Data Structures and Algorithms
CMPSC 465

Midterm Review

- Midterm topics
- Some examples
- Solutions to HW3 D&C problem

Paul Medvedev

Topics (I)

You should be able to:	Examples we've covered
Predict the output of a basic python program	Many code snippets in codeacademy tutorial
Prove correctness of non-recursive algorithm using loop invariants	Examples from lecture/recitation: • Insertion sort • Binary search Example from book: • Merge routine from Merge Sort
Write down the run time of a non-recursive algorithm	Examples from lecture: • Insertion sort
Prove correctness of a recursive algorithm using induction	Examples from lecture: • Merge sort
Write down the run time of a recursive algorithm by setting up a recurrence relation	Examples from lecture: • Merge Sort, Binary Search, Max Subarray Problem
Identify a brute-force algorithm for a problem and give its running time	Examples from lecture: • Max Subarray Problem
Establish asymptotic relationship between functions using 1. first principles 2. known properties	Plenty examples in lectures Exercises: • Piazza post • Homework 2

Topics (II)

You should be able to:	Examples we've covered
Solve recurrence relations using 1. Recursion tree method to form a guess 2. Substitution method to verify a guess 3. Master's Theorem (when it applies)	Plenty examples in lectures Exercises: • Homework 3, Problem 2
Solve a problem using divide & conquer strategy	Examples from lecture: • Merge Sort • Binary Search • Maximum Subarray Problem Exercises: • Median (Hw 3)

- Dynamic programming algorithms will not be included in 1st midterm.
- Focus will be on material covered in lectures and homeworks.
- Sample midterm gives you idea of size and format of exam

Examples

- Examples and solutions in sample midterm
  - already covered in recitations
- Some more examples now...

Example Question: Prove correctness of non-recursive algorithm using loop invariants

```
MERGE(A, p, q, r)
 1  n1 = q − p + 1
 2  n2 = r − q
 3  let L[1 . . n1 + 1] and R[1 . . n2 + 1] be new arrays
 4  for i = 1 to n1
 5      L[i] = A[p + i − 1]
 6  for j = 1 to n2
 7      R[j] = A[q + j]
 8  L[n1 + 1] = ∞
 9  R[n2 + 1] = ∞
10  i = 1
11  j = 1
12  for k = p to r
13      if L[i] ≤ R[j]
14          A[k] = L[i]
15          i = i + 1
16      else A[k] = R[j]
17          j = j + 1
```

Question: Consider the Merge alg on left.

- Write down the output specifications of the alg
- Write down a loop invariant for the for loop on line 12 that can be used to prove correctness
- Prove the initialization, maintenance, and termination properties of this loop invariant

Solution: Given in sec 2.3.1 of textbook.

Example Question: Write down run time of recursive algorithm by setting up a recurrence relation

MERGE-SORT(*A, p, r)*

1. If $p < r$, ▻ *if $p \geq r$, do nothing*
2. q := floor(p+r)/2)
3. Merge-Sort(*A*, p, q)
4. Merge-Sort(*A, q+1,r)*
5. Merge(A,p,q,r)
6. return

Question: Consider the above algorithm. Let T(n) be the time it takes for inputs of length n. Write down a recurrence relation for T(n) and argue why it is correct

Solution: $T(n) = 2T(n/2) + \Theta(n)$. Lines 1 and 2 take $\Theta(1)$ time, lines 3 and 4 take $T(n/2)$ time each, and the merge step takes $\Theta(n)$ time.

Example Question: establish asymptotic relationships using first principles

$$2n^2 = o(n^3)$$

Question: Give a value of n_0 which proves the above statement.

Know your asymptotic notation definitions

Notation	… means …	Think…	E.g.
$f(n)=O(n)$	$\exists c > 0, n_0 > 0, \forall n \geq n_0$: $0 \leq f(n) \leq cg(n)$	Upper bound "≤"	$100n^2 = O(n^3)$
$f(n)=\Omega(g(n))$	$\exists c > 0, n_0 > 0, \forall n \geq n_0$: $0 \leq cg(n) \leq f(n)$	Lower bound "≥"	$2^n = \Omega(n^{100})$
$f(n)=\Theta(g(n))$	$f(n) \in O(g(n))$ **and** $f(n) \in \Omega(g(n))$	Tight bound "="	$\log(n!) = \Theta(n \log n)$
$f(n)=o(g(n))$	$\forall c > 0, \exists n_0 > 0, \forall n \geq n_0$: $0 \leq f(n) < cg(n)$	"<"	$n^2 = o(2^n)$
$f(n)=\omega(g(n))$	$\forall c > 0, \exists n_0 > 0, \forall n \geq n_0$: $0 \leq cg(n) < f(n)$	">"	$n^2 = \omega(\log n)$

Example Question: establish asymptotic relationships using first principles

$$2n^2 = o(n^3)$$

Question: Give a value of n_0 which proves the above statement.

