Recurrences II

COMS10017 - Algorithms 1

Dr Christian Konrad

Recursion Tree Method

Recursion Tree:

Recursion Tree Method

Recursion Tree:

- Each node represents cost of single subproblem

Recursion Tree Method

Recursion Tree:

- Each node represents cost of single subproblem
- Recursive invocations become children of a node

Recursion Tree Method

Recursion Tree:

- Each node represents cost of single subproblem
- Recursive invocations become children of a node

Example

Recursion Tree Method

Recursion Tree:

- Each node represents cost of single subproblem
- Recursive invocations become children of a node

Example

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Recursion Tree Method

Recursion Tree:

- Each node represents cost of single subproblem
- Recursive invocations become children of a node

Example

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

$T(64)$

Recursion Tree:

- Each node represents cost of single subproblem
- Recursive invocations become children of a node

Example

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

$$T(64) = 2T(16) + 32$$

Recursion Tree Method

Recursion Tree:

- Each node represents cost of single subproblem
- Recursive invocations become children of a node

Example

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

$$T(64) \;=\; 2T(16) + 32 = 2(2T(4) + 8) + 32$$

Recursion Tree Method

Recursion Tree:

- Each node represents cost of single subproblem
- Recursive invocations become children of a node

Example

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

$$\begin{aligned} T(64) &= 2T(16) + 32 = 2(2T(4) + 8) + 32 \\ &= 2(2(2T(1) + 2) + 8) + 32 \end{aligned}$$

Recursion Tree Method

Recursion Tree:

- Each node represents cost of single subproblem
- Recursive invocations become children of a node

Example

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

$$\begin{aligned} T(64) &= 2T(16) + 32 = 2(2T(4) + 8) + 32 \\ &= 2(2(2T(1) + 2) + 8) + 32 \\ &= 2(2(2 \cdot 1 + 2) + 8) + 32 \end{aligned}$$

Recursion Tree Method

Recursion Tree:

- Each node represents cost of single subproblem
- Recursive invocations become children of a node

Example

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

$$\begin{aligned} T(64) &= 2T(16) + 32 = 2(2T(4) + 8) + 32 \\ &= 2(2(2T(1) + 2) + 8) + 32 \\ &= 2(2(2 \cdot 1 + 2) + 8) + 32 = 64 \end{aligned}$$

Example

Example

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + \underbrace{n/2}_{\text{cost of subproblem}}$$

Example

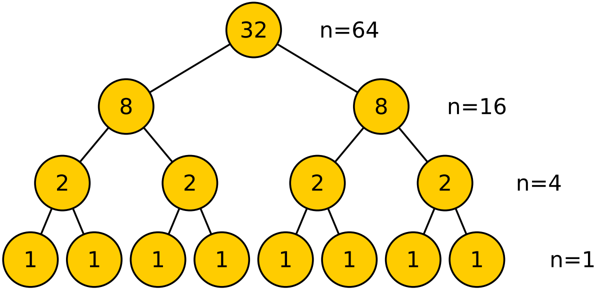
$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + \underbrace{n/2}_{\text{cost of subproblem}}$$

Recursion Tree for $n = 64$:

Example

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + \underbrace{n/2}_{\text{cost of subproblem}}$$

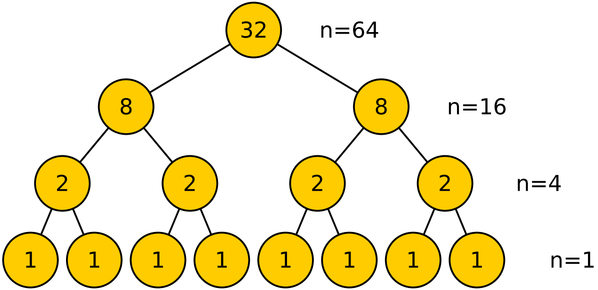
Recursion Tree for $n = 64$:



Example

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + \underbrace{n/2}_{\text{cost of subproblem}}$$

