Design and Analysis of Algorithms I

Introduction

Merge Sort (Analysis)

Tim

Running Time of Merge Sort

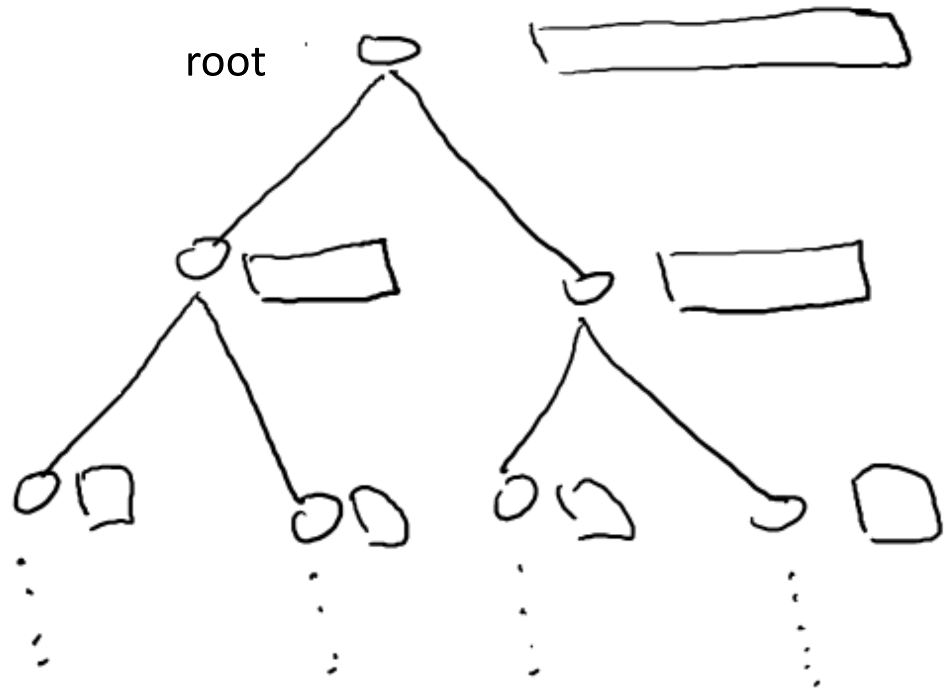
Claim: For every input array of n numbers, Merge Sort produces a sorted output array and uses at most $6n\log_2 n + 6n$ operations.

Proof of claim (assuming n = power of 2):

Level 0
[outer call to
Merge Sort]

Level 1
(1st recursive
calls)

Level 2



Te
ve

Roughly how many levels does this recursion tree have (as a function of n, the length of the input array)?

- ◯ A constant number (independent of n).
- ◯ $\log_2 n$ $(\log_2 n + 1)$ to be exact!
- ◯ $\sqrt{n}$
- ◯ n

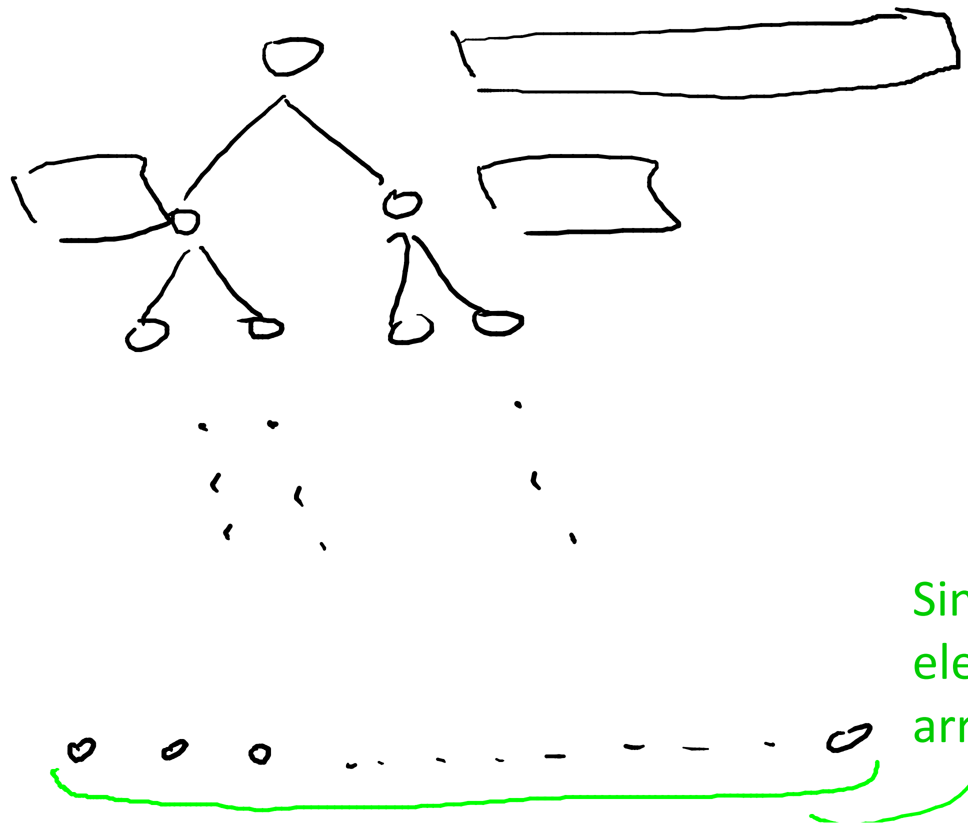
Proof of claim (assuming n = power of 2):

Level 0

Level 1

Level 2

Level $\log_2 n$



Single
element
arrays

What is the pattern ? Fill in the blanks in the following statement: at each level j = 0,1,2,.., $\log_2 n$, there are <blank> subproblems, each of size <blank>.

- ○ 2^j *and* 2^j, *respectively*
- ○ $n/2^j$ *and* $n/2^j$, *respectively*
- ○ 2^j *and* $n/2^j$, *respectively*
- ○ $n/2^j$ *and* 2^j, *respectively*

Ter
ver

Proof of claim (assuming n = power of 2) :

At each level j=0,1,2,.., $\log_2 n$,

Total # of operations at level j = 0,1,2,...,$\log_2 n$

$$\leq 2^j * 6(\frac{n}{2^j}) = 6n$$

# of level-j subproblems

Size of level-j subproblem

Work per level – j subproblem

Total

$$6n(\log_2 n + 1)$$

Work per level

# of levels

Running Time of Merge Sort

Claim: For every input array of n numbers, Merge Sort produces a sorted output array and uses at most $6n\log_2 n + 6n$ operations.

QED!