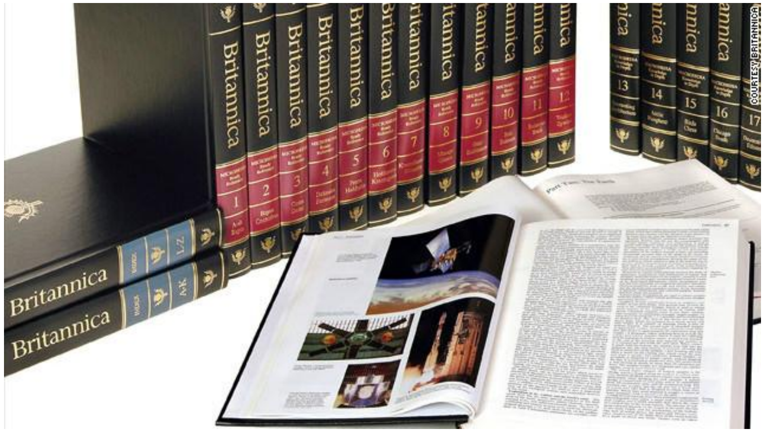
Divide and conquer
(recursion tree/unrolling)

Reading: Kleinberg & Tardos
Ch. 5.1 and 5.2
Additional resource: CLRS Ch. 4.4

Golden lion tamarins

Suppose you grew up in the 80s (!) and needed to write a report on the golden lion tamarin. You go to the library and find the encyclopedia for “G” with your fingers crossed that there is an entry for golden lion tamarin.



*How do you **search** for the entry quickly?*

Binary search, of course!

```
binarySearch( searchKey, sorted array A, loIndex, hiIndex ):
    if ( hiIndex < loIndex )
        return -1
    else
        midIndex = ( loIndex + hiIndex ) / 2
        if ( A[midIndex] == searchKey )
            return midIndex
        else if ( searchKey < A[midIndex] )
            return binarySearch( searchKey, A, loIndex, midIndex)
        else // searchKey > A[midIndex]
            return binarySearch( searchKey, A, midIndex, hiIndex)
```

What is the running time?

In ye olden days…

Suppose you needed to organize the contact information for your friends, but you don’t have access to any digital devices. You do have a fascinating solution called a “Rolodex” but… you dropped it on the floor, and now the cards are all mixed up!

*How can you **sort** your cards efficiently?*

Merge sort, of course!

```
mergeSort( array A ):
    if ( A.length == 1 ) // base case
        return
    else // recursive case
        midIndex = A.length / 2
        leftA = copy( A, 0, midIndex )
        mergeSort( leftA ) // recursively sort left half
        rightA = copy( A, midIndex + 1, A.length )
        mergeSort( rightA ) // recursively sort right half
        merge( leftA, rightA, A ) // merge halves back to A
```

What is the running time?

Divide and conquer design technique

- Divide into smaller problems (usually *recursively*)
- Conquer small problems (*base cases*)
- Combine solutions to smaller problems (*recursive* “glueing”)

How do we analyze the running time?

Recurrence relation

- Binary search

$$T(n) = T(n/2) + O(1) \qquad T(1) = O(1)$$

total running time

recursive half-sized call time

overhead for breaking into smaller problem and combining

Recurrence relation

- Binary search

$$T(n) = T(n/2) + O(1) \qquad T(1) = O(1)$$

total running time

recursive half-sized call time

overhead for breaking into smaller problem and combining

- Merge sort

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n) \qquad T(1) = O(1)$$

total running time

recursive left half call time

recursive right half call time

overhead for breaking into smaller problem and combining

Solving recurrence relations

- Goal: find closed form (no dependence on T)
- Approaches:
  - Recursion tree/unrolling
  - Guess solution & check with induction
  - Master theorem [when applicable]
- Examples
  - Binary search

