Lecture 2: Recurrences

Viet Tung Hoang

The slides are loosely based on those of Dr Kevin Wayne, Princeton University and Prof. Mary Wootters, Stanford University.
Agenda

1. Divide and Conquer

2. Tower of Hanoi
Merge Sort

Recursive magic!

Recursive magic!
Merge Sort

Recursive magic!

Merge
Merge Sort Implementation

```plaintext
procedure MergeSort(A[1 : n])
if n = 1 then return A // Base case
MergeSort(A[1 : n/2]) // Sort left half
MergeSort(A[n/2 + 1 : n]) // Sort right half
B ← Merge(A[1 : n/2], A[n/2 + 1 : n])
return B
```
Running-time Analysis

\(T(n)\): maximum number of steps to Merge-Sort an array of size \(n\)

**Merge-Sort recurrence:**

\[
T(n) = \begin{cases} 
1 & \text{if } n = 1 \\
2T(n/2) + n & \text{otherwise}
\end{cases}
\]
Recursion Tree

\[ T(n) = 2T(n/2) + n \text{ and } T(1) = 1 \]

Assume that \( n \) is a power of 2

\[ \log_2(n) + 1 \]

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

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

\[ 2^i \cdot \frac{n}{2^i} = n \]

\[ n \]

\[ T(n) = n(\log_2(n) + 1) \]
Another Recurrence

Assume that $n$ is a power of 2

$$T(n) = \begin{cases} 
1 & \text{if } n = 1 \\
2T(n/2) + \log_2(n) & \text{otherwise}
\end{cases}$$
Recursion Tree

\[ T(n) = 2T(n/2) + \log_2(n) \text{ and } T(1) = 1 \]

Assume that \( n = 2^k \)

\[
\begin{align*}
  k & \quad k + 1 \\
  2(k - 1) & \quad 4 \cdot (k - 2) \\
  2^i \cdot (k - i) & \quad 2^k \\
  T(n) = ?
\end{align*}
\]
Agenda

1. Divide and Conquer

2. Tower of Hanoi
An Example

How to move the disks to another rod, such that a big disk never sits on top of a small one?
A Demo
A Demo
A Demo
A Demo
A Demo
A Demo
A Demo

Totally there are 7 moves
The General Case

There are now $n$ disks. Find $T_n$, the optimal number of moves.
The Recursion

Move the top $n - 1$ disks for $T_{n-1}$ moves
The Recursion

Another move for the biggest disk
The Recursion

Move the top $n - 1$ disks on top of the biggest, for $T_{n-1}$ moves

$$T_n = 2T_{n-1} + 1$$