CSC 225 - Summer 2019

Recurrence Relations I

Bill Bird

Department of Computer Science
University of Victoria

May 29, 2019

Fibonacci Numbers (1)

Exercise: The Fibonacci sequence

$$1, \quad 1, \quad 2, \quad 3, \quad 5, \quad 8, \quad 13, \quad 21, \ldots$$

is defined by the recurrence

$$\begin{aligned} F(n) &= 1 && \text{if } n = 0 \\ &= 1 && \text{if } n = 1 \\ &= F(n-1) + F(n-2) && \text{if } n \geq 2 \end{aligned}$$

Prove that $F(n) \in O(2^n)$.

Fibonacci Numbers (2)

For many of the recurrences in this course, it is easier to find a closed form before proving any asymptotic relationships.

Recurrences with multiple recursive calls on the right hand side are not easy to solve (you may have seen solution methods in Math 122 or Math 222), and we will not cover techniques to solve them. However, we can still prove asymptotic relationships.

n	0	1	2	3	4	5
$F(n)$	1	1	2	3	5	8
2^n	1	2	4	8	16	32

We will prove that $F(n) \leq 2^n$ for all $n \geq 0$, which is sufficient to prove that $F(n) \in O(2^n)$.

Fibonacci Numbers (3)

$$\begin{aligned} F(n) &= 1 && \text{if } n = 0 \\ &= 1 && \text{if } n = 1 \\ &= F(n-1) + F(n-2) && \text{if } n \geq 2 \end{aligned}$$

Basis: We will use $n = 0$ and $n = 1$ as base cases.

- $F(0) = 1 \leq 2^0 = 1$
- $F(1) = 1 \leq 2^1 = 2$

For both base cases, $F(n) \leq 2^n$.

Since the recursive part of the recurrence refers to two different values of F, a strong hypothesis is necessary.

Induction Hypothesis: Suppose that $F(n) \leq 2^n$ for all n in the range $1, 2, \ldots, m$ for some $m \geq 0$.

Fibonacci Numbers (4)

$$\begin{aligned} F(n) &= 1 && \text{if } n = 0 \\ &= 1 && \text{if } n = 1 \\ &= F(n-1) + F(n-2) && \text{if } n \geq 2 \end{aligned}$$

Induction Step: Consider the value m+1. By the definition of the recurrence,

$$F(m+1) = F(m) + F(m-1).$$

Since the hypothesis covers both m and $m-1$, we can apply it to yield

$$F(m+1) \leq 2^m + 2^{m-1}$$

and since $2^m \geq 2^{m-1}$,

$$F(m+1) \leq 2^m + 2^{m-1} \leq 2^m + 2^m = 2^{m+1}$$

Therefore, by induction, $F(n) \leq 2^n$ for all $n \geq 0$.

Fibonacci Numbers (5)

It turns out that $F(n) \in \Theta(\varphi^n)$, where

$$\varphi = \frac{1 + \sqrt{5}}{2}.$$

We are not going to investigate the matter any further.

Merge Sort Analysis (1)

procedure MERGESORT(A)
    $n \leftarrow$ LENGTH(A)
    if $n = 1$ **then**
        //An array of size 1 is already sorted.
        return
    end if
    $n_1 \leftarrow \lfloor n/2 \rfloor$
    $n_2 \leftarrow n - n_1$
    Split A into two arrays A_1 (with size n_1) and A_2 (with size n_2).
    MERGESORT(A_1)
    MERGESORT(A_2)
    Merge the sorted A_1 and A_2 together into A.
end procedure

- Consider the following recurrence for merge sort:

$$\begin{aligned} T(n) &= 1 && \text{if } n = 1 \\ &= 2T(n/2) + n + 1 && \text{if } n \geq 2 \end{aligned}$$

- **Exercise**: Solve the recurrence to obtain a closed form. Assume that n is a power of 2.

Merge Sort Analysis (2)

```
procedure MergeSort(A)
    n ← Length(A)
    if n = 1 then
        //An array of size 1 is already sorted.
        return
    end if
    n1 ← ⌊n/2⌋
    n2 ← n − n1
    Split A into two arrays A1 (with size n1) and A2 (with size n2).
    MergeSort(A1)
    MergeSort(A2)
    Merge the sorted A1 and A2 together into A.
end procedure
```

- It is completely acceptable to find a closed form by 'guessing' using the first few values.
- Any closed form must be proven correct (usually by induction). It is not relevant how the closed form was derived if it is correct.