    $T(n) = T(n/2) + O(1)$ $\qquad$ $T(1) = O(1)$

    $\mathbf{T(n) = O(\log n)}$
  - Merge sort

    $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n)$ $\qquad$ $T(1) = O(1)$

    $\mathbf{T(n) = O(n \log n)}$

Recursion tree/unrolling: binary search recurrence

$T(n) = T(n/2) + O(1)$
$T(1) = O(1)$

Recursion tree

$T(n)$

Unrolling

$T(n)$

Recursion tree/unrolling: binary search recurrence

$T(n) = T(n/2) + O(1)$
$T(1) = O(1)$

Recursion tree

$$
\begin{array}{c}
O(1) \\
| \\
T(n/2)
\end{array}
$$

Unrolling

$T(n)$

$= T(n/2) + O(1)$

Recursion tree/unrolling: binary search recurrence

$T(n) = T(n/2) + O(1)$
$T(1) = O(1)$

Recursion tree

$$\begin{array}{c} O(1) \\ | \\ T(n/2) \end{array}$$

Unrolling

$T(n)$

$= \mathbf{T(n/2)} + O(1)$

Recursion tree/unrolling: binary search recurrence

$T(n) = T(n/2) + O(1)$
$T(1) = O(1)$

Recursion tree

$$
\begin{array}{c}
O(1) \\
| \\
O(1) \\
| \\
T(n/4)
\end{array}
$$

Unrolling

$T(n)$

$= \mathbf{T(n/2)} + O(1)$

$= \mathbf{[T(n/4) + O(1)]} + O(1)$

Recursion tree/unrolling: binary search recurrence

$T(n) = T(n/2) + O(1)$
$T(1) = O(1)$

Recursion tree

$O(1)$

$O(1)$

$T(n/4)$

Unrolling

$T(n)$

$= T(n/2) + O(1)$

$= T(n/4) + O(1) + O(1)$

Recursion tree/unrolling: binary search recurrence

$T(n) = T(n/2) + O(1)$
$T(1) = O(1)$

Recursion tree

$$\begin{array}{c} O(1) \\ | \\ O(1) \\ | \\ T(n/4) \end{array}$$

Unrolling

$T(n)$

$= T(n/2) + O(1)$

$= \mathbf{T(n/4)} + O(1) + O(1)$

Recursion tree/unrolling: binary search recurrence

$T(n) = T(n/2) + O(1)$
$T(1) = O(1)$

Recursion tree

$$\begin{array}{c} O(1) \\ | \\ O(1) \\ | \\ O(1) \\ | \\ T(n/8) \end{array}$$

Unrolling

$T(n)$

$= T(n/2) + O(1)$

$= \mathbf{T(n/4)} + O(1) + O(1)$

$= \mathbf{[T(n/8) + O(1)]} + O(1) + O(1)$

Recursion tree/unrolling: binary search recurrence

$T(n) = T(n/2) + O(1)$
$T(1) = O(1)$

Recursion tree

$O(1)$

$O(1)$

$O(1)$

$T(n/8)$

Unrolling

$T(n)$

$= T(n/2) + O(1)$

$= T(n/4) + O(1) + O(1)$

$= T(n/8) + O(1) + O(1) + O(1)$

Recursion tree/unrolling: binary search recurrence

$T(n) = T(n/2) + O(1)$
$T(1) = O(1)$

Recursion tree

tree height is?

$O(1)$
$O(1)$
$O(1)$
$O(1)$
…
$O(1)$

Unrolling

$T(n)$

$= T(n/2) + O(1)$

$= T(n/4) + O(1) + O(1)$

$= T(n/8) + O(1) + O(1) + O(1)$

…

$= \underbrace{O(1) + O(1) + \ldots + O(1)}_{\#\text{ unrollings is?}}$

Recursion tree/unrolling: binary search recurrence

$T(n) = T(n/2) + O(1)$
$T(1) = O(1)$

Recursion tree

$$\text{O(log n) height} \left\{ \begin{array}{c} O(1) \\ | \\ O(1) \\ | \\ O(1) \\ | \\ O(1) \\ \ldots \\ O(1) \end{array} \right.$$

