Algorithms and Data Structures 2

6 – Analysis of recursive algorithms

Dr Michele Sevegnani

School of Computing Science
University of Glasgow

michele.sevegnani@glasgow.ac.uk

Outline

- **Analysis of recursive algorithms**
  - Solving recurrence equations
  - Master theorem

Analysis of recursive algorithms

- **The running time T(n) of recursive algorithms can be described by a recurrence equation**
  - Describes the overall running time on a problem of size n in terms of the running time on smaller inputs
- **Example**

```
MERGE-SORT(A,p,r)
  if p < r
    q := (p+r)/2
    MERGE-SORT(A,p,q)
    MERGE-SORT(A,q+1,r)
    MERGE(A,p,q,r)
```

$T(1) = O(1)$ — Base case

$T(n) = T(n/2) + T(n/2) + O(n)$
$= 2\,T(n/2) + O(n)$ — Recursive case

Recurrence equation

- **Exercise: find the recurrence equation for the running time T(n) of FACT(n)**

```
FACT(n)
  if n = 1
    return 1
  else
    return n * FACT(n-1)
```

Recurrence equation

- **Exercise: find the recurrence equation for the running time T(n) of FACT(n)**

```
FACT(n)
  if n = 1
    return 1
  else
    return n * FACT(n-1)
```

$T(1) = O(1)$

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

Solving recurrence equations

- **There are several methods to solve this kind of equations**
- **We will cover only three**
  - Iterative method
  - Recursion tree method
  - Master theorem

Iterative method

- **Continually substitute the recurrence relation on right hand-side of the recursive case**
- **Stop at the base case**
- **Example**

$$T(1) = O(1)$$

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

Iterative method

- **Continually substitute the recurrence relation on right hand-side of the recursive case**
- **Stop at the base case**
- **Example**

$$T(1) = c_1$$

$$T(n) = T(n - 1) + c_2$$

Rewrite equation to explicitly represent the time of executing constant operations

Iterative method

- **Continually substitute the recurrence relation on right hand-side of the recursive case**
- **Stop at the base case**
- **Example**

$T(1) = c_1$

$T(n) = T(n - 1) + c_2$

$\quad = T(n-2) + 2c_2$

Substitute $T(n-1)$ with $T(n-2) + c_2$

Iterative method

- **Continually substitute the recurrence relation on right hand-side of the recursive case**
- **Stop at the base case**
- **Example**

$$T(1) = c_1$$

$$\begin{aligned} T(n) &= T(n - 1) + c_2 \\ &= T(n-2) + 2c_2 \\ &= T(n-3) + 3c_2 \end{aligned}$$

Substitute $T(n-2)$ with $T(n-3) + c_2$

Iterative method

- **Continually substitute the recurrence relation on right hand-side of the recursive case**
- **Stop at the base case**
- **Example**

$T(1) = c_1$

$T(n) = T(n - 1) + c_2$

$= T(n-2) + 2c_2$

$= T(n-3) + 3c_2$

…

$= T(n-k) + kc_2$

Continue substituting

Iterative method

- **Continually substitute the recurrence relation on right hand-side of the recursive case**
- **Stop at the base case**
- **Example**

$$T(1) = c_1$$

$$T(n) = T(n-k) + kc_2$$

$$= T(1) + (n - 1)c_2$$

If we pick $k = n – 1$ then $T(n-(n-1)) = T(1)$

We hit the base case

Iterative method

- **Continually substitute the recurrence relation on right hand-side of the recursive case**
- **Stop at the base case**
- **Example**

$T(1) = c_1$

$T(n) = T(n\text{-}k) + kc_2$

$= T(1) + (n - 1)c_2$

$= c_1 + (n - 1)c_2$

Substitute T(1) with c_1

Iterative method

- **Continually substitute the recurrence relation on right hand-side of the recursive case**
- **Stop at the base case**
- **Example**

$$T(1) = c_1$$

$$\begin{aligned} T(n) &= T(n\text{-}k) + kc_2 \\ &= T(1) + (n - 1)c_2 \\ &= c_1 + (n - 1)c_2 \\ &= O(n) \end{aligned}$$

