

CSC236 fall 2016

more complexity: mergesort

Danny Heap

heap@cs.toronto.edu / BA4270 (behind elevators)

<http://www.cdf.toronto.edu/~heap/236/F16/>

416-978-5899

Using Introduction to the Theory of Computation,
Chapter 3



Outline

vexing complexity

mergesort

Divide-and-conquer

Notes



Upper bound on $T(n)$

trouble!



recurrence for MergeSort

```
MergeSort(A,b,e):  
    if b == e: return  
    m = (b + e) / 2  
    MergeSort(A,b,m)  
    MergeSort(A,m+1,e)  
    # merge sorted A[b..m] and A[m+1..e] back into A[b..e]  
    for i in [b,...,e]: B[i] = A[i]  
    c = b  
    d = m+1  
    for in [b,...,e]:  
        if d > e or (c <= m and B[c] < B[d]):  
            A[i] = B[c]  
            c = c + 1  
        else: # d <= e and (c > m or B[c] >= B[d])  
            A[i] = B[d]  
            d = d + 1
```



Unwind (repeated substitution)

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



Prove that T is non-decreasing

See Course Notes, Lemma 3.6 Exercise: Prove the recurrence for binary search is non-decreasing

Prove $T \in O(n \lg n)$ for general case

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



General case

Class of algorithms: partition problem into b *roughly* equal subproblems, solve, and recombine:

$$T(n) = \begin{cases} k & \text{if } n \leq B \\ a_1 T(\lceil n/b \rceil) + a_2 T(\lfloor n/b \rfloor) + f(n) & \text{if } n > B \end{cases}$$

where $b, k > 0$, $a_1, a_2 \geq 0$, and $a = a_1 + a_2 > 0$. $f(n)$ is the cost of splitting and recombining.



Master Theorem

If f from the previous slide has $f \in \theta(n^d)$, then

$$T(n) \in \begin{cases} \theta(n^d) & \text{if } a < b^d \\ \theta(n^d \log n) & \text{if } a = b^d \\ \theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$



Proof sketch

1. Unwind the recurrence, and prove a result for $n = b^k$
2. Prove that T is non-decreasing
3. Extend to all n , similar to MergeSort



Notes