CS 457, Fall 2016

Drexel University, Department of Computer Science

Lecture 4

Today's Lecture

- Methods for proving recurrences
  - Substitution
  - Recursion tree
  - Master theorem
- Quicksort
  - Correctness
  - Running Time

Merge Sort

To sort A[1 .. n], make initial call to MERGE-SORT (A, 1, n).

MERGE-SORT (A, p, r)

1. **if** $p < r$ // Check for base case
2. $q = \lfloor (p + r)/2 \rfloor$ // Divide step
3. MERGE-SORT (A, p, q) // Conquer step.
4. MERGE-SORT (A, $q + 1$, r) // Conquer step.
5. MERGE (A, p, q, r) // Conquer step.

Merging Two Sorted Lists

MERGE (A, p, q, r)

1. $n_1 = q - p + 1$
2. $n_2 = r - q$
3. Create arrays $L[1 \,..\, n_1 + 1]$ and $R[1 \,..\, n_2 + 1]$
4. **for** $i = 1$ **to** n_1
5. $L[i] = A[p + i - 1]$
6. **for** $j = 1$ **to** n_2
7. $R[j] = A[q + j]$
8. $L[n_1 + 1] = \infty$
9. $R[n_2 + 1] = \infty$
10. $i = 1$
11. $j = 1$
12. **for** $k = p$ **to** r
13.   **if** $L[i] \leq R[j]$
14. $A[k] = L[i]$
15. $i = i + 1$
16.   **else**
17. $A[k] = R[j]$
18. $j = j + 1$

<u>Loop Invariant:</u>

At the start of each iteration of the for loop of lines 12-17, the subarray A[p,.. k-1] contains the k-p smallest elements of $L[1 \,..\, n_1+1]$ and $R[1 \,..\, n_2+1]$, in sorted order.

Moreover, L[i] and R[j] are the smallest elements of their arrays that have not been copied back into A.

Things to show about invariant:

1. Initialization
2. Maintenance
3. Termination

Running Time and Recurrence Equations

- Recurrence equation for divide and conquer algorithms:
  - $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$
- Recurrence equation for Merge Sort
  - $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{otherwise} \end{cases}$

Methods for Solving Recurrences

Three methods:

1. Substitution method
   - Guess a bound and use mathematical induction to prove its correctness
2. Recursion-tree method
   - Covert into a tree and measure cost incurred at the various levels
3. Master method
   - Directly provides bounds for recurrences of the form $T(n) = T\left(\frac{n}{b}\right) + f(n)$

Substitution Method

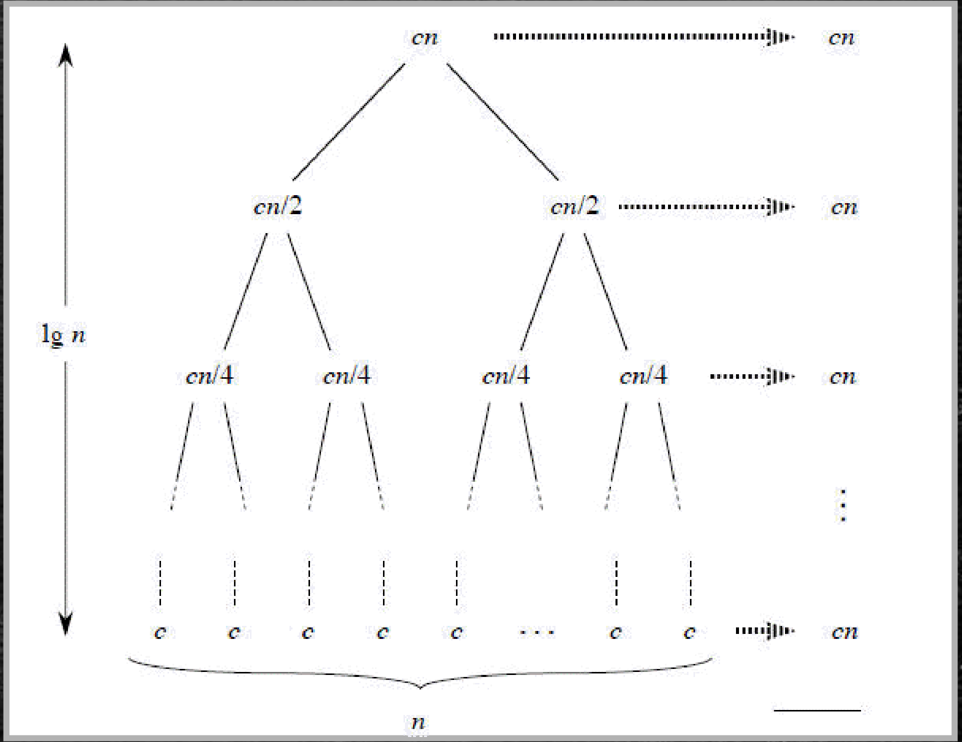
1. Guess the form of the solution
2. Use mathematical induction to find the constants and show that it works