Since c_1 and c_2 are constants

Another example

- **Recurrence equation of MERGE-SORT**

$$T(1) = c_1$$

$$T(n) = 2\ T(n/2) + c_2 n$$

With explicit constants

Another example

- **Recurrence equation of MERGE-SORT**

$$
\begin{aligned}
T(1) &= c_1 \\
T(n) &= 2\,T(n/2) + c_2 n = \\
&= 4\,T(n/4) + 2c_2 n = \\
&= 8\,T(n/8) + 3c_2 n = \\
&= 16\,T(n/16) + 4c_2 n = \\
&= \ldots = \\
&= 2^k\,T(n/2^k) + kc_2 n
\end{aligned}
$$

Iteratively substitute

$T(n/2) = 2\,T(n/4) + c_2 n/2$
$T(n/4) = 2\,T(n/8) + c_2 n/4$
…

Another example

- **Recurrence equation of MERGE-SORT**

$T(1) = c_1$

$T(n) = 2\,T(n/2) + c_2 n =$

$= 4\,T(n/4) + 2c_2 n =$

$= 8\,T(n/8) + 3c_2 n =$

$= 16\,T(n/16) + 4c_2 n =$

$= \ldots =$

$= 2^k\,T(n/2^k) + kc_2 n =$

$= 2^{\log n}\,T(1) + (\log n \; * \; c_2 n) =$

$= c_1 n + c_2 n \log n =$

$= O(n \log n)$

We hit the base case if $T(n/2^k) = T(1)$

$n/2^k = 1$
$n = 2^k$

Take the logarithm on both sides

$\log n = \log 2^k$
$\log n = k \log 2$
$\log n = k$

$2^{\log n} = n$

Recursion tree method

1. **Draw the recursion tree (with running times)**
2. **Calculate the the running time of each level of the tree**
3. **Finally, sum the costs of running all levels**

- **Sometimes it can be tricky to use this method as the recursion tree can be degenerate**
  - We will see an example when analysing QUICKSORT

Example

- **Recursion tree of MERGE-SORT with running times**

$T(1) = c_1$

$T(n) = 2\ T(n/2) + c_2 n$

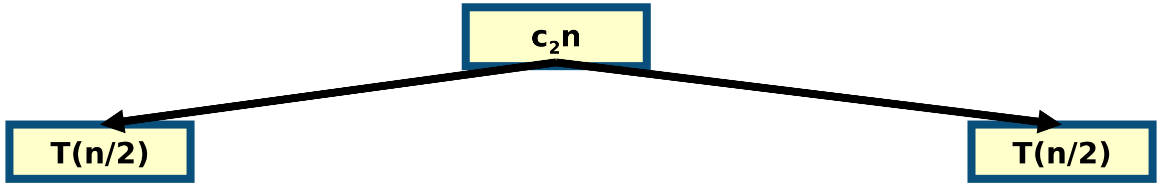
T(n)

