CSC 212: Data Structures and Abstractions
Spring 2018
University of Rhode Island

Weekly Problem Set #8

1 Recurrences

1. Find a closed-form equivalent of the following recurrences:

   (a) The Towers of Hanoi:

   $$T(0) = 0; T(n) = 2T(n-1) + 1$$

   $$\begin{aligned} T(n) &= 2T(n-1) + 1 \\ &= 2(2T(n-2) + 1) + 1 \\ &= 4T(n-2) + 3 \\ &= 4(2T(n-3) + 1) + 3 \\ &= 8T(n-3) + 7 \\ &= \dots \end{aligned} \tag{1}$$

   This pattern can be written as follows:

   $$T(n) = 2^k T(n-k) + (2^k - 1)$$

   Unrolling n times would yield: $T(n) = 2^n T(0) + (2^n - 1)$ Plugging in the base case $T(0) = 0$ gives us $T(n) = 2^n - 1$

   (b) The Merge Sort:

   $$T(1) = 1; T(n) = 2T(\frac{n}{2}) + n$$

   $$\begin{aligned} T(n) &= 2T(\frac{n}{2} + n) \\ &= 2(2T(\frac{n}{4}) + \frac{n}{2}) + n \\ &= 4T(\frac{n}{4}) + 2n \\ &= 8T(\frac{n}{8}) + 3n \\ &= \dots \end{aligned} \tag{2}$$

   This pattern can be written as follows:

   $$T(n) = 2^k T(\frac{n}{2}) + kn$$

   Becoming trivial when $\frac{n}{2^k} = 1$ or $k = \log_2 n$ Putting it all together:

   $$T(n) = nT(1) + n\log_2 n = n\log_2 n + n$$

(c) Generic:

$$T(0) = 1; T(n) = T(n-1) + 2^n$$

$$\begin{aligned}
T(n) &= T(n-1) + 2^n \\
T(n-1) &= T(n-2) + 2^{n-1} + 2^n \\
&= T(n-3) + 2^{n-2} + 2^{n-1} + 2^n \\
&= T(n-4) + 2^{n-3} + 2^{n-2} + 2^{n-1} + 2^n \\
&= \dots \\
&= T(n-k) + \sum_{i=n-k+1}^{n} (2^i) \\
&= T(n-n) + \sum_{i=n-n+1}^{n} (2^i) \\
&= 2^{n+1} - 1 \\
&= \Theta(2^n)
\end{aligned} \tag{3}$$

(d) Generic:

$$T(1) = 1; T(n) = T(\frac{n}{3}) + 1$$

$$\begin{aligned}
T(n) &= T(\frac{n}{3}) + 1 \\
T(\frac{n}{3}) &= T((\frac{n}{3})/3) + 1 + 1 = T(\frac{n}{9}) + 2 \\
T(\frac{n}{9}) &= T(\frac{n}{27}) + 3 \\
T(\frac{n}{27}) &= T(\frac{n}{81}) + 4 \\
&= T(\frac{n}{3^k}) + k \\
\text{Finding constants: } \frac{n}{3^k} &= 1 \\
n &= 3^k \\
k &= log_3 n \\
&= \sum_{i=1}^{k} 1 + \log_3 n \\
&= \Theta(\log_3 n)
\end{aligned} \tag{4}$$

2 Merge Sort

1. Determine the running-time of merge sort for a) sorted input; b) reverse-ordered input; c) random input; d) all identical input. Justify your answers.

   (a) Merge Sort is guaranteed $O(n \log n)$ for all cases. The natural variant supports $O(n)$ for already sorted inputs.

The following is considered optional.

1. Research and implement Tim Sort. A link about Tim Sort

2. Find a closed-form equivalent of the following recurrence:

$$f(1) = 3; f(n) = f(\frac{n}{2}) + 1$$