Recursion Tree for $n = 64$:



Sum of values assigned to nodes equals $T(64)$

Obtaining a Good Guess for Solution

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Obtaining a Good Guess for Solution

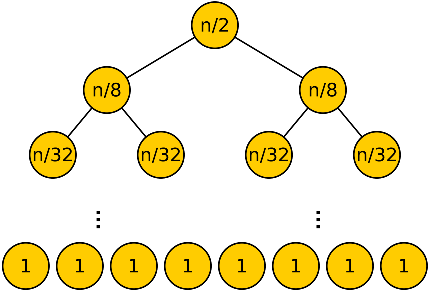
$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Draw Recursion Tree for general n (Observe: we ignore $\lfloor . \rfloor$)

Obtaining a Good Guess for Solution

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

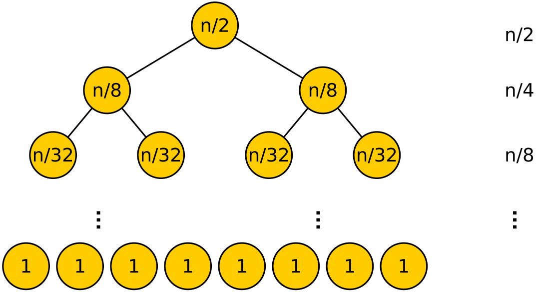
Draw Recursion Tree for general n (Observe: we ignore $\lfloor . \rfloor$)



Obtaining a Good Guess for Solution

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

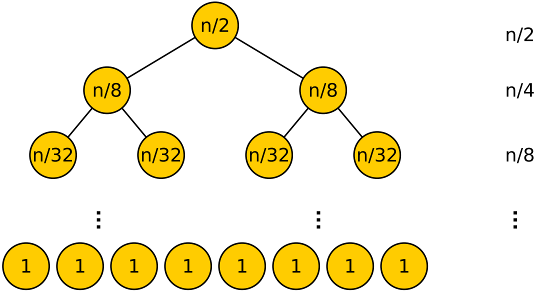
Draw Recursion Tree for general n (Observe: we ignore $\lfloor . \rfloor$)



Obtaining a Good Guess for Solution

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Draw Recursion Tree for general n (Observe: we ignore $\lfloor . \rfloor$)



Sum of Nodes in Level i**:** $\frac{n}{2^i}$ (except the last level)

Obtaining a Good Guess for Solution (2)

Number of Levels: ℓ

Obtaining a Good Guess for Solution (2)

Number of Levels: ℓ

- We have $\frac{n}{4^{\ell-1}} \approx 1$

Obtaining a Good Guess for Solution (2)

Number of Levels: ℓ

- We have $\frac{n}{4^{\ell-1}} \approx 1$
- $\ell = \log_4(n) + 1$

Obtaining a Good Guess for Solution (2)

Number of Levels: ℓ

- We have $\frac{n}{4^{\ell-1}} \approx 1$
- $\ell = \log_4(n) + 1$

Cost on last Level: $=$ number of nodes on last level

Obtaining a Good Guess for Solution (2)

Number of Levels: ℓ

- We have $\frac{n}{4^{\ell-1}} \approx 1$
- $\ell = \log_4(n) + 1$

Cost on last Level: $=$ number of nodes on last level

$$\approx 2^{\log_4(n)}$$

Obtaining a Good Guess for Solution (2)

Number of Levels: ℓ

- We have $\frac{n}{4^{\ell-1}} \approx 1$
- $\ell = \log_4(n) + 1$

Cost on last Level: $=$ number of nodes on last level

$$\approx 2^{\log_4(n)} = 2^{\frac{\log n}{\log 4}}$$

Obtaining a Good Guess for Solution (2)