Unrolling

$T(n)$

$= T(n/2) + O(1)$

$= T(n/4) + O(1) + O(1)$

$= T(n/8) + O(1) + O(1) + O(1)$

$\ldots$

$$= \underbrace{O(1) + O(1) + \ldots + O(1)}_{\text{O(log n) unrollings}}$$

Recursion tree/unrolling: binary search recurrence

$T(n) = T(n/2) + O(1)$
$T(1) = O(1)$

Recursion tree

$$O(\log n) * O(1) = \mathbf{O(\log n)} \quad \left\{ \begin{array}{c} O(1) \\ | \\ O(1) \\ | \\ O(1) \\ | \\ O(1) \\ \ldots \\ O(1) \end{array} \right.$$

Unrolling

$T(n)$

$= T(n/2) + O(1)$

$= T(n/4) + O(1) + O(1)$

$= T(n/8) + O(1) + O(1) + O(1)$

$\ldots$

$= \underbrace{O(1) + O(1) + \ldots + O(1)}_{O(\log n) * O(1) = \mathbf{O(\log n)}}$

Recurrence relation: merge sort

Solve the recurrence relation for merge sort using both the recursion tree and unrolling methods.

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n) \qquad T(1) = O(1)$$

When n is even…

$$T(n) = 2*T(n/2) + O(n) \qquad T(1) = O(1)$$

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$
$T(1) = O(1)$

Recursion tree **Unrolling**

$T(n)$ $T(n)$

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$
$T(1) = O(1)$

Recursion tree

$O(n)$

$T(n/2)$ $T(n/2)$

Unrolling

$T(n)$

$= 2*T(n/2) + O(n)$

Recursion tree/unrolling: merge sort recurrence

T(n) = 2*T(n/2) + O(n)
T(1) = O(1)

Recursion tree

O(n)

T(n/2) T(n/2)

Unrolling

T(n)

= 2*T(n/2) + O(n)