Solution: Recall definition

$o(g(n)) = \{ f(n)$: for any constant $c > 0$,
there is a constant $n_0 > 0$
such that $0 \leq f(n) < cg(n)$
for all $n \geq n_0 \}$

Apply definition: $0 \leq 2n^2 < cn^3$

Solve for n: $n > \frac{2}{c}$

Let $n_0 = \frac{2}{c} + 1$

Example Question: Establish asymptotic relationship using known properties

True or False?

1) $n^2 - 5n - 100 = O(n)$
2) $n^3 + 10n^2 + 125 = \omega(n)$
3) $n^2 + O(n) = O(n^2)$
4) $2^{n+1} = O(2^n)$
5) $2^{5n} = O(2^n)$
6) $\log(n^2) = O(\log(n))$

Sort by asymptotic order:

note ☆

Piazza exercise from lecture 08/31

(Transcription of a photo of the slide -- *thanks Collin Forsyth!*)

Exercise: Sort by asymptotic order

1. $\sqrt{n}$
2. $n \log n$
3. $n^{1/\log n}$
4. n^2
5. $2^{\log n}$
6. $2^{\log_3 n}$
7. $2^{n/100}$
8. $\log(n!)$
9. $n\sqrt{n}$

Know your log and exponential functions

For all real $a > 0$, m, and n, we have the following identities:

$$
\begin{aligned}
a^0 &= 1 , \\
a^1 &= a , \\
a^{-1} &= 1/a , \\
(a^m)^n &= a^{mn} , \\
(a^m)^n &= (a^n)^m , \\
a^m a^n &= a^{m+n} .
\end{aligned}
$$

CLRS, page 55

For all real $a > 0$, $b > 0$, $c > 0$, and n,

$$
\begin{aligned}
a &= b^{\log_b a} , \\
\log_c(ab) &= \log_c a + \log_c b , \\
\log_b a^n &= n \log_b a , \\
\log_b a &= \frac{\log_c a}{\log_c b} , \\
\log_b(1/a) &= -\log_b a , \\
\log_b a &= \frac{1}{\log_a b} , \\
a^{\log_b c} &= c^{\log_b a} ,
\end{aligned}
$$

where, in each equation above, logarithm bases are not 1.

CLRS, page 56

Known relationships of common functions

- **Polynomials.** $a_0 + a_1 n + \ldots + a_d n^d$ is $\Theta(n^d)$ if $a_d > 0$.

- **Logarithms.** $\log_a n = \Theta(\log_b n)$ for all constants a, b > 0.

  ↑ can avoid specifying the base

  ↓ log grows slower than every polynomial

  For every $x > 0$, $\log n = O(n^x)$.

- **Exponentials.** For all $r > 1$ and all $d > 0$, $n^d = O(r^n)$.

  ↓ Every exponential grows faster than every polynomial

- **Factorial.**

  $$\log(n)! = \Theta(n \log n)$$

  factorial grows faster than every exponential

Example Question: Establish asymptotic relationship using known properties

True or False?

1) $n^2 - 5n - 100 = O(n)$
2) $n^3 + 10n^2 + 125 = \omega(n)$
3) $n^2 + O(n) = O(n^2)$
4) $2^{n+1} = O(2^n)$
5) $2^{5n} = O(2^n)$
6) $\log(n^2) = O(\log(n))$

Example Question: Establish asymptotic relationship using known properties

note

Piazza exercise from lecture 08/31

(Transcription of a photo of the slide -- *thanks Collin Forsyth!*)

Exercise: Sort by asymptotic order

1. $\sqrt{n}$
2. $n \log n$
3. $n^{1/\log n}$
4. n^2
5. $2^{\log n}$
6. $2^{\log_2 n}$
7. $2^{n/100}$
8. $\log(n!)$
9. $n\sqrt{n}$

Paul Medvedev 17 days ago
Some hints:

3: $n^{1/\log n} = (2^{\log n})^{1/\log n} = 2^{\log n/\log n} = 2$

5: $2^{\log n} = n$

6: $2^{\log_2 n} = n^{\log_2 2}$

7: $2^{n/100} = (2^{1/100})^n$

Paul Medvedev 7 days ago $2 \in o(\sqrt{n}) \in o(n^{0.63}) \in o(n) \in o(n \log n) \in o(n^{1.5}) \in n^2 \in o(2^n)$

Example Question: Solve recurrence relations using recursion trees

Consider the recurrence $T(n) = 5T(n/3) + n$

a) Draw the recursion tree (at least two levels)
b) What is the depth of the tree?
c) What is the amount of work done at level i?
d) What is the total amount of work done?

Know your geometric series

$$\sum_{i=0}^{n} x^i = \frac{x^{n+1} - 1}{x - 1} = \Theta(x^n)$$

Example Question: Solve recurrence relations using the substitution method

Question: Consider the recurrence $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + 2n$

Use the substitution method to show that $T(n)=\Omega(n \log n)$

Solution: We will prove $T(n) \geq cn \log n$ for some c. We first prove the induction step.