Example

- **Recursion tree of MERGE-SORT with running times**

$T(1) = c_1$

$T(n) = 2\,T(n/2) + c_2 n$



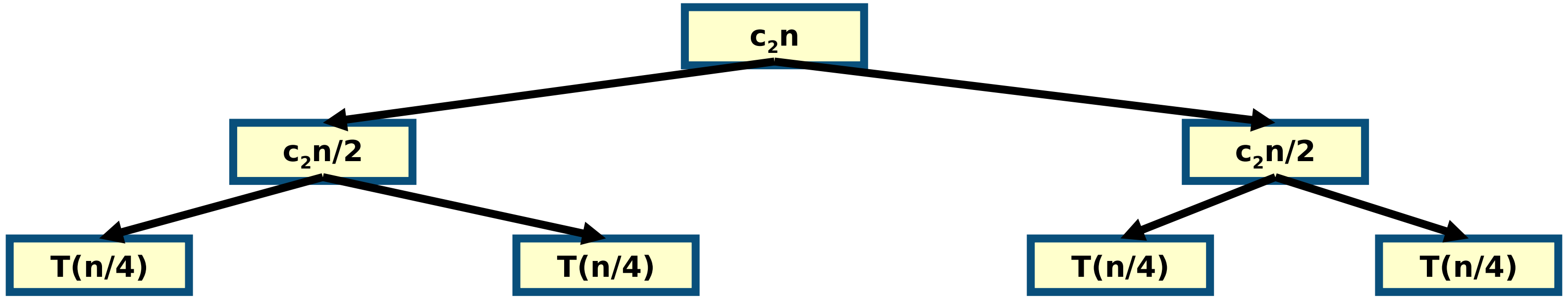
  - Expand applying recurrence equation

Example

- **Recursion tree of MERGE-SORT with running times**

$$T(1) = c_1$$

$$T(n) = 2\,T(n/2) + c_2 n$$



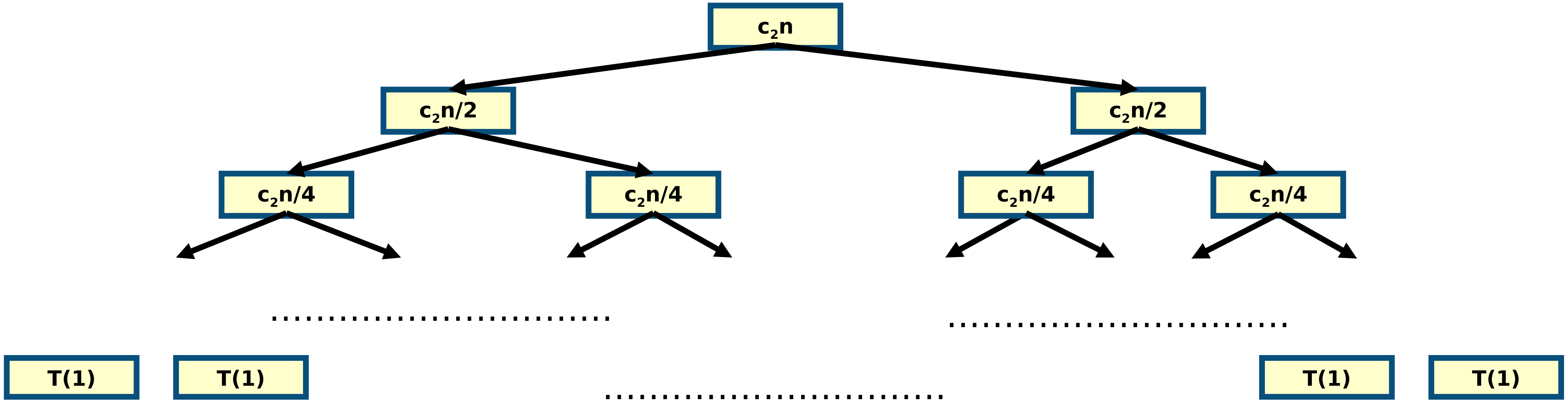
  - Continue expanding applying recurrence equation

Example

- **Recursion tree of MERGE-SORT with running times**

$T(1) = c_1$

$T(n) = 2\,T(n/2) + c_2 n$



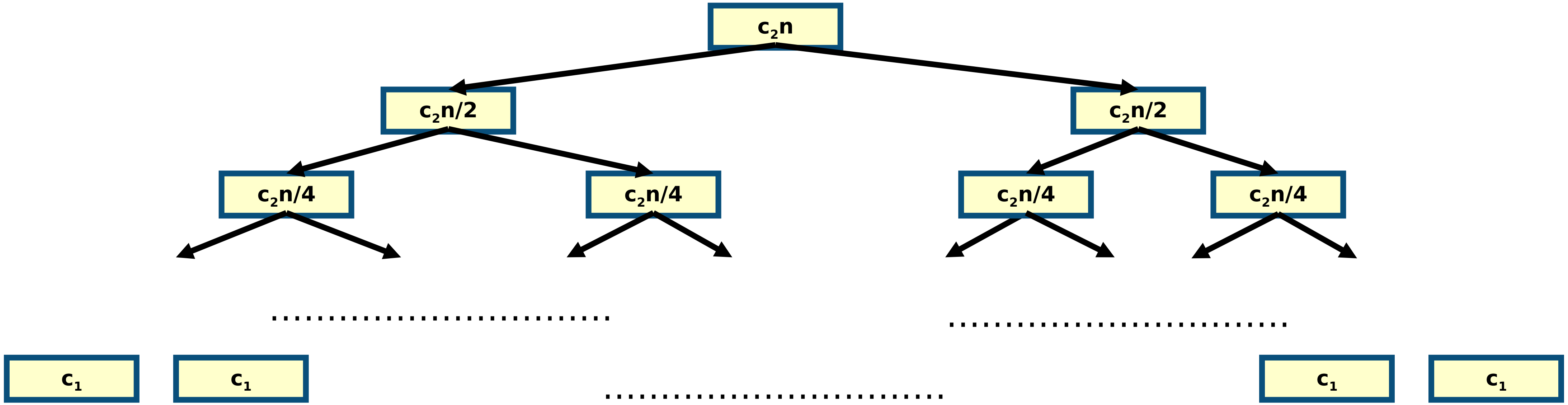
  - Continue expanding applying recurrence equation

