Week 7: Recursive Algorithm Analysis

CSC 236:Introduction to the Theory of Computation

Summer 2024

Instructor: Lily

Announcement

- Email note: please include your UTORID
- A3 peer reviews extended to Thursday (maybe the last one)
- A4 due date extended to July 11th
- Midterm 1 results end of this week or start of next
  - Still finishing up oral exams; solutions out at the end of the week
- Midterm 2: 6:00~8:00pm EX 100 (July 17)
  - Covers proof-of-correctness material (weeks 4~7, assignments 3 and 4)
  - Same structure as midterm 1 (5 questions), one sided aid sheet
- Tutorials cover examples. This week: run-time analysis

Quick Sort (Worst Case)

```
def partition(A, i, j):
  # Pre: i, j indices of A (i <= j)
  # Post: index p so that A[k] < A[p] for
  # k = i, …, p-1 and A[l] > A[p] for
  # l = p+1, …, j
  p = i
  pivot = A[j-1]
  for k in range(i, j-1):
    if pivot > A[k]:
      swap(A, p, k)
      p += 1
  swap(A, p, j-1)
  return p
```

```
def quicksort(A, i, j):
  # Pre: list A. i, j indices
  # Post: A is sorted
  if j-i <= 1:
    return
  p = partition(A, i, j)
  quicksort(A, i, p)
  quicksort(A, p+1, j)
```

$$T(n) = \Theta(n) + T(n-1) + T(1)$$

Substitution Method

$$T(n) = \Theta(n) + T(n-1) + T(1)$$

Substitution Method

$$T(n) = \Theta(n) + T(n-1) + T(1)$$

Pitfalls

$$T(n) = n + 2 \cdot T\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$$

Pitfalls

$$T(n) = n + 2 \cdot T\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$$

Quick Sort (Expected Case)

```
def partition(A, i, j):
  # Pre: i, j indices of A (i <= j)
  # Post: index p so that A[k] < A[p] for
  # k = i, …, p-1 and A[l] > A[p] for
  # l = p+1, …, j
  p = i
  pivot = A[j-1]
  for k in range(i, j-1):
    if pivot > A[k]:
      swap(A, p, k)
      p += 1
  swap(A, p, j-1)
  return p
```

```
def quicksort(A, i, j):
  # Pre: list A. i, j indices
  # Post: A is sorted
  if j-i <= 1:
    return
  p = partition(A, i, j)
  quicksort(A, i, p)
  quicksort(A, p+1, j)
```

$$T(n) = \Theta(n) + 2 \cdot T\left(\frac{n}{2}\right)$$

Recurrence Tree Method

$$T(n) = \Theta(n) + 2 \cdot T\left(\frac{n}{2}\right)$$

Master Method

Recurrence relation of the form: $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

Case 1. (Root Heavy) If $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then

$$T(n) = \Theta(f(n))$$

Case 2. (Leaf Heavy) If $f(n) = \mathrm{O}\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then

$$T(n) = \Theta(n^{\log_b a})$$

Case 3. (Balanced) If $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

Root Heavy

(Root Heavy) $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(f(n))$
(Leaf Heavy) $f(n) = \mathrm{O}\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
(Balanced) $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

$$T(n) = 3T\left(\frac{n}{4}\right) + n\log(n)$$

Root Heavy

(Root Heavy) $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(f(n))$
(Leaf Heavy) $f(n) = \mathrm{O}\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
(Balanced) $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

$$T(n) = 3T\left(\frac{n}{4}\right) + n\log(n)$$

Leaf Heavy

(Root Heavy) $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(f(n))$
(Leaf Heavy) $f(n) = O\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
(Balanced) $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

$$T(n) = 9T\left(\frac{n}{3}\right) + n$$

Leaf Heavy

(Root Heavy) $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(f(n))$
(Leaf Heavy) $f(n) = O\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
(Balanced) $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

$$T(n) = 9T\left(\frac{n}{3}\right) + n$$

Balanced

(Root Heavy) $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(f(n))$
(Leaf Heavy) $f(n) = O\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
(Balanced) $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

$$T(n) = T\left(\frac{2n}{3}\right) + 1$$

Balanced

(Root Heavy) $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(f(n))$
(Leaf Heavy) $f(n) = O\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
(Balanced) $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

$$T(n) = T\left(\frac{2n}{3}\right) + 1$$

How about…

(Root Heavy) $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(f(n))$
(Leaf Heavy) $f(n) = O\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
(Balanced) $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

$$T(n) = 2T\left(\frac{n}{2}\right) + n\log n?$$

Now you try!

1. Recall Karatsuba's algorithm from last lecture. Use the Master Method to find its asymptotic running time.
2. Use the Master Method to compute the closed form for the following recurrence relations:

   a) $T(n) = 8T\left(\frac{n}{4}\right) + n^2 \log n$

   b) $T(n) = 3T\left(\frac{n}{9}\right) + \sqrt{n}$

   c) $T(n) = 36T\left(\frac{n}{6}\right) + n^{1.5}$

```
def karatsuba(a, b):
   # Pre: a and b are natural
number with at most n digits
   # Post: returns a*b
if |a| == 1 and |b| == 1:
  return a*b
a1, a2 = a[0..n/2], a[n/2..n]
b1, b2 = b[0..n/2], b[n/2..n]
p1 = karatsuba(a1, b1)
p2 = karatsuba(a1+a2, b1+b2)
p3 = karatsuba(a2, b2)
return p1*10^{n} + (p2-p1-
p3)*10^{n/2} + p3
```

Q1. Recall Karatsuba's algorithm from last lecture. Use the Master Method to find its asymptotic running time.

```
def karatsuba(a, b):
   # Pre: a and b are natural number with at most n digits
   # Post: returns a*b
if |a| == 1 and |b| == 1:
  return a*b
a1, a2 = a[0..n/2], a[n/2..n]
b1, b2 = b[0..n/2], b[n/2..n]
p1 = karatsuba(a1, b1)
p2 = karatsuba(a1+a2, b1+b2)
p3 = karatsuba(a2, b2)
return p1*10^{n} + (p2-p1-p3)*10^{n/2} + p3
```

Q2. Use the Master Method to compute the closed form for following:

a) $T(n) = 8T\left(\frac{n}{4}\right) + n^2 \log n$

b) $T(n) = 3T\left(\frac{n}{9}\right) + \sqrt{n}$

c) $T(n) = 36T\left(\frac{n}{6}\right) + n^{1.5}$

How about…

$$T(n) = 2T\left(\left\lfloor\sqrt{n}\right\rfloor\right) + \log n?$$

How about…

$T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$?

Recap

- Substitution method
- Recurrence tree method (guess closed form)
- Master method + examples
- Karatsuba multiplication
- Quick Sort best and worst case

Next time… finite automaton