Design and Analysis of Algorithms I

Introduction

Merge Sort (Pseudocode)

Tim

Merge Sort: Pseudocode

-- recursively sort 1st half of the input array

-- recursively sort 2nd half of the input array

-- merge two sorted sublists into one

[ignores base cases]

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)

Merge Sort Running Time?

Key Question : running time of Merge Sort 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
```

4 operations for each k

(ignores end cases)

Tim Roughgarden

Running Time of Merge

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

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

一共有logn个C array 所以最后的
merge sort operations = 6n * logn + 6n

Running Time of Merge Sort

Claim : Merge Sort requires $\leq 6n \log_2 n + 6n$ operations to sort n numbers.

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