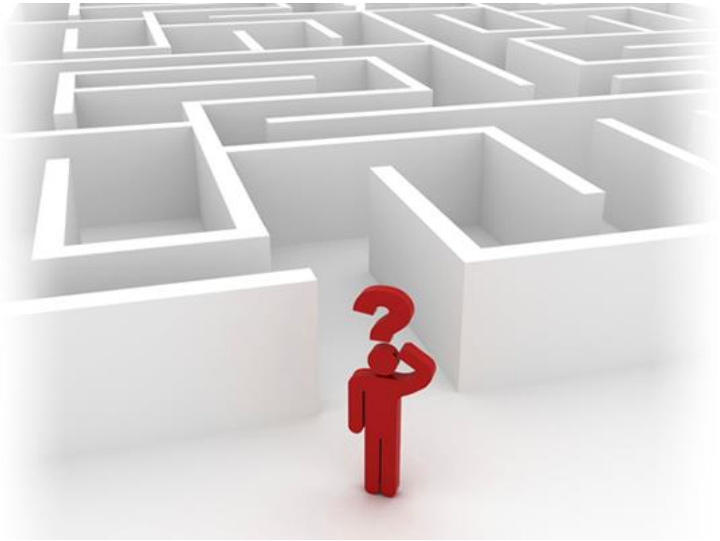


Design and Analysis of Algorithms I

Introduction

Merge Sort (Pseudocode)

Tim

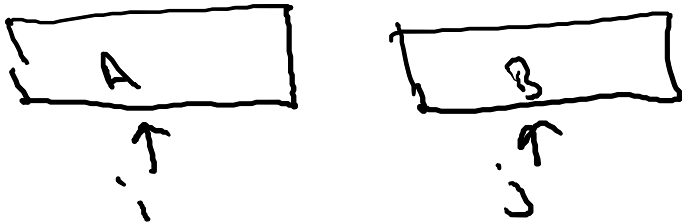
Merge Sort: Pseudocode

- recursively sort 1st half of input array
- " " 2nd " " " "
- merge two sorted sublists into one

[ignores base cases]

Pseudocode for Merge:

```
C = output array [length = n]
A = 1st sorted array [n/2]
B = 2nd   "     "    [n/2]
```




```
i = 1
j = 1
```

```
for k = 1 to n
    if A(i) < B(j)
        C(k) = A(i)
        i++
    else [B(j) < A(i)]
        C(k) = B(j)
        j++
end    [ignore end cases]
```

Tim Roughgarden

Pseudocode for Merge:

```
C = output [length = n]
A = 1st sorted array [n/2]
B = 2nd sorted array [n/2]
i = 1
j = 1

for k = 1 to n
        if A(i) < B(j)
                C(k) = A(i)
                i++
        else [B(j) < A(i)]
                C(k) = B(j)
                j++
end
```

(ignores end cases)

Tim Roughgarden

Merge Sort Running Time?

Key Question: running time of MergeSort on array of n numbers?

[running time ≈ # of lines of code executed]

Pseudocode for Merge:

C = output [length = n]

A = 1st sorted array [n/2]

B = 2nd sorted array [n/2]

i = 1

j = 1

2 operations

```
for k = 1 to n
        if A(i) < B(j)
                C(k) = A(i)
                i++
        else [B(j) < A(i)]
                C(k) = B(j)
                j++
end
```

(ignores end cases)

Running Time of Merge

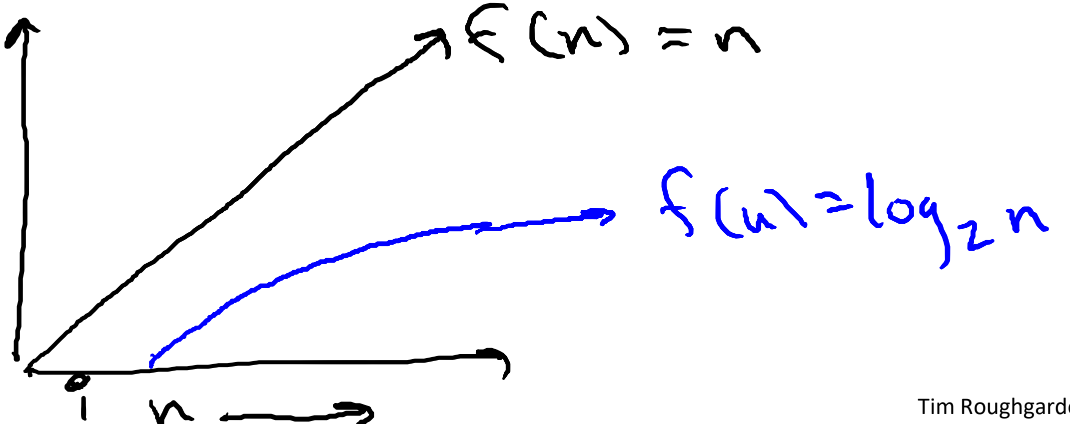
Upshot: running time of Merge on array of m numbers is $\leq 4m + 2$

$\leq 6m$ (Since $m \geq 1$)

Running Time of Merge Sort

Claim: MergeSort requires $\boxed{\le 6n\log_2 n + 6n}$ operations to sort n numbers.

Recall: $\log_2 n$ = # of times you divide by 2 until you get down to 1.



Tim Roughgarden