

# Ch6: Analyzing Merge-Sort

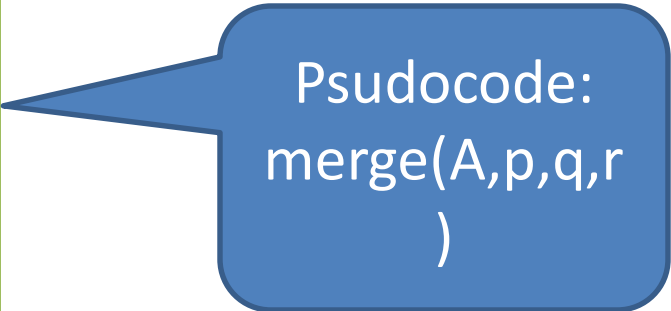
305234

Algorithm Analysis and Design

Jiraporn Pooksook  
Naresuan University

# Analyze the Correctness of Algorithm

```
N1 = q - p + 1
N2 = r - q
Create arrays L[1..N1+1] and R[1...N2+1]
For i = 1 to N1
    do L[i] = A[ p + i -1]
For j = 1 to N2
    do R[j] = A[q + j]
L[N1+1] = ∞
R[N2+1] = ∞
i = 1
j = 1
For k=p to r
    do if L[ i ] <= R[ j ]
        then A[k] = L[ i ]
            i = i+1
        else A[k] = R[ j ]
            j = j+1
```



Pseudocode:  
merge(A,p,q,r  
)

# Loop invariants with merge(A,p,q,r)

theoretically

loop invariant =  
before running loop k, all elements in  $A[p \dots k-1]$  contains k-p smallest number of  $L[1..N1+1]$ ,  $R[1..N2+1]$  are in sorted order and,  $L[i]$ ,  $R[j]$  are the smallest number of their arrays that have never been copied into A.

## Initialization:

Before running loop k,  $k = p$  then all elements in  $A[p \dots p-1]$  is empty.

Hence  $A[p \dots k-1]$  are sorted.

And  $i = j = 1$  hence,  $L[i]$  and  $R[j]$  are the smallest number. (True!!)

## Maintenance:

If before running loop k, all elements in  $A[p \dots k-1]$  contains k-p smallest number of  $L[1..N1+1]$ ,  $R[1..N2+1]$  are in sorted order.

then after running loop k, if  $L[i] \leq R[j]$  then  $L[i]$  is the smallest and will be copied to position k and  $A[p \dots k-1]$  are sorted.

Hence  $A[p \dots k]$  will contains  $k - p + 1$  smallest number.

if  $L[i] > R[j]$  then  $R[j]$  is the smallest and will be copied to position k and  $A[p \dots k-1]$  are sorted.

Hence  $A[p \dots k]$  will contains  $k - p + 1$  smallest number.

Hence before running loop k+1,  $A[p \dots k]$  are sorted. (True!!)

# Loop invariants with merge(A,p,q,r)

theoretically

loop invariant =  
before running loop k, all elements in  $A[p \dots k-1]$  contains k-p smallest number of  $L[1..N1+1]$ ,  $R[1..N2+1]$  are in sorted order and,  $L[i]$ ,  $R[j]$  are the smallest number of their arrays that have never been copied into A.

## Maintenance:

If before running loop k, all elements in  $A[p \dots k-1]$  contains k-p smallest number of  $L[1..N1+1]$ ,  $R[1..N2+1]$  are in sorted order.

then after running loop k, if  $L[i] \leq R[j]$  then  $L[i]$  is the smallest and will be copied to position k and  $A[p \dots k-1]$  are sorted.

Hence  $A[p \dots k]$  will contains k - p + 1 smallest number.

if  $L[i] > R[j]$  then  $R[j]$  is the smallest and will be copied to position k and  $A[p \dots k-1]$  are sorted.

Hence  $A[p \dots k]$  will contains k - p + 1 smallest number.

Hence before running loop k+1,  $A[p \dots k]$  are sorted. (True!!)

**Termination:** at starting of loop k+1,  $A[p \dots k]$  contain k+1 - p smallest number of  $L[1..N1+1]$ ,  $R[1..N2+1]$  are in sorted order. (True!!)