Merge Sort Analysis (3)

procedure MERGESORT(A)
    $n \leftarrow$ LENGTH(A)
    if $n = 1$ **then**
        //An array of size 1 is already sorted.
        return
    end if
    $n_1 \leftarrow \lfloor n/2 \rfloor$
    $n_2 \leftarrow n - n_1$
    Split A into two arrays A_1 (with size n_1) and A_2 (with size n_2).
    MERGESORT(A_1)
    MERGESORT(A_2)
    Merge the sorted A_1 and A_2 together into A.
end procedure

- In general, the method of **repeated substitution** is very effective at finding closed forms for the recurrences studied in this course.
- **Note**: Repeated substitution is a *heuristic device*, not a proof technique. It is essentially just a tool to help you guess the right solution.

Repeated Substitution (1)

Before we try to solve the merge sort recurrence, consider the example below.

Task: Find a closed form for the recurrence

$$\begin{aligned} T(n) &= 1 && \text{if } n = 0 \\ &= T(n-1) + n + 3 && \text{if } n \geq 1 \end{aligned}$$

Repeated Substitution (2)

$$
\begin{aligned}
T(n) &= 1 && \text{if } n = 0 \\
&= T(n-1) + n + 3 && \text{if } n \geq 1
\end{aligned}
$$

Step 1: Start with the recursive form.

$$T(n) = T(n-1) + n + 3$$

Step 2: Expand once and gather terms.

$$
\begin{aligned}
T(n) &= [T(n-2) + (n-1) + 3] + n + 3 \\
&= T(n-2) + (n-1) + n + 2 \cdot 3
\end{aligned}
$$

Step 3: Expand again and gather terms.

$$
\begin{aligned}
T(n) &= [T(n-3) + (n-2) + 3] + (n-1) + n + 2 \cdot 3 \\
&= T(n-3) + (n-2) + (n-1) + n + 3 \cdot 3
\end{aligned}
$$

Repeated Substitution (3)

$$
\begin{aligned}
T(n) &= 1 && \text{if } n = 0 \\
&= T(n-1) + n + 3 && \text{if } n \geq 1
\end{aligned}
$$

Step 3: Expand again and gather terms.

$$
\begin{aligned}
T(n) &= [T(n-3) + (n-2) + 3] + (n-1) + n + 2 \cdot 3 \\
&= T(n-3) + (n-2) + (n-1) + n + 3 \cdot 3
\end{aligned}
$$

Step 4, 5, 6, ... : Continue expanding until you observe a pattern.

...

Step i (for $i \geq 1$):

$$
\begin{aligned}
T(n) &= T(n-i) + (n-(i-1)) + \ldots + (n-1) + n + 3i \\
&= T(n-i) + \left(\sum_{j=n-(i-1)}^{n} j \right) + 3i
\end{aligned}
$$

Repeated Substitution (4)

After finding a representation of the form

$$T(n) = T(n-i) + \left(\sum_{j=n-(i-1)}^{n} j \right) + 3i$$

The term $T(n-i)$ can be removed by choosing i such that $T(n-i)$ is the base case (in this case, the base case is $n = 0$ so we take $i = n$):

$$\begin{aligned} T(n) &= T(n-i) + \left(\sum_{j=n-(i-1)}^{n} j \right) + 3i \\ &= T(n-n) + \left(\sum_{j=n-(n-1)}^{n} j \right) + 3n \\ &= T(0) + \left(\sum_{j=1}^{n} j \right) + 3n \end{aligned}$$

Repeated Substitution (5)

Finally, we replace $T(0)$ with the base case value from the definition.

$$T(n) = T(0) + \left(\sum_{j=1}^{n} j\right) + 3n$$

$$= 1 + \left(\sum_{j=1}^{n} j\right) + 3n$$

Since the sum $\sum_{j=1}^{n} j = 1 + 2 + \ldots + n$ is equal to $\frac{1}{2}n(n+1)$, our finished closed form is

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

$$= \frac{1}{2}n^2 + \frac{7}{2}n + 1$$

We can now prove the closed form correct by induction.

Repeated Substitution (6)

Finally, we replace $T(0)$ with the base case value from the definition.

$$T(n) = T(0) + \left(\sum_{j=1}^{n} j\right) + 3n$$

$$= 1 + \left(\sum_{j=1}^{n} j\right) + 3n$$

Since the sum $\sum_{j=1}^{n} j = 1 + 2 + \ldots + n$ is equal to $\frac{1}{2}n(n+1)$, our finished closed form is

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

$$= \frac{1}{2}n^2 + \frac{7}{2}n + 1$$

We can now prove the closed form correct by induction.