Number of Levels: ℓ

- We have $\frac{n}{4^{\ell-1}} \approx 1$
- $\ell = \log_4(n) + 1$

Cost on last Level: $=$ number of nodes on last level

$$\approx 2^{\log_4(n)} = 2^{\frac{\log n}{\log 4}} = 2^{\log(n)/2}$$

Obtaining a Good Guess for Solution (2)

Number of Levels: ℓ

- We have $\frac{n}{4^{\ell-1}} \approx 1$
- $\ell = \log_4(n) + 1$

Cost on last Level: $=$ number of nodes on last level

$$\approx 2^{\log_4(n)} = 2^{\frac{\log n}{\log 4}} = 2^{\log(n)/2} = n^{\frac{1}{2}}$$

Obtaining a Good Guess for Solution (2)

Number of Levels: ℓ

- We have $\frac{n}{4^{\ell-1}} \approx 1$
- $\ell = \log_4(n) + 1$

Cost on last Level: $=$ number of nodes on last level

$$\approx 2^{\log_4(n)} = 2^{\frac{\log n}{\log 4}} = 2^{\log(n)/2} = n^{\frac{1}{2}} = \sqrt{n} \ .$$

Our Guess:

Obtaining a Good Guess for Solution (2)

Number of Levels: ℓ

- We have $\frac{n}{4^{\ell-1}} \approx 1$
- $\ell = \log_4(n) + 1$

Cost on last Level: = number of nodes on last level

$$\approx 2^{\log_4(n)} = 2^{\frac{\log n}{\log 4}} = 2^{\log(n)/2} = n^{\frac{1}{2}} = \sqrt{n} \ .$$

Our Guess:

$$\left(\sum_{i=1}^{\log_4(n)} \frac{n}{2^i} \right) + \sqrt{n}$$

Obtaining a Good Guess for Solution (2)

Number of Levels: ℓ

- We have $\frac{n}{4^{\ell-1}} \approx 1$
- $\ell = \log_4(n) + 1$

Cost on last Level: $=$ number of nodes on last level

$$\approx 2^{\log_4(n)} = 2^{\frac{\log n}{\log 4}} = 2^{\log(n)/2} = n^{\frac{1}{2}} = \sqrt{n} \ .$$

Our Guess:

$$\left(\sum_{i=1}^{\log_4(n)} \frac{n}{2^i} \right) + \sqrt{n} = \left(n \cdot \underbrace{\sum_{i=1}^{\log_4(n)} \frac{1}{2^i}}_{\text{geom. series}} \right) + \sqrt{n}$$

Obtaining a Good Guess for Solution (2)

Number of Levels: ℓ

- We have $\frac{n}{4^{\ell-1}} \approx 1$
- $\ell = \log_4(n) + 1$

Cost on last Level: $=$ number of nodes on last level

$$\approx 2^{\log_4(n)} = 2^{\frac{\log n}{\log 4}} = 2^{\log(n)/2} = n^{\frac{1}{2}} = \sqrt{n} \, .$$

Our Guess:

$$\left(\sum_{i=1}^{\log_4(n)} \frac{n}{2^i} \right) + \sqrt{n} = \left(n \cdot \underbrace{\sum_{i=1}^{\log_4(n)} \frac{1}{2^i}}_{\text{geom. series}} \right) + \sqrt{n} = n \cdot O(1) + \sqrt{n}$$

Obtaining a Good Guess for Solution (2)

Number of Levels: ℓ

- We have $\frac{n}{4^{\ell-1}} \approx 1$
- $\ell = \log_4(n) + 1$

Cost on last Level: $=$ number of nodes on last level

$$\approx 2^{\log_4(n)} = 2^{\frac{\log n}{\log 4}} = 2^{\log(n)/2} = n^{\frac{1}{2}} = \sqrt{n} \ .$$

Our Guess:

$$\left(\sum_{i=1}^{\log_4(n)} \frac{n}{2^i} \right) + \sqrt{n} = \left(n \cdot \underbrace{\sum_{i=1}^{\log_4(n)} \frac{1}{2^i}}_{\text{geom. series}} \right) + \sqrt{n} = n \cdot O(1) + \sqrt{n} = O(n) \ .$$