E.g., $T(n) = 2T(\lfloor n/2 \rfloor) + n$

$$T(n) \leq 2[c\lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)] + n \quad \Rightarrow$$
$$T(n) \leq cn\log(n/2) + n \quad \Rightarrow$$
$$T(n) \leq cn\log n - cn\log 2 + n \quad \Rightarrow$$
$$T(n) \leq cn\log n - cn + n \quad \Rightarrow$$
$$T(n) \leq cn\log n \quad \Rightarrow$$

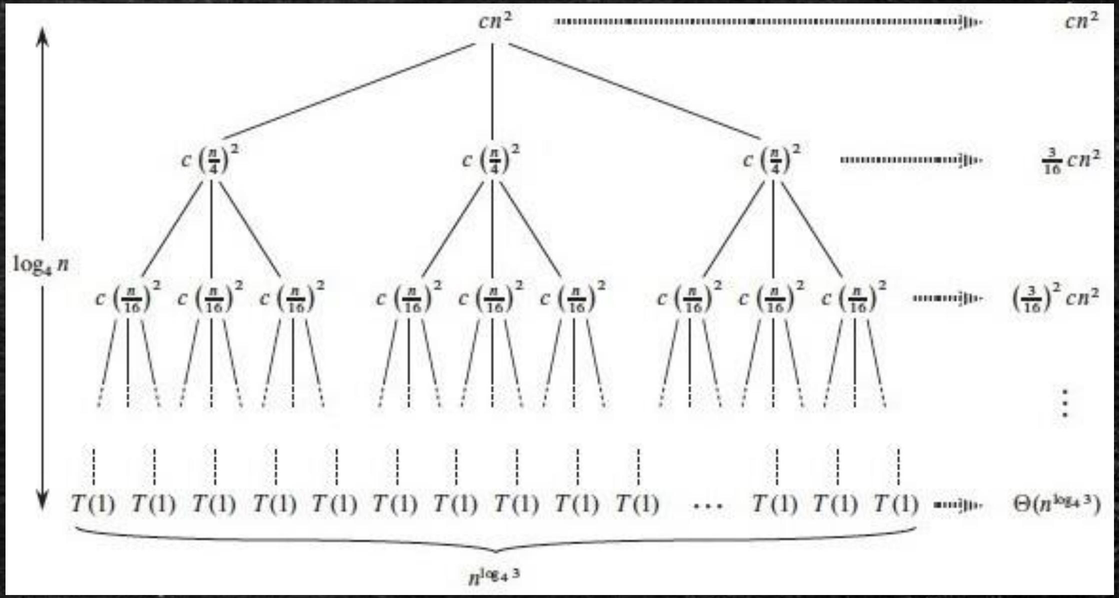
Recursion-Tree Method

- Recurrence equation for Merge Sort
  - $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{otherwise} \end{cases}$



Recursion-Tree Method

- Recurrence equation
  - $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ 3T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + \Theta(n^2) & \text{otherwise} \end{cases}$