Example

- **Recursion tree of MERGE-SORT with running times**

$T(1) = c_1$

$T(n) = 2\,T(n/2) + c_2 n$



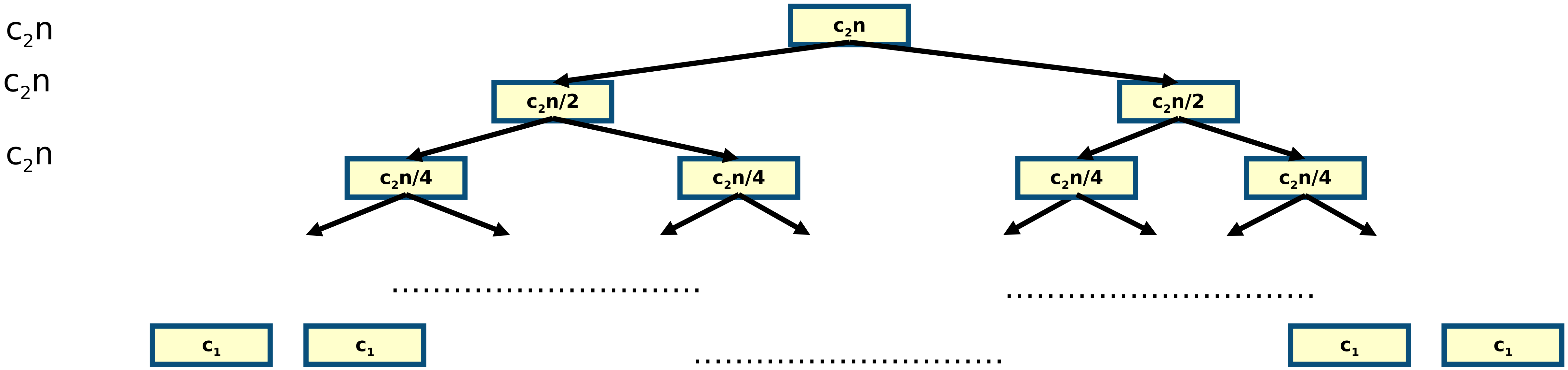
  - Base case

Example

- **Recursion tree of MERGE-SORT with running times**

$T(1) = c_1$

$T(n) = 2\,T(n/2) + c_2 n$



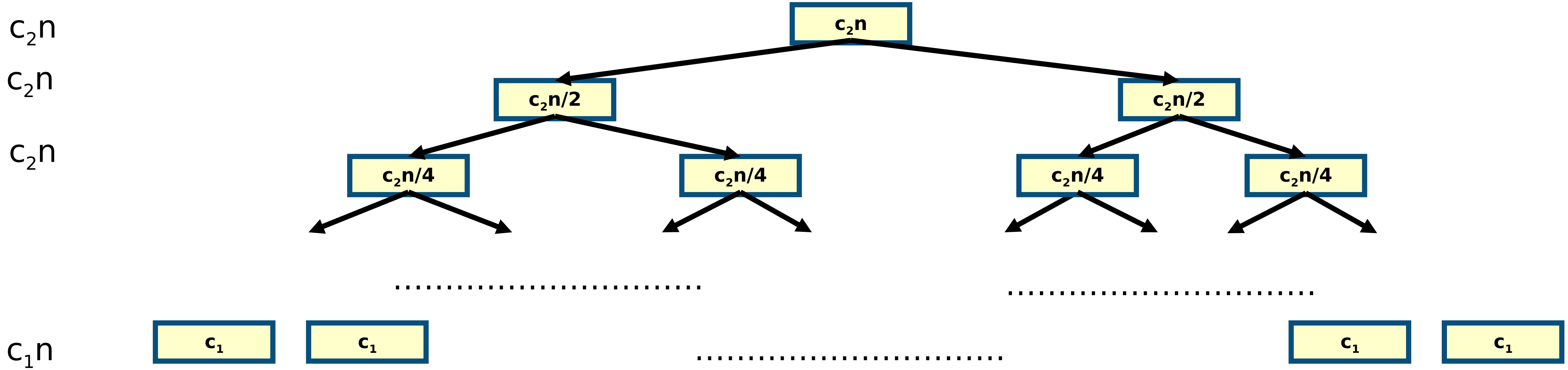
  - Compute the cost of executing each level