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$
$T(1) = O(1)$

Recursion tree

$O(n)$

$T(n/2)$ $T(n/2)$

Unrolling

$T(n)$

$= 2*\mathbf{T(n/2)} + O(n)$

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$
$T(1) = O(1)$

Recursion tree

$O(n)$

$O(n/2)$ $O(n/2)$

$T(n/4)$ $T(n/4)$ $T(n/4)$ $T(n/4)$

Unrolling

$T(n)$

$= 2*\mathbf{T(n/2)} + O(n)$

$= 2*\mathbf{[2T(n/4)+O(n/2)]} + O(n)$

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$

$T(1) = O(1)$

Recursion tree

$O(n)$

$O(n/2)$ $O(n/2)$

$T(n/4)$ $T(n/4)$ $T(n/4)$ $T(n/4)$

Unrolling

$T(n)$

$= 2*T(n/2) + O(n)$

$= 2*2T(n/4)+ 2*O(n/2) + O(n)$

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$
$T(1) = O(1)$

Recursion tree

$O(n)$

$O(n/2)$ $O(n/2)$

$T(n/4)$ $T(n/4)$ $T(n/4)$ $T(n/4)$

Unrolling

$T(n)$

$= 2*T(n/2) + O(n)$

$= 2*2T(n/4)+ O(n) + O(n)$

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$
$T(1) = O(1)$

Recursion tree

$O(n)$

$O(n/2)$ $O(n/2)$

$T(n/4)$ $T(n/4)$ $T(n/4)$ $T(n/4)$

Unrolling

$T(n)$

$= 2*T(n/2) + O(n)$

$= 2*2\mathbf{T(n/4)} + O(n) + O(n)$

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$
$T(1) = O(1)$

Recursion tree

$O(n)$

$O(n/2)$ $O(n/2)$

$O(n/4)$ $O(n/4)$ $O(n/4)$ $O(n/4)$

$T(n/8)$ $T(n/8)$ $T(n/8)$ $T(n/8)$ $T(n/8)$ $T(n/8)$ $T(n/8)$ $T(n/8)$

Unrolling

$T(n)$

$= 2*T(n/2) + O(n)$

$= 2*2\mathbf{T(n/4)} + O(n) + O(n)$

$= 2*2*\mathbf{[2T(n/8) + O(n/4)]} + O(n) + O(n)$

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$
$T(1) = O(1)$

Recursion tree

$O(n)$

$O(n/2)$ $O(n/2)$

$O(n/4)$ $O(n/4)$ $O(n/4)$ $O(n/4)$

$T(n/8)$ $T(n/8)$ $T(n/8)$ $T(n/8)$ $T(n/8)$ $T(n/8)$ $T(n/8)$ $T(n/8)$

Unrolling

$T(n)$

$= 2*T(n/2) + O(n)$

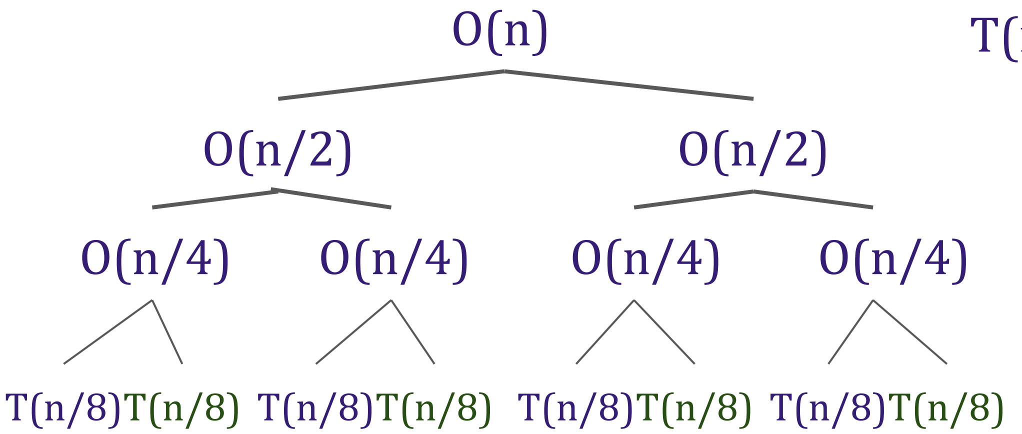
$= 2*2T(n/4)+ O(n) + O(n)$

$= 2*2*2T(n/8) + 4*O(n/4)+O(n)+O(n)$

Recursion tree/unrolling: merge sort recurrence

$$T(n) = 2*T(n/2) + O(n)$$
$$T(1) = O(1)$$

Recursion tree



Unrolling

$T(n)$

$= 2*T(n/2) + O(n)$

$= 2*2T(n/4)+ O(n) + O(n)$

$= 2*2*2T(n/8) + O(n)+O(n)+O(n)$

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$
$T(1) = O(1)$

Recursion tree

$O(n)$

$O(n/2) \quad O(n/2)$

$O(n/4) \quad O(n/4) \quad O(n/4) \quad O(n/4)$

$O(n/8)O(n/8) \quad O(n/8)O(n/8) \quad O(n/8)T(n/8) \quad O(n/8)O(n/8)$

…

$O(1) \dots O(1)$

levels is?

Unrolling

$T(n)$

$= 2*T(n/2) + O(n)$

$= 2*2T(n/4)+ O(n) + O(n)$

$= 2*2*2T(n/8) + O(n)+O(n)+O(n)$

…

$= \underbrace{2 * \dots *2}T(0) + \underbrace{O(n) + \dots + O(n)}$

unrollings is?

Recursion tree/unrolling: merge sort recurrence

T(n) = 2*T(n/2) + O(n)
T(1) = O(1)

Recursion tree

O(n)

O(n/2) O(n/2)

O(n/4) O(n/4) O(n/4) O(n/4)

O(n/8)O(n/8) O(n/8)O(n/8) O(n/8)T(n/8) O(n/8)O(n/8)

…

O(1) … O(1)

#levels:
1 + log n

Unrolling

T(n)

= 2*T(n/2) + O(n)

= 2*2T(n/4)+ O(n) + O(n)

= 2*2*2T(n/8) + O(n)+O(n)+O(n)

…

$$= \underbrace{2 * \ldots *2}T(0) + \underbrace{O(n) + \ldots + O(n)}$$

unrollings: log n

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$
$T(1) = O(1)$

Recursion tree

O(n)

O(n/2) O(n/2)

O(n/4) O(n/4) O(n/4) O(n/4)

O(n/8)O(n/8) O(n/8)O(n/8) O(n/8)T(n/8) O(n/8)O(n/8)

…

O(1) … O(1)

#levels:
1 + log n

work at
depth i
?

