k \\
&\geq 2i \log_2 \tfrac{i}{2} + 2i + k && (x \geq 2 \Rightarrow x - 1 \geq \tfrac{x}{2}) \\
&\geq 2i \log_2 i - 2i + 2i + k \\
&\geq 2i \log_2 i && (k > 0)
\end{aligned}$$

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

□

$[T(n) \in O(n \log n)]$: 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:

$$\begin{aligned}
T(2) &= T(1) + T(1) + 4 + k \\
&= 12 + k \\
&< 12 + 2k \\
&< \frac{12+k}{\log_2 \frac{4}{3}} && (\log_2 \tfrac{4}{3} < \log_2 2 = 1) \\
&= \frac{6+k}{2-\log_2 3} \times 2\log_2 2
\end{aligned}$$

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

$$\begin{aligned}
T(i) &= T(\lceil \tfrac{i}{2} \rceil) + T(\lfloor \tfrac{i}{2} \rfloor) + 4i + k \\
&\leq c(\lceil \tfrac{i}{2} \rceil \log_2 \lceil \tfrac{i}{2} \rceil) + c(\lfloor \tfrac{i}{2} \rfloor \log_2 \lfloor \tfrac{i}{2} \rfloor) + 4i + k && \text{(by IH)} \\
&\leq ci \log_2 \lceil \tfrac{i}{2} \rceil + 4i + k \\
&\leq ci \log_2 \tfrac{3i}{4} + 4i + k && (2 \leq x \Rightarrow \lceil \tfrac{x}{2} \rceil \leq \tfrac{x+1}{2} \leq \tfrac{3x}{4}) \\
&= ci \log_2 i - ci \log_2 \tfrac{4}{3} + 4i + k \\
&\leq ci \log_2 i - (6+k)i + 4i + k && \text{(by choice of } c\text{)} \\
&= ci \log_2 i - (2+k)i + k \\
&\leq ci \log_2 i - (2+k) + k && (2 \leq i) \\
&\leq ci \log_2 i
\end{aligned}$$

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 techinque known as "divide and conquer". To solve a large instance of a problem, a divide and conquer techinque 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 soluton for the large instance.

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

$$T(n) = \begin{cases} K & \text{if } n \leq B \\ a_1 T(\lceil \frac{n}{b} \rceil) + a_2 T(\lfloor \frac{n}{b} \rfloor) + f(n) & \text{if } n > B \end{cases} \tag{1}$$

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.*