# **THE RUNNING TIME OF ALGORITHM**

# Analyze the Running time of merge-sort

If  $p < r$

then  $q = \lfloor (p+r)/2 \rfloor$

merge-sort(A, p, q)

merge-sort(A, q+1, r)

merge(A, p, q, r)

Divide

$$T_1 = t_1 * 1$$

Conquer

$$T_2 = 2T(n/2)$$

Combine

$$T_3 = t_3 * n$$

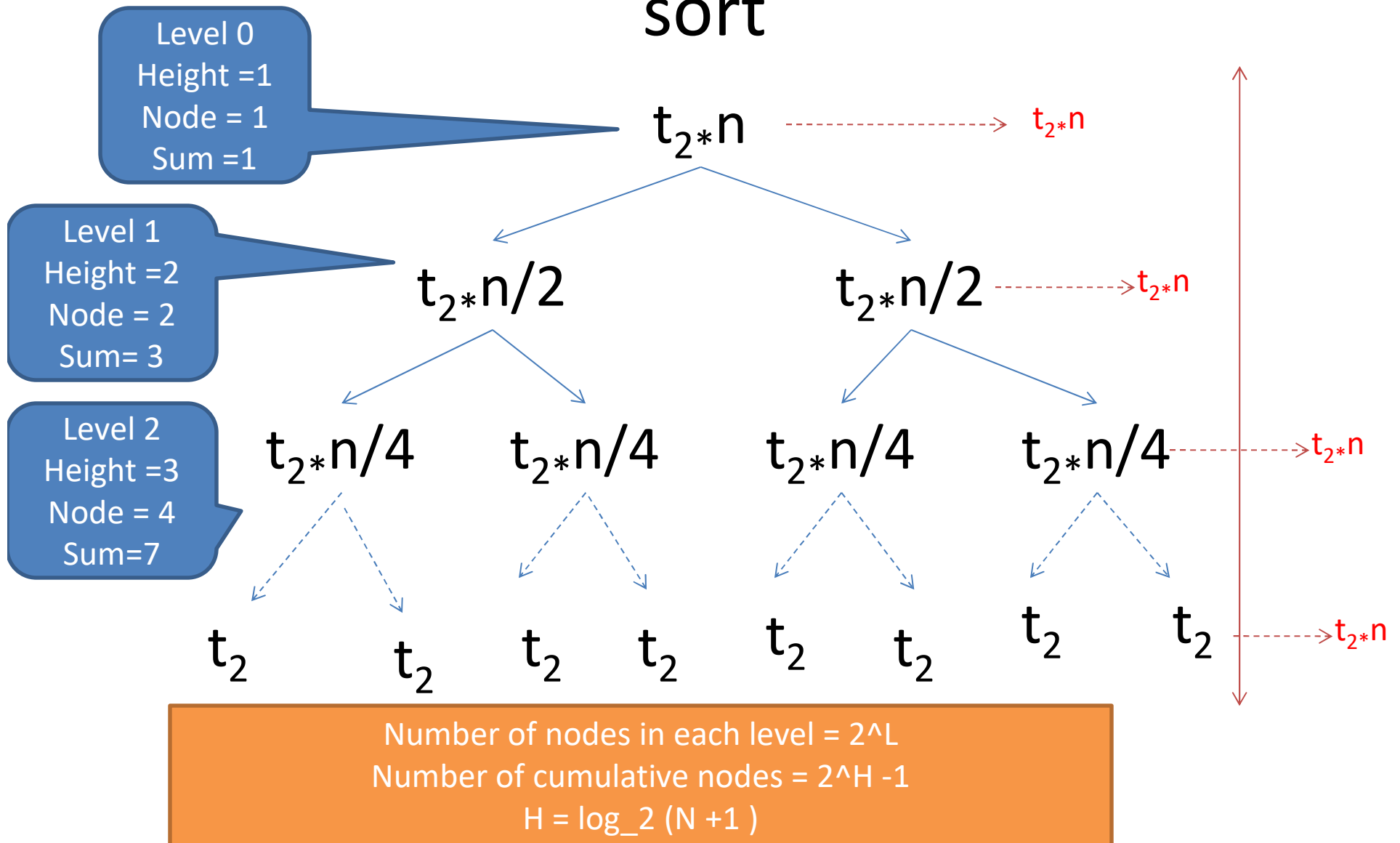
If  $n = 1$

$$T(n) = t_1$$

$$T(n) = 2T(n/2) + t_3 \times n$$

If  $n > 1$

# Analyze the Running time of merge-sort



# Analyze the Running time of merge-sort