Unrolling

$T(n)$

$= 2*T(n/2) + O(n)$

$= 2*2T(n/4) + O(n) + O(n)$

$= 2*2*2T(n/8) + O(n)+O(n)+O(n)$

…

$= \underbrace{2 * \dots *2}T(0) + \underbrace{O(n) + \dots + O(n)}$

unrollings: log n

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$
$T(1) = O(1)$

depth 0:
1 node
O(n) work

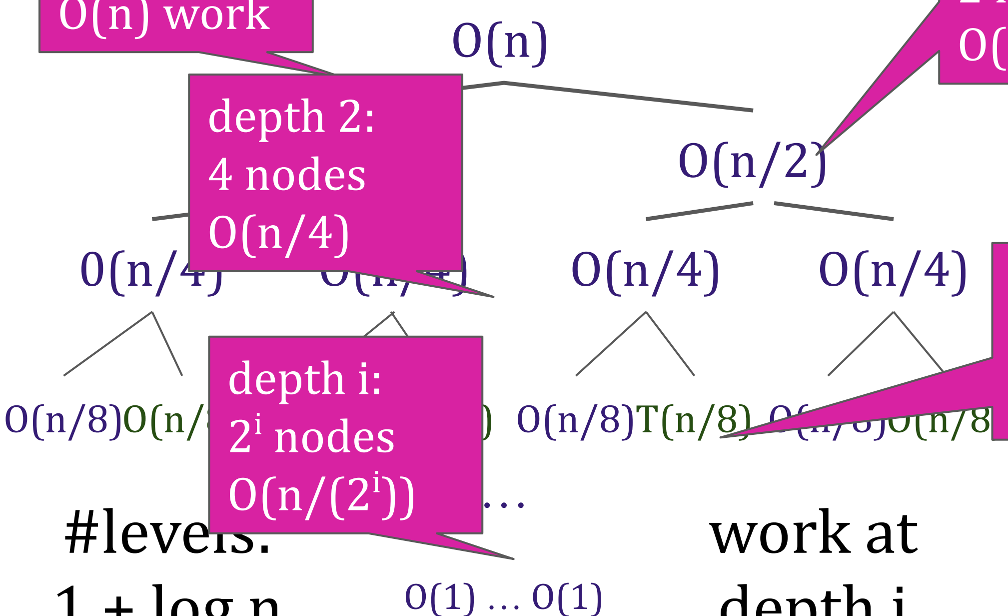
depth 1:
2 nodes
O(n/2)

depth 2:
4 nodes
O(n/4)

depth 3:
8 nodes
O(n/8)

depth i:
2^i nodes
$O(n/(2^i))$

Recursion tree



#levels:
1 + log n

work at
depth i
?

Unrolling

$= 2*T(n/2) + O(n)$

$/4)+ O(n) + O(n)$

$(n/8) + O(n)+O(n)+O(n)$

…

$= \underbrace{2 * \ldots *2}T(0) + \underbrace{O(n) + \ldots + O(n)}$

unrollings: log n

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$
$T(1) = O(1)$

Recursion tree

$O(n)$

$O(n/2)$ $O(n/2)$

$O(n/4)$ $O(n/4)$ $O(n/4)$ $O(n/4)$

$O(n/8)O(n/8)$ $O(n/8)O(n/8)$ $O(n/8)T(n/8)$ $O(n/8)O(n/8)$

…

$O(1)$ … $O(1)$

#levels:
$1 + \log n$

work at depth i:
$2^i * O(n/2^i)$

Unrolling

$T(n)$

$= 2*T(n/2) + O(n)$

$= 2*2T(n/4) + O(n) + O(n)$

$= 2*2*2T(n/8) + O(n)+O(n)+O(n)$

…

$= \underbrace{2 * \ldots *2}T(0) + \underbrace{O(n) + \ldots + O(n)}$

unrollings: log n

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$
$T(1) = O(1)$

Recursion tree

$O(n)$

$O(n/2)$ $O(n/2)$

$O(n/4)$ $O(n/4)$ $O(n/4)$ $O(n/4)$

$O(n/8)$ $O(n/8)$ $O(n/8)$ $O(n/8)$ $O(n/8)$ $T(n/8)$ $O(n/8)$ $O(n/8)$

…

$O(1)$ … $O(1)$

#levels: 1 + log n

work at depth i: O(n)

