CS3000: Algorithms & Data
Jonathan Ullman

Lecture 10:
- Midterm I Review

Oct 9, 2018

- HW3 solutions posted Wed
- HW3 grades out Thu
- HW4 due Fri
- HW5 will be due 10/26

Midterm 1 Review

Midterm I Topics

- Fundamentals:
  - Induction
  - Asymptotics
  - Recurrences
- Stable Matching
- Divide and Conquer
- Dynamic Programming

Topics: Induction

$T(n) = 2T(\frac{n}{2}) + Cn$

$T(n) = Cn\log_2(2n)$

- Proof by Induction:
    - Mathematical formulas, e.g. $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
    - Spot the bug
    - Solutions to recurrences
    - Correctness of divide-and-conquer algorithms

- Good way to study:
    - Lehman-Leighton-Meyer, *Mathematics for CS*
    - Review divide-and-conquer in Kleinberg-Tardos

Practice Question: Induction

- Suppose you have an unlimited supply of 3 and 7 cent coins, prove by induction that you can make any amount $n \geq 12$.

  - If I can make n, I can make $n+3$
  - If I can make n, $n+1$, $n+2$, I can make any $m \geq n$

$12 = 4 \times 3$

$13 = 2 \times 3 + 1 \times 7$

$14 = 0 \times 3 + 2 \times 7$

Inductive Hypotheses:

$H(n)$ = Can make change for any $12 \le k \le n$

The stmt I want to prove is equivalent

$\forall n \geq 12$, $H(n)$ is true

Base Cases: $H(12)$, $H(13)$, $H(14)$ are true (by inspection)

Inductive Step:

(To prove) If $H(n)$ is true for some $n \geq 14$ then $H(n+1)$ is true

- By $H(n)$, I can make change for $12 \le k \le n$
- If I can make change for $n-2$ then I can make change for $n+1$

  ($n-2$ → ≥ 12)

  ∴ by $H(n)$, I can make change for $n+1$

  ∴ $H(n+1)$ is true

Topics: Asymptotics

- Asymptotic Notation
  - $o, O, \omega, \Omega, \Theta$ (Definitions)
  - Relationships between common function types

- Good way to study:
  - Kleinberg-Tardos Chapter 2

Topics: Asymptotics

"similar" $\lim_{n \to \infty} \frac{f(n)}{g(n)} < \infty$

Notation	… means …	Think…	E.g.
f(n)=O(n)	$\exists c > 0, n_0 > 0, \forall n \geq n_0$: $0 \leq f(n) \leq cg(n)$	At most "≤"	$100n^2 = O(n^3)$
f(n)=Ω(g(n))	$\exists c > 0, n_0 > 0, \forall n \geq n_0$: $0 \leq cg(n) \leq f(n)$	At least "≥"	$2^n = \Omega(n^{100})$
f(n)=Θ(g(n))	$f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$	Equals "="	$\log(n!) = \Theta(n \log n)$
f(n)=o(g(n))	$\forall c > 0, \exists n_0 > 0, \forall n \geq n_0$: $0 \leq f(n) < cg(n)$	Less than "<"	$n^2 = o(2^n)$
f(n)=ω(g(n))	$\forall c > 0, \exists n_0 > 0, \forall n \geq n_0$: $0 \leq cg(n) < f(n)$	Greater than ">"	$n^2 = \omega(\log n)$

Topics: Asymptotics

- **Constant factors can be ignored**
  - $\forall C > 0 \quad Cn = O(n)$
- **Smaller exponents are Big-Oh of larger exponents**
  - $\forall a > b \quad n^b = O(n^a)$
- **Any logarithm is Big-Oh of any polynomial**
  - $\forall a, \varepsilon > 0 \quad \log_2^a n = O(n^\varepsilon)$
- **Any polynomial is Big-Oh of any exponential**
  - $\forall\ a > 0, b > 1 \quad n^a = O(b^n)$
- **Lower order terms can be dropped**
  - $n^2 + n^{3/2} + n = O(n^2)$

Practice Question: Asymptotics

$2^{3\log_2\log_2 n}$ $\sum_{i=1}^{n} i$ $n^2\log_2(n)$ $n^{\log_2 7}$ $8^{\log_2 n}$ $2^{\log_2^2(n)}$

- Put these functions in order so that $f_i = O(f_{i+1})$
  - ✓ $n^{\log_2 7}$ $n^{2+\varepsilon}$ $\varepsilon \in (0,1)$
  - ✓ $8^{\log_2 n} = 2^{(\log_2 8)(\log_2 n)} = n^{\log_2 8} = n^3$
  - ✓ $2^{3\log_2\log_2 n}$
  - $2^{(\log_2 n)^2} = \left(2^{\log_2 n}\right)^{\log_2 n} = n^{\log_2 n}$ $\left(\text{if } n \geq 8,\ n^{\log_2 n} \geq n^3\right)$
  - ✓ $\sum_{i=1}^{n} i = \Theta(n^2)$
  - ✓ $n^2\log_2 n$

$$2^{3\log_2\log_2 n} = \left(2^{\log_2(\log_2 n)}\right)^3 = \left(\log_2(n)\right)^3$$

Practice Question: Asymptotics

- Suppose $f_1 = O(g)$ and $f_2 = O(g)$.
  Prove that $f_1 + 4f_2 = O(g)$.

Topics: Recurrences

- Recurrences
  - Representing running time by a recurrence
  - Solving common recurrences $T(n) = T(\frac{7n}{10}) + T(\frac{n}{5}) + cn$
  - Master Theorem $T(n) = a \cdot T(\frac{n}{b}) + cn^d$

- Good way to study:
  - Erickson book
  - Kleinberg-Tardos divide-and-conquer chapter

Practice Question: Recurrences

```
F(n):
  For i = 1,…,n²: Print "Hi"
  For i = 1,…,3: F(n/3)
```

$] \; O(n^2)$

$] \; 3 \times T(\frac{n}{3})$

- Write a recurrence for the running time of this algorithm. Write the asymptotic running time given by the recurrence.

$$T(n) = 3 \times T\left(\frac{n}{3}\right) + Cn^2$$

$a = 3$

$b = 3$

$d = 2$

$$\frac{a}{b^d} = \frac{3}{3^2} < 1$$

$$T(n) = \Theta(n^2)$$

Topics: Recurrences

