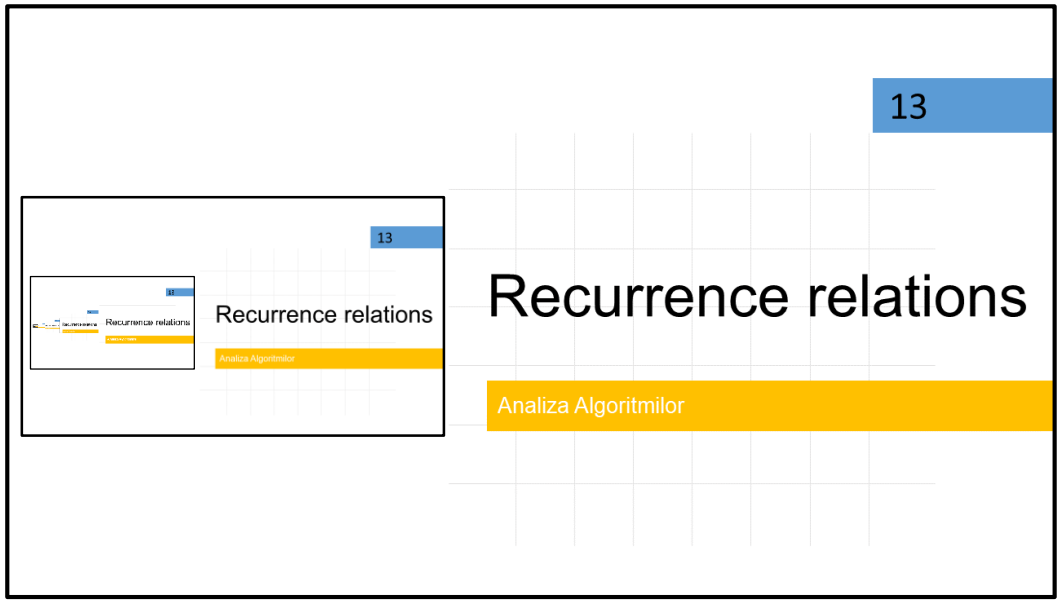
13



Recurrence relations

Analiza Algoritmilor

Merge sort

```
merge_sort(A, n):
1.      if (n <= 1)
2.          return A
3.      middle = n / 2
4.      A_left = merge_sort(A, middle)
5.      A_right = merge_sort(A + middle, n - middle)
6.      return merge(A_left, middle, A_right, n - middle)
```

$O(1)$

Merge sort

```
merge_sort(A, n):
1.      if (n <= 1)
2.          return A

3.      middle = n / 2
4.      A_left = merge_sort(A, middle)
5.      A_right = merge_sort(A + middle, n - middle)
6.      return merge(A_left, middle, A_right, n - middle)
```

$O(1)$

$O(n)$

Merge sort

```
merge_sort(A, n):
1.      if (n <= 1)
2.          return A

3.      middle = n / 2
4.      A_left = merge_sort(A, middle)
5.      A_right = merge_sort(A + middle, n - middle)
6.      return merge(A_left, middle, A_right, n - middle)
```

$???$

$O(1)$

$O(n)$

Merge sort

```
merge_sort(A, n):
1.      if (n <= 1)
2.          return A

3.      middle = n / 2
4.      A_left = merge_sort(A, middle)
5.      A_right = merge_sort(A + middle, n - middle)
6.      return merge(A_left, middle, A_right, n - middle)
```

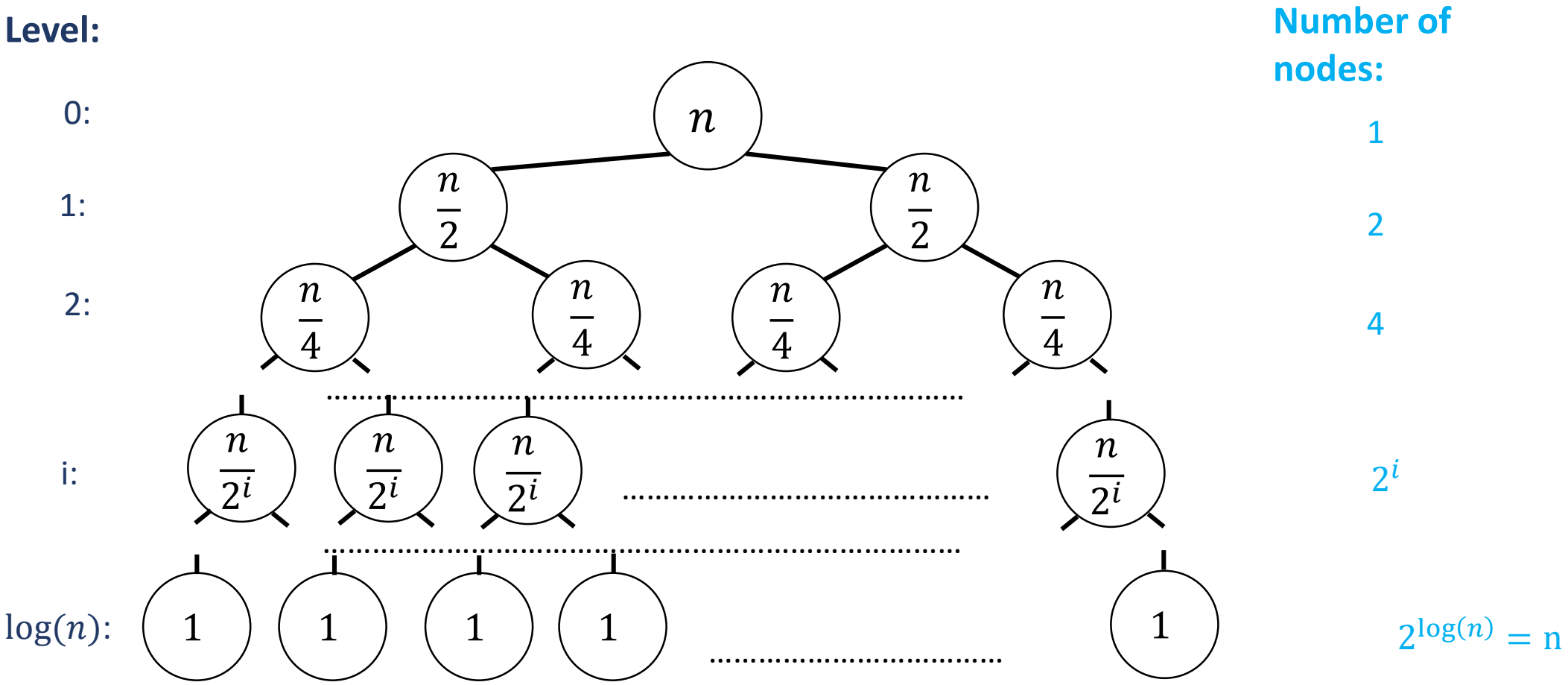
$???$

$O(1)$

$O(n)$

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

The recursion tree method



The substitution method

1. Make a guess
2. Prove it by induction

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

Guess: $T(n) \in O(n \log(n))$

Real-world recurrence relations

Maximum element of a list: $T(n) = T(n-1) + \Theta(1)$

Binary search: $T(n) = T\left(\frac{n}{2}\right) + \Theta(1)$

Karatsuba's algorithm: $T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n)$

Strassen's algorithm: $T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2)$

Cooley-Tukey FFT: $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$

Naïve SAT solving: $T(n) = 2T(n-1) + \Theta(1)$