Obtaining a Good Guess for Solution (2)

Number of Levels: ℓ

- We have $\frac{n}{4^{\ell-1}} \approx 1$
- $\ell = \log_4(n) + 1$

Cost on last Level: $=$ number of nodes on last level

$$\approx 2^{\log_4(n)} = 2^{\frac{\log n}{\log 4}} = 2^{\log(n)/2} = n^{\frac{1}{2}} = \sqrt{n} .$$

Our Guess:

$$\left(\sum_{i=1}^{\log_4(n)} \frac{n}{2^i} \right) + \sqrt{n} = \left(n \cdot \underbrace{\sum_{i=1}^{\log_4(n)} \frac{1}{2^i}}_{\text{geom. series}} \right) + \sqrt{n} = n \cdot O(1) + \sqrt{n} = O(n) .$$

Use substitution method to prove that guess is correct!

Verification via Substitution Method

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Verification via Substitution Method

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Our Guess: $T(n) \leq c \cdot n$

Verification via Substitution Method

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Our Guess: $T(n) \leq c \cdot n$

Substitute into the Recurrence:

Verification via Substitution Method

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Our Guess: $T(n) \leq c \cdot n$

Substitute into the Recurrence:

$$T(n)$$

Verification via Substitution Method

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Our Guess: $T(n) \leq c \cdot n$

Substitute into the Recurrence:

$$T(n) \;=\; 2T(\lfloor n/4 \rfloor) + n/2$$

Verification via Substitution Method

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Our Guess: $T(n) \leq c \cdot n$

Substitute into the Recurrence:

$$T(n) = 2T(\lfloor n/4 \rfloor) + n/2 \leq 2c\lfloor \frac{n}{4} \rfloor + \frac{n}{2}$$

Verification via Substitution Method

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Our Guess: $T(n) \leq c \cdot n$

Substitute into the Recurrence:

$$T(n) \;=\; 2T(\lfloor n/4 \rfloor) + n/2 \leq 2c\lfloor \frac{n}{4} \rfloor + \frac{n}{2} \leq n\frac{c+1}{2} \leq c \cdot n \;,$$

Verification via Substitution Method

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Our Guess: $T(n) \leq c \cdot n$

Substitute into the Recurrence:

$$T(n) \;=\; 2T(\lfloor n/4 \rfloor) + n/2 \leq 2c\lfloor \frac{n}{4} \rfloor + \frac{n}{2} \leq n\frac{c+1}{2} \leq c \cdot n \,,$$

for every $c \geq 1$.

Verification via Substitution Method

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Our Guess: $T(n) \leq c \cdot n$

Substitute into the Recurrence:

$$T(n) = 2T(\lfloor n/4 \rfloor) + n/2 \leq 2c\lfloor \frac{n}{4} \rfloor + \frac{n}{2} \leq n\frac{c+1}{2} \leq c \cdot n \;,$$

for every $c \geq 1$.

Verify the Base Case:

Verification via Substitution Method

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Our Guess: $T(n) \le c \cdot n$

Substitute into the Recurrence:

$$T(n) = 2T(\lfloor n/4 \rfloor) + n/2 \le 2c\lfloor \frac{n}{4} \rfloor + \frac{n}{2} \le n\frac{c+1}{2} \le c \cdot n \ ,$$

for every $c \ge 1$.

Verify the Base Case: $T(1)$

Verification via Substitution Method

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Our Guess: $T(n) \leq c \cdot n$

Substitute into the Recurrence:

$$T(n) = 2T(\lfloor n/4 \rfloor) + n/2 \leq 2c\lfloor \frac{n}{4} \rfloor + \frac{n}{2} \leq n\frac{c+1}{2} \leq c \cdot n \ ,$$

for every $c \geq 1$.