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + 2n \geq \frac{cn}{4}\log\frac{n}{4} + \frac{3cn}{4}\log\left(\frac{3n}{4}\right) + 2n$$

After some math (omitted), we get that $T(n) \geq cn \log n$ is true whenever $c \leq \frac{2}{2-\left(\frac{3}{4}\right)\log 3}$ and $n \geq 1$. In particular, we can set c=1

Example Question: Solve recurrence relations using Master's theorem

Use the Master Theorem, if possible, to solve each of the recurrences below. If the Master Theorem does not apply, state why. If it does, state which of the three cases you are using and give the asymptotic running time for $T(n)$.

i. $T(n) = 27T(n/3) + n^3$
ii. $T(n) = 7T(n/3) + n^2$
iii. $T(n) = 7T(n/2) + n^2$
iv. $T(n) = 3T(n/3) + n/\log n$
v. $T(n) = 3T(n/9) + \sqrt{n}$
vi. $T(n) = 3T(n/4) + n\log n$
vii. $T(n) = 2T(n/5) + \log^2 n$
viii. $T(n) = 8T(n/2) + 2^n$

Know your master theorem

$T(n) = a\ T(n/b) + f(n)$

1. $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$.
   → $T(n) = \Theta(n^{\log_b a})$
2. $f(n) = \Theta(n^{\log_b a} \lg^k n)$ for some constant $k \geq 0$.
   → $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$
3. $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$,
   → $T(n) = \Theta(f(n))$

Use the Master Theorem, if possible, to solve each of the recurrences below. If the Master Theorem does not apply, state why. If it does, state which of the three cases you are using and give the asymptotic running time for $T(n)$.

i. $T(n) = 27T(n/3) + n^3$
ii. $T(n) = 7T(n/3) + n^2$
iii. $T(n) = 7T(n/2) + n^2$
iv. $T(n) = 3T(n/3) + n/\log n$
v. $T(n) = 3T(n/9) + \sqrt{n}$
vi. $T(n) = 3T(n/4) + n\log n$
vii. $T(n) = 2T(n/5) + \log^2 n$
viii. $T(n) = 8T(n/2) + 2^n$

Example Question: Solve a problem using divide & conquer strategy

3. Suppose you are given two sorted arrays, each with n elements. There are $2n$ values in total, and you may assume that no two values are the same. You would like to determine the median of this set of $2n$ values. We define the median as the n^{th} smallest value.

a) Consider a divide and conquer solution given by the pseudocode below. Please fill in the three missing lines so that the algorithm is correct

```
MEDIAN(A, A_start, A_end, B, B_start, B_end)
1  ▷ returns the median of A[A_start..A_end] and B[B_start..B_end]
2  ▷ to find the median of A and B, you would call MEDIAN(A, 1, A.length, B, 1, B.length)
3  midA ← A_start + ⌊(A_end − A_start)/2⌋
4  midB ← B_start + ⌊(B_end − B_start)/2⌋
5  if A_start = A_end ▷ only one element in each array
6     then
7          return
8  if A[midA] > A[midB]
9     then
10         return MEDIAN(                              ).
11    else
12         return MEDIAN(                              ).
```

Example Question: Solve a problem using divide & conquer strategy

3. Suppose you are given two sorted arrays, each with n elements. There are $2n$ values in total, and you may assume that no two values are the same. You would like to determine the median of this set of $2n$ values. We define the median as the n^{th} smallest value.

a) Consider a divide and conquer solution given by the pseudocode below. Please fill in the three missing lines so that the algorithm is correct

```
MEDIAN(A, A_start, A_end, B, B_start, B_end)
1  ▷ returns the median of A[A_start..A_end] and B[B_start..B_end]
2  ▷ to find the median of A and B, you would call MEDIAN(A, 1, A.length, B, 1, B.length)
3  midA ← A_start + ⌊(A_end − A_start)/2⌋
4  midB ← B_start + ⌊(B_end − B_start)/2⌋
5  if A_start = A_end ▷ only one element in each array
6     then
7          return   min(A[A_start], B[B_start])
8  if A[midA] > A[midB]
9     then
10         return MEDIAN(A, A_start, midA, B, B_start + ⌈(B_end − B_start)/2⌉, B_end).
11    else
12         return MEDIAN(A, A_start + ⌈(A_end − A_start)/2⌉, A_end), B, B_start, midB ).
```

Example Question: Solve a problem using divide & conquer strategy

3. Suppose you are given two sorted arrays, each with n elements. There are $2n$ values in total, and you may assume that no two values are the same. You would like to determine the median of this set of $2n$ values. We define the median as the n^{th} smallest value.

a) Consider a divide and conquer solution given by the pseudocode below. Please fill in the three missing lines so that the algorithm is correct

b) Argue why the algorithm is correct

c) Write down and solve the recurrence for the running time