Merge Sort Recurrence: Method 1 (1)

Task: Solve the recurrence

$$\begin{aligned} T(n) &= 1 && \text{if } n = 1 \\ &= 2T(n/2) + n + 1 && \text{if } n \geq 2 \end{aligned}$$

You may assume that n is a power of 2.

Merge Sort Recurrence: Method 1 (2)

$$
\begin{aligned}
T(n) &= 1 && \text{if } n = 1 \\
&= 2T(n/2) + n + 1 && \text{if } n \geq 2
\end{aligned}
$$

Step 1: Start with the recursive form.

$$T(n) = 2T(n/2) + n + 1$$

Step 2: Expand once and gather terms.

$$
\begin{aligned}
T(n) &= 2\left[2T(n/4) + \frac{n}{2} + 1\right] + n + 1 \\
&= 4T(n/4) + \frac{2n}{2} + 2 + n + 1 \\
&= 4T(n/4) + 2n + 3
\end{aligned}
$$

Merge Sort Recurrence: Method 1 (3)

$$\begin{aligned} T(n) &= 1 && \text{if } n = 1 \\ &= 2T(n/2) + n + 1 && \text{if } n \geq 2 \end{aligned}$$

(end of step 2)

$$T(n) = 4T(n/4) + 2n + 3$$

Step 3: Expand again and gather terms.

$$\begin{aligned} T(n) &= 4\left[2T(n/8) + \frac{n}{4} + 1\right] + 2n + 3 \\ &= 8T(n/8) + 4\frac{n}{4} + 4 + 2n + 3 \\ &= 8T(n/8) + 3n + 7 \end{aligned}$$

Merge Sort Recurrence: Method 1 (4)

$$\begin{aligned} T(n) &= 1 && \text{if } n = 1 \\ &= 2T(n/2) + n + 1 && \text{if } n \geq 2 \end{aligned}$$

(end of step 3)

$$T(n) = 8T(n/8) + 3n + 7$$

Step 4: Expand again and gather terms.

$$\begin{aligned} T(n) &= 8\left[2T(n/16) + \frac{n}{8} + 1\right] + 3n + 7 \\ &= 16T(n/16) + 8\frac{n}{8} + 8 + 3n + 7 \\ &= 16T(n/16) + 4n + 15 \end{aligned}$$

Merge Sort Recurrence: Method 1 (5)

$$\begin{aligned} T(n) &= 1 && \text{if } n = 1 \\ &= 2T(n/2) + n + 1 && \text{if } n \geq 2 \end{aligned}$$

Step 4: Expand again and gather terms.

$$\begin{aligned} T(n) &= 8\left[2T(n/16) + \frac{n}{8} + 1\right] + 3n + 7 \\ &= 16T(n/16) + 8\frac{n}{8} + 8 + 3n + 7 \\ &= 16T(n/16) + 4n + 15 \end{aligned}$$

Step i:

$$T(n) = 2^i T(n/2^i) + i \cdot n + (2^i - 1)$$

Merge Sort Recurrence: Method 1 (6)

$$\begin{aligned} T(n) &= 1 && \text{if } n = 1 \\ &= 2T(n/2) + n + 1 && \text{if } n \geq 2 \end{aligned}$$

Step i:

$$T(n) = 2^i T(n/2^i) + i \cdot n + (2^i - 1)$$

To eliminate the recursive term $T(n/2^i)$, we choose $i = \log_2 n$, since $n/2^{\log_2 n} = 1$.

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

Merge Sort Recurrence: Method 2 (1)

Task: Solve the recurrence

$$\begin{aligned} T(n) &= 1 && \text{if } n = 1 \\ &= 2T(n/2) + n + 1 && \text{if } n \geq 2 \end{aligned}$$

You may assume that n is a power of 2.

Merge Sort Recurrence: Method 2 (2)

$$\begin{aligned} T(n) &= 1 && \text{if } n = 1 \\ &= 2T(n/2) + n + 1 && \text{if } n \geq 2 \end{aligned}$$

Since n is a power of 2, it may be easier to write

$$n = 2^k$$

and then solve the recurrence in terms of k.