Verify the Base Case: $T(1) = 1 \leq c \cdot 1$

Verification via Substitution Method

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Our Guess: $T(n) \leq c \cdot n$

Substitute into the Recurrence:

$$T(n) \;=\; 2T(\lfloor n/4 \rfloor) + n/2 \leq 2c\lfloor \frac{n}{4} \rfloor + \frac{n}{2} \leq n\frac{c+1}{2} \leq c \cdot n \,,$$

for every $c \geq 1$.

Verify the Base Case: $T(1) = 1 \leq c \cdot 1 = c$ for every $c \geq 1$.

Verification via Substitution Method

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Our Guess: $T(n) \leq c \cdot n$

Substitute into the Recurrence:

$$T(n) \;=\; 2T(\lfloor n/4 \rfloor) + n/2 \leq 2c\lfloor \frac{n}{4} \rfloor + \frac{n}{2} \leq n\frac{c+1}{2} \leq c \cdot n \;,$$

for every $c \geq 1$.

Verify the Base Case: $T(1) = 1 \leq c \cdot 1 = c$ for every $c \geq 1$.

Summary:

Verification via Substitution Method

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Our Guess: $T(n) \leq c \cdot n$

Substitute into the Recurrence:

$$T(n) \;=\; 2T(\lfloor n/4 \rfloor) + n/2 \leq 2c\lfloor \frac{n}{4} \rfloor + \frac{n}{2} \leq n\frac{c+1}{2} \leq c \cdot n \;,$$

for every $c \geq 1$.

Verify the Base Case: $T(1) = 1 \leq c \cdot 1 = c$ for every $c \geq 1$.

Summary:

- We proved $T(n) \leq n$, for every $n \geq 1$

Verification via Substitution Method

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/4 \rfloor) + n/2$$

Our Guess: $T(n) \le c \cdot n$

Substitute into the Recurrence:

$$T(n) \;=\; 2T(\lfloor n/4 \rfloor) + n/2 \le 2c\lfloor \frac{n}{4} \rfloor + \frac{n}{2} \le n\frac{c+1}{2} \le c \cdot n \ ,$$

for every $c \ge 1$.

Verify the Base Case: $T(1) = 1 \le c \cdot 1 = c$ for every $c \ge 1$.

Summary:

- We proved $T(n) \le n$, for every $n \ge 1$
- Hence $T(n) \in O(n)$

Summary

Recursion Tree Method

Summary

Recursion Tree Method

- Assign contribution of subproblem to each node

Summary

Recursion Tree Method

- Assign contribution of subproblem to each node
- Sum up contributions using tree structure

Summary

Recursion Tree Method

- Assign contribution of subproblem to each node
- Sum up contributions using tree structure
- Allows us to be sloppy, since we only aim for a good guess

Summary

Recursion Tree Method

- Assign contribution of subproblem to each node
- Sum up contributions using tree structure
- Allows us to be sloppy, since we only aim for a good guess
- Verify guess with subsitution method

Summary

Recursion Tree Method

- Assign contribution of subproblem to each node
- Sum up contributions using tree structure
- Allows us to be sloppy, since we only aim for a good guess
- Verify guess with subsitution method

Substitution Method

Summary

Recursion Tree Method

- Assign contribution of subproblem to each node
- Sum up contributions using tree structure
- Allows us to be sloppy, since we only aim for a good guess
- Verify guess with subsitution method

Substitution Method

- Guess correct solution

Summary

Recursion Tree Method

- Assign contribution of subproblem to each node
- Sum up contributions using tree structure
- Allows us to be sloppy, since we only aim for a good guess
- Verify guess with subsitution method

Substitution Method

- Guess correct solution
- Verify guess using mathematical induction

Summary

Recursion Tree Method

- Assign contribution of subproblem to each node
- Sum up contributions using tree structure
- Allows us to be sloppy, since we only aim for a good guess
- Verify guess with subsitution method

Substitution Method

- Guess correct solution
- Verify guess using mathematical induction
- Guessing can be difficult and requires experience