Master Theorem

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the non-negative integers by the recurrence:

$$T(n) = aT(n/b) + f(n),$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant ε, then $T(n) = \Theta\left(n^{\log_b a}\right)$
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$
3. If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant ε, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n ,then $T(n) = \Theta(f(n))$

Quicksort

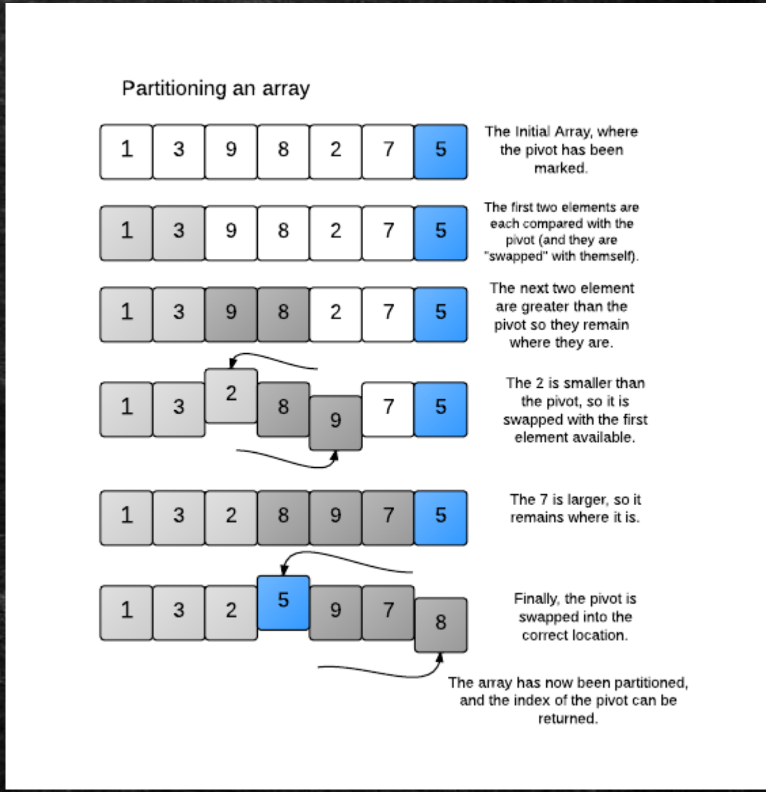
QUICKSORT (A, p, r)

```
1.  if p < r                            // Check for base case
2.      q = PARTITION(A, p, r)          // Divide step
3.      QUICKSORT (A, p, q - 1)         // Conquer step.
4.      QUICKSORT (A, q + 1, r)         // Conquer step.
```

Quicksort

PARTITION (A, p, r)

1. x = A[r]
2. i = p – 1
3. **for** j = p **to** r – 1
4.   **if** A[j] <= x
5.   i = i + 1
6.   exchange A[i] with A[j]
7. exchange A[i+1] with A[r]
8. **return** i+1



Quicksort (Correctness)

PARTITION (A, p, r)

```
1.    x = A[r]
2.    i = p – 1
3.    for j = p to r – 1
4.        if A[ j] <= x
5.            i = i + 1
6.            exchange A[i] with A[j]
7.    exchange A[i+1] with A[r]
8.    return i+1
```

<u>Loop Invariant:</u>

At the beginning of each iteration of the loop of lines 3-6, for any array index k,

1. If $\mathrm{p} \leq \mathrm{k} \leq \mathrm{i}$, then $A[k] \leq x$
2. If $i + 1 \leq k \leq j - 1$, then $A[k] > x$
3. If $k = r$, then $A[k] = x$

Quicksort (Running Time)

QUICKSORT (A, p, r)

```
1.    if p < r                          // Check for base case
2.        q = PARTITION(A, p, r)        // Divide step
3.        QUICKSORT (A, p, q - 1)       // Conquer step.
4.        QUICKSORT (A, q + 1, r)       // Conquer step.
```