Step 0: Rewrite the recurrence for $n = 2^k$.

$$\begin{aligned} T(2^k) &= 1 && \text{if } k = 0 \\ &= 2T(2^{k-1}) + 2^k + 1 && \text{if } k \geq 1 \end{aligned}$$

Merge Sort Recurrence: Method 2 (3)

$$\begin{aligned} T(2^k) &= 1 && \text{if } k = 0 \\ &= 2T(2^{k-1}) + 2^k + 1 && \text{if } k \geq 1 \end{aligned}$$

Step 1: Start with the recursive form.

$$T(2^k) = 2T(2^{k-1}) + 2^k + 1$$

Step 2: Expand and gather terms.

$$\begin{aligned} T(2^k) &= 2\left[2T(2^{k-2}) + 2^{k-1} + 1\right] + 2^k + 1 \\ &= 2^2 T(2^{k-2}) + 2 \cdot 2^{k-1} + 2 + 2^k + 1 \\ &= 2^2 T(2^{k-2}) + 2^k + 2 + 2^k + 1 \\ &= 2^2 T(2^{k-2}) + 2 \cdot 2^k + 2 + 1 \end{aligned}$$

Merge Sort Recurrence: Method 2 (4)

$$
\begin{aligned}
T(2^k) &= 1 && \text{if } k = 0 \\
&= 2T(2^{k-1}) + 2^k + 1 && \text{if } k \geq 1
\end{aligned}
$$

(end of step 2)

$$= 2^2 T(2^{k-2}) + 2 \cdot 2^k + 2 + 1$$

Step 3: Expand and gather terms.

$$
\begin{aligned}
T(2^k) &= 2^2 \left[2T(2^{k-3}) + 2^{k-2} + 1\right] + 2 \cdot 2^k + 2 + 1 \\
&= 2^3 T(2^{k-3}) + 2^2 2^{k-2} + 2^2 + 2 \cdot 2^k + 2 + 1 \\
&= 2^3 T(2^{k-3}) + 3 \cdot 2^k + 2^2 + 2 + 1
\end{aligned}
$$

Merge Sort Recurrence: Method 2 (5)

$$
\begin{aligned}
T(2^k) &= 1 && \text{if } k = 0 \\
&= 2T(2^{k-1}) + 2^k + 1 && \text{if } k \geq 1
\end{aligned}
$$

(end of step 3)

$$= 2^3 T(2^{k-3}) + 3 \cdot 2^k + 2^2 + 2 + 1$$

Step i:

$$
\begin{aligned}
T(2^k) &= 2^i T(2^{k-i}) + i \cdot 2^k + \left[2^{i-1} + \ldots + 2^2 + 2 + 1\right] \\
&= 2^i T(2^{k-i}) + i \cdot 2^k + \sum_{j=0}^{i-1} 2^j
\end{aligned}
$$

Merge Sort Recurrence: Method 2 (6)

$$\begin{aligned} T(2^k) &= 1 && \text{if } k = 0 \\ &= 2T(2^{k-1}) + 2^k + 1 && \text{if } k \geq 1 \end{aligned}$$

Step i:

$$T(2^k) = 2^i T(2^{k-i}) + i \cdot 2^k + \sum_{j=0}^{i-1} 2^j$$

Since the base case is $k = 0$, we take $i = k$, giving

$$T(2^k) = 2^k T(2^{k-k}) + k \cdot 2^k + \sum_{j=0}^{k-1} 2^j$$

$$= 2^k \cdot 1 + k \cdot 2^k + \sum_{j=0}^{k-1} 2^j$$

Merge Sort Recurrence: Method 2 (7)

$$\begin{aligned} T(2^k) &= 1 && \text{if } k = 0 \\ &= 2T(2^{k-1}) + 2^k + 1 && \text{if } k \geq 1 \end{aligned}$$

$$T(2^k) = 2^k \cdot 1 + k \cdot 2^k + \sum_{j=0}^{k-1} 2^j$$

We can apply the identity

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$$

to the closed form to yield

$$T(2^k) = 2^k \cdot 1 + k \cdot 2^k + 2^k - 1$$

Merge Sort Recurrence: Method 2 (8)

$$\begin{aligned} T(2^k) &= 1 && \text{if } k = 0 \\ &= 2T(2^{k-1}) + 2^k + 1 && \text{if } k \geq 1 \end{aligned}$$

$$T(2^k) = 2^k \cdot 1 + k \cdot 2^k + 2^k - 1$$

Finally, we can get an answer in terms of n by setting $k = \log_2 n$ (since we assumed that $n = 2^k$):

$$\begin{aligned} T(n) &= 2^{\log_2 n} \cdot 1 + (\log_2 n) \cdot 2^{\log_2 n} + 2^{\log_2 n} - 1 \\ &= n + (\log_2 n) \cdot n + n - 1 \\ &= n \log_2 n + 2n - 1 \end{aligned}$$