Example

- **Recursion tree of MERGE-SORT with running times**

$T(1) = c_1$

$T(n) = 2\,T(n/2) + c_2 n$



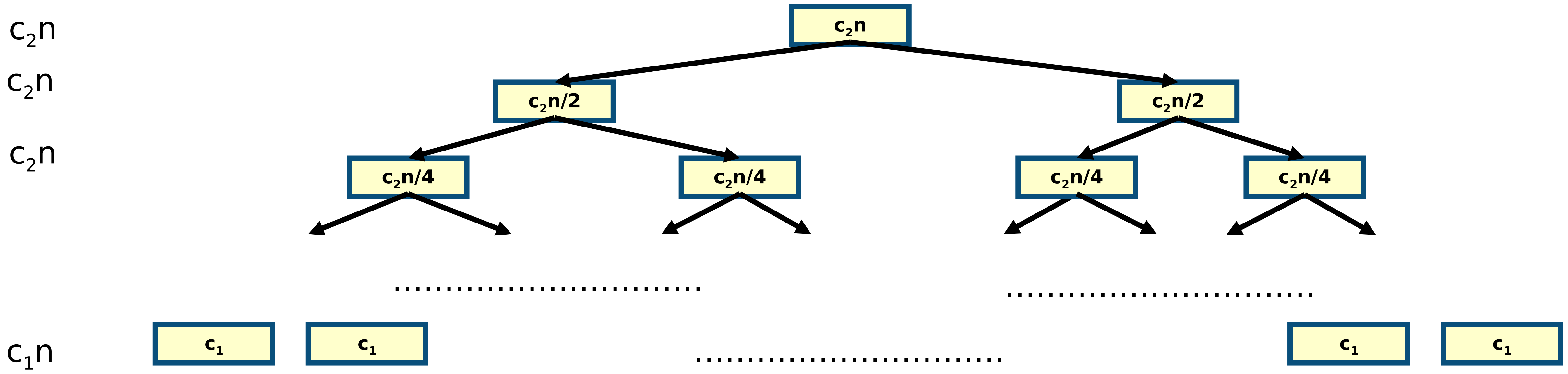
- We have n leaves and log n +1 levels

Example

- **Recursion tree of MERGE-SORT with running times**

$T(1) = c_1$

$T(n) = 2\,T(n/2) + c_2n$



– Adding up the costs of all the levels we obtain $c_2n \log n + c_1n = O(n \log n)$

Master theorem

- **Derived from recursion tree method**
- **Direct way to get the solution for the following type of recurrences**

$T(n) = aT(n/b) + f(n)$ with $a \geq 1$ and $b > 1$

- **There are three cases**

1. If $f(n) = \Theta(n^c)$ where $c < \log_b a$ then $T(n) = \Theta()$
2. If $f(n) = \Theta(n^c)$ where $c = \log_b a$ then $T(n) = \Theta(n^c \log n)$
3. If $f(n) = \Theta(n^c)$ where $c > \log_b a$ then $T(n) = \Theta(f(n))$

Example

$T(n) = aT(n/b) + f(n)$ **with** $a \geq 1$ **and** $b > 1$

1. If $f(n) = \Theta(n^c)$ where $c < \log_b a$ then $T(n) = \Theta()$
2. If $f(n) = \Theta(n^c)$ where $c = \log_b a$ then $T(n) = \Theta(n^c \log n)$
3. If $f(n) = \Theta(n^c)$ where $c > \log_b a$ then $T(n) = \Theta(f(n))$

- **MERGE-SORT recurrence is** $T(n) = 2T(n/2) + \Theta(n)$
  - $a = 2$, $b = 2$, $f(n) = \Theta(n)$
- **It falls in case 2 as** $c = 1$ **and** $\log_b a = \log_2 2 = 1$
- **The solution is** $\Theta(n^c \log n) = \Theta(n \log n)$

Summary

- **Analysis of recursive algorithms**
  - Solving recurrence equations
    - Iterative method
    - Recursion tree method
    - Master theorem