Unrolling

$T(n)$

$= 2*T(n/2) + O(n)$

$= 2*2T(n/4)+ O(n) + O(n)$

$= 2*2*2T(n/8) + O(n)+O(n)+O(n)$

…

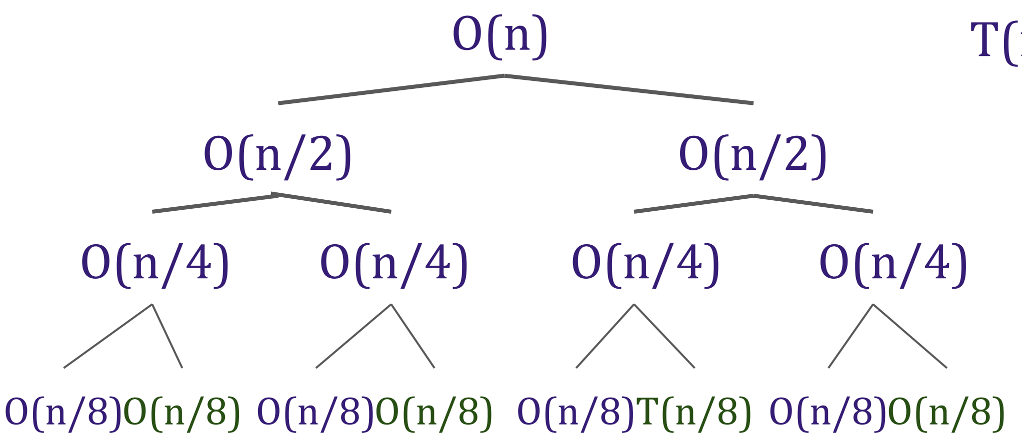
$= \underbrace{2 * \ldots *2}T(0) + \underbrace{O(n) + \ldots + O(n)}$

unrollings: log n

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$
$T(1) = O(1)$

Recursion tree



…

O(1) … O(1)

#levels:
1 + log n

work at
each level:
O(n)

Unrolling

$T(n)$

$= 2*T(n/2) + O(n)$

$= 2*2T(n/4)+ O(n) + O(n)$

$= 2*2*2T(n/8) + O(n)+O(n)+O(n)$

…

$\underbrace{= 2 * \ldots *2}T(0) + \underbrace{O(n) + \ldots + O(n)}$

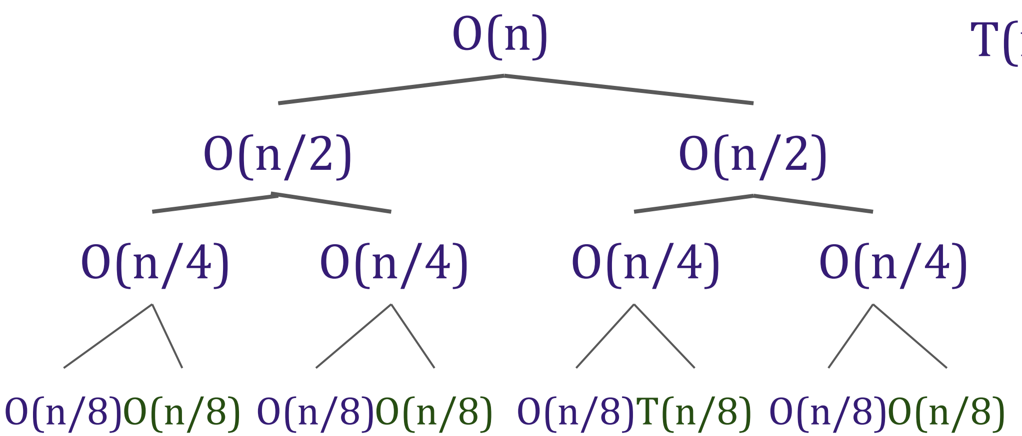
unrollings: log n

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$

$T(1) = O(1)$

Recursion tree



…

O(1) … O(1)

