Solving Recurrences
An important hammer

C. Seshadhri

University of California, Santa Cruz
sesh@ucsc.edu

April 6, 2016
Mergesort

**Merge-Sort**\((A)\)

1. \(n = A.length\)
2. **if** \(n == 1\), **return** \(A\)
3. \(L = A[1 \ldots n/2]\) // Technically, use \([n/2]\)
4. \(R = A[n/2 + 1 \ldots n]\)
5. \(L_{\text{sort}} = \text{Merge-Sort}(L)\)
6. \(R_{\text{sort}} = \text{Merge-Sort}(R)\)
7. **return** \(\text{Merge}(L_{\text{sort}}, R_{\text{sort}})\) // Process to merge two sorted arrays
The time complexity of Mergesort

As usual, let \( T(n) \) denote the worst-case time-complexity of Mergesort. Which of the following is true? Let \( c \) be some constant, and following should hold for all \( n > 1 \).

- (R) \( T(n) \leq T(n/2) + cn \)
- (B) \( T(n) \leq 2T(n/2) + cn \)
- (G) \( T(n) \leq 2T(n/2) \)
- (O) \( T(n) \leq T(n - 1) + cn \)
The time complexity of Mergesort

As usual, let $T(n)$ denote the worst-case time-complexity of Mergesort. Which of the following is true? Let $c$ be some constant, and following should hold for all $n > 1$.

- (R) $T(n) \leq T(n/2) + cn$
- (B) $T(n) \leq 2T(n/2) + cn$
- (G) $T(n) \leq 2T(n/2)$
- (O) $T(n) \leq T(n-1) + cn$
Mergesort

**MERGE-SORT(A)**

1. \( n = A.length \)
2. \( \text{if } n == 1, \text{ return } A \)
3. \( L = A[1..n/2] \) // Technically, use \( \lfloor n/2 \rfloor \)
4. \( R = A[n/2 + 1, \ldots n] \)
5. \( L_{sort} = \text{MERGE-SORT}(L) \quad T(n/2) \)
6. \( R_{sort} = \text{MERGE-SORT}(R) \quad T(n/2) \)
7. \( \text{return } \text{MERGE}(L_{sort}, R_{sort}) \) // Process to merge two sorted arrays

All additional run time is \( \Theta(n) \), so it’s at most \( cn \).
An approach for Mergesort analysis

Find the “best function” that satisfies the recurrence

\[ T(n) \leq 2T(n/2) + cn \]

\[ T(1) \leq c \]
Consider the Divide-Conquer-Combine paradigm

One gets recurrences of the form

\[ T(n) \leq aT\left(\frac{n}{b}\right) + f(n) \]

where \( a \geq 1 \), \( b > 1 \), and \( f(n) \) is the time complexity of the combine step.

Try to find the “least” function \( T(n) \) that satisfies this
The Broad Approach

- **Draw a Recursion Tree:** Like we did yesterday. Add up the work. Cut corners to get a good guess
- **Use Induction:** To finally verify your guess
An approach for Mergesort analysis

Find the “best function” that satisfies the recurrence

\[ T(n) \leq 2T(n/2) + cn \]

\[ T(1) \leq c \]
The Recursion Tree

- Associate a “cost” with each node, which is the “combine” run time
- Sum up costs over a level. Get an expression
- Sum up previous expression over all levels
The recursion tree

\[ n \]

\[ \frac{n}{2} \]

\[ \frac{n}{4} \]

\[ \frac{n}{8} \]

\[ \frac{n}{8} \]

\[ \frac{n}{8} \]

\[ \frac{n}{8} \]

\[ \frac{n}{8} \]

\[ \frac{n}{8} \]

\[ \frac{n}{8} \]
The recursion tree

\[ \begin{array}{c}
 n \\
 n/2 \\
 n/4 \\
 n/8 \\
 \text{cn/8} \\
 \text{cn/8} \\
 \text{cn/8} \\
 \text{cn/8} \\
 n/4 \\
 n/8 \\
 \text{cn/8} \\
 \text{cn/8} \\
 n/2 \\
 \text{cn/2} \\
 \text{cn/2} \\
 n/4 \\
 n/8 \\
 \text{cn/8} \\
 \text{cn/8} \\
 cn/2 \\
 \text{cn/2} \\
 cn/2 \\
 n \\
 \text{cn} \\
 \end{array} \]
The recursion tree