#levels:
1 + log n

work at each level: O(n)

total: O(n log n)

Unrolling

$T(n)$

$= 2*T(n/2) + O(n)$

$= 2*2T(n/4)+ O(n) + O(n)$

$= 2*2*2T(n/8) + O(n)+O(n)+O(n)$

…

$= \underbrace{2 * \ldots *2}T(0) + \underbrace{O(n) + \ldots + O(n)}$

\# unrollings: log n

Recursion tree/unrolling: merge sort recurrence

T(n) = 2*T(n/2) + O(n)
T(1) = O(1)

Recursion tree

O(n)

O(n/2) O(n/2)

O(n/4) O(n/4) O(n/4) O(n/4)

O(n/8)O(n/8) O(n/8)O(n/8) O(n/8)T(n/8) O(n/8)O(n/8)

…

O(1) … O(1)

#levels:
1 + log n

work at
each level:
O(n)

total: O(n log n)

Unrolling

T(n)

= 2*T(n/2) + O(n)

= 2*2T(n/4)+ O(n) + O(n)

= 2*2*2T(n/8) + O(n)+O(n)+O(n)

…

$$= \underbrace{2 * \ldots *2}T(0) + \underbrace{O(n) + \ldots + O(n)}$$

unrollings: log n

= $(2^{\log n})$T(0) + (log n)O(n) = **O(n log n)**

Recursion tree/unrolling: merge sort recurrence

$T(n) = 2*T(n/2) + O(n)$
$T(1) = O(1)$

Recursion tree

O(n)

O(n/2) O(n/2)

O(n/4) O(n/4) O(n/4) O(n/4)

O(n/8) O(n/8) O(n/8) O(n/8) O(n/8) T(n/8) O(n/8) O(n/8)

…

O(1) … O(1)

#levels:
1 + log n

work at each level:
O(n)

total: O(n log n)

Unrolling

$T(n)$

$= 2*T(n/2) + O(n)$

$= 2*2T(n/4) + O(n) + O(n)$

$= 2*2*2T(n/8) + O(n) + O(n) + O(n)$

…

base cases
*base case work

$= \underbrace{2 * \ldots * 2}T(0) + \underbrace{O(n) + \ldots + O(n)}$

unrollings: log n

$= (2^{\log n})T(0) + (\log n)O(n) = \mathbf{O(n \log n)}$

Recurrence relation: merge sort

$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n)$ $\quad T(1) = O(1)$

When n is even…

$T(n) = 2*T(n/2) + O(n)$ $\quad T(1) = O(1)$

$T(n) = O(n \log n)$

Recurrence relation: merge sort

$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n)$ $\quad T(1) = O(1)$

When n is even…

$T(n) = 2*T(n/2) + O(n)$ $\quad T(1) = O(1)$

$\mathbf{T(n) = O(n \log n)}$

When n is odd…

$T(n) \leq T(n+1)$

$T(n) = O((n+1)\log(n+1))$

<u>Aside: $\log(n+1) = O(\log n)$</u>

$\log(n+1) \leq \log(n + n)$

$\log(n+1) \leq \log(2n)$

$\log(n+1) \leq \log(2) + \log(n)$

$\log(n+1) \leq 1 + \log(n)$

$\log(n+1) = O(\log n)$

Recurrence relation: merge sort

$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n)$ $T(1) = O(1)$

When n is even…

$T(n) = 2*T(n/2) + O(n)$ $T(1) = O(1)$

$\mathbf{T(n) = O(n \log n)}$

When n is odd…

$T(n) \leq T(n+1)$

$T(n) = O((n+1)\log(n))$

$T(n) = O(n \log n + \log n)$

Recurrence relation: merge sort

$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n)$ $T(1) = O(1)$

When n is even…

$T(n) = 2*T(n/2) + O(n)$ $T(1) = O(1)$

$\mathbf{T(n) = O(n \log n)}$

When n is odd…

$T(n) \leq T(n+1)$

$T(n) = O((n+1)\log(n))$

$T(n) = O(n \log n + \log n)$

$\mathbf{T(n) = O(n \log n)}$