\[
\begin{align*}
\text{n} & \\
\frac{n}{2} & \quad \frac{n}{2} \\
\frac{n}{4} & \quad \frac{n}{4} \\
\frac{n}{8} & \quad \frac{n}{8} \\
\frac{n}{8} & \quad \frac{n}{8} \\
\frac{n}{8} & \quad \frac{n}{8} \\
\frac{n}{8} & \quad \frac{n}{8} \\
\frac{n}{8} & \quad \frac{n}{8} \\
\frac{n}{8} & \quad \frac{n}{8} \\
\frac{n}{8} & \quad \frac{n}{8} \\
\frac{n}{8} & \quad \frac{n}{8} \\
\frac{n}{8} & \quad \frac{n}{8} \\
\frac{n}{8} & \quad \frac{n}{8} \\
\frac{n}{8} & \quad \frac{n}{8} \\
\end{align*}
\]
An approach for Mergesort analysis

Find the “best function” that satisfies the recurrence

\[ T(n) \leq 2T(n/2) + cn \]

Thus, our answer is \( T(n) \leq cn \log_2 n \)
Mergesort

**Merge-Sort**(A)

1. \( n = A.\text{length} \)
2. \textbf{if} \( n == 1 \), \textbf{return} \( A \)
3. \( L = A[1..n/3] \)
4. \( R = A[n/3 + 1,..n] \)
5. \( L_{\text{sort}} = \text{Merge-Sort}(L) \)
6. \( R_{\text{sort}} = \text{Merge-Sort}(R) \)
7. \textbf{return} \( \text{Merge}(L_{\text{sort}}, R_{\text{sort}}) \) \quad // Process to merge two sorted arrays
Let $T(n)$ denote the worst-case time-complexity of modified Mergesort. Which of the following is true? Let $c$ be some constant, and following should hold for all $n > 1$.

- (R) $T(n) \leq 2T(n/2) + cn$
- (B) $T(n) \leq 2T(n/3) + cn$
- (G) $T(n) \leq 2T(2n/3) + cn$
- (O) $T(n) \leq T(n/3) + T(2n/3) + cn$
Let $T(n)$ denote the worst-case time-complexity of modified Mergesort. Which of the following is true? Let $c$ be some constant, and following should hold for all $n > 1$.

- (R) $T(n) \leq 2T(n/2) + cn$
- (B) $T(n) \leq 2T(n/3) + cn$
- (G) $T(n) \leq 2T(2n/3) + cn$
- (O) $T(n) \leq T(n/3) + T(2n/3) + cn$
The recursion tree

\[ \begin{align*}
  n \\
  \frac{n}{3} & \to \frac{2n}{9} \\
  \frac{n}{9} & \to \frac{2n}{27} \\
  \frac{n}{27} & \\
 \end{align*} \]
The recursion tree

\[ \begin{align*}
\text{n} & \quad \text{cn} \\
\frac{n}{3} & \quad \frac{cn}{3} \\
\frac{n}{9} & \quad \frac{cn}{9} \\
\frac{n}{27} & \quad \frac{cn}{27} \\
\frac{2n}{9} & \quad 2\frac{cn}{9} \\
\frac{2n}{27} & \quad 2\frac{cn}{27} \\
\frac{2n}{27} & \quad 2\frac{cn}{27} \\
\frac{4n}{27} & \quad 4\frac{cn}{27} \\
\frac{4n}{27} & \quad 4\frac{cn}{27} \\
\frac{2n}{27} & \quad 2\frac{cn}{27} \\
\frac{4n}{27} & \quad 4\frac{cn}{27} \\
\frac{4n}{27} & \quad 4\frac{cn}{27} \\
\frac{4n}{27} & \quad 4\frac{cn}{27} \\
\frac{8n}{27} & \quad 8\frac{cn}{27} \\
\end{align*} \]
What is the total “cost” of level $i$? Give some upper bound

- (R) $cn$
- (B) $c(2/3)^i n$
- (G) $c(3/2)^i n$
- (O) $c2^i n$
What is the total “cost” of level $i$? Give some upper bound

- (R) $cn$
- (B) $c(2/3)^i n$
- (G) $c(3/2)^i n$
- (O) $c2^i n$
Is the cost exactly $cn$?

- (R) Yes
- (B) No
Question!

Is the cost exactly $cn$?

- (R) Yes
- (B) No

Some of the tree paths end “earlier” than others. Thus, $cn$ is only an upper bound.
What is the maximum depth of the recursion tree?

- (R) $O(\log_2 n)$
- (B) $O(n)$
- (G) Neither of these
What is the maximum depth of the recursion tree?

- (R) $O(\log_2 n)$
- (B) $O(n)$
- (G) Neither of these

The longest path is the one corresponding to a $(2/3)$-factor reduction in each step. At depth $i$, input size is $(2/3)^i n$. Thus, longest path has length $\log_{3/2} n$. 

What is the maximum depth of the recursion tree?

- (R) $O(\log_2 n)$
- (B) $O(n)$
- (G) Neither of these

The longest path is the one corresponding to a $(2/3)$-factor reduction in each step. At depth $i$, input size is $(2/3)^i n$. Thus, longest path has length $\log_{3/2} n$.

By the properties of log:

$$\log_{3/2} n = \frac{\log_2 n}{\log_2 (3/2)} = O(\log_2 n)$$
Different version of MergeSort

Standard Mergesort and Mergesort with 1/3 - 2/3 split both have time complexity $O(n \log_2 n)$
Another recurrence

Find the “best function” that satisfies the recurrence

\[ T(n) \leq 2T(n/2) + c\sqrt{n} \]

\[ T(1) \leq c \]
The recursion tree

```
        n
       / \
     n/2  n/2
    /    / \
 n/4  n/4 n/4 n/4
  /  /  /  /  /  /  /
n/8 n/8 n/8 n/8 n/8 n/8 n/8 n/8
```

C. Seshadhri (UCSC)  CMPS101  25 / 29
The recursion tree

\[ n \quad c\sqrt{n} \]

\[ \frac{n}{2} \quad c\sqrt{\frac{n}{2}} \]

\[ \frac{n}{4} \quad c\sqrt{\frac{n}{4}} \]

\[ \frac{n}{8} \quad c\sqrt{\frac{n}{8}} \]

C. Seshadhri (UCSC)
Cost of level

What is the total "cost" of level $i$? Give some upper bound

- (R) $cn$
- (B) $c\sqrt{n}$
- (G) $c\sqrt{2^i n}$
- (O) $c\sqrt{n/2^i}$
Cost of level

What is the total “cost” of level $i$? Give some upper bound:

- (R) $cn$
- (B) $c\sqrt{n}$
- (G) $c\sqrt{2^i n}$
- (O) $c\sqrt{n/2^i}$

The cost of a node at depth $i$ is $c\sqrt{n/2^i}$. The total number of nodes at depth $i$ is $2^i$.

\[
2^i \sqrt{n/2^i} = \sqrt{2^i n}
\]
Finally...

What is the total cost?

- (R) $O(n)$
- (B) $O(n \log_2 n)$
- (G) $O(n\sqrt{n})$
- (O) $O(n^2)$
Finally...

What is the total cost?

- (R) $O(n)$
- (B) $O(n \log_2 n)$
- (G) $O(n \sqrt{n})$
- (O) $O(n^2)$

\[
\sum_{i=0}^{\log_2 n} \sqrt{2^i n} = \sqrt{n} \sum_{i=0}^{\log_2 n} (\sqrt{2})^i \\
= \sqrt{n}(\sqrt{2}^{\log_2 n+1} - 1)/(\sqrt{2} - 1) \\
= \sqrt{n}(\sqrt{2}\sqrt{n} - 1)(\sqrt{2} - 1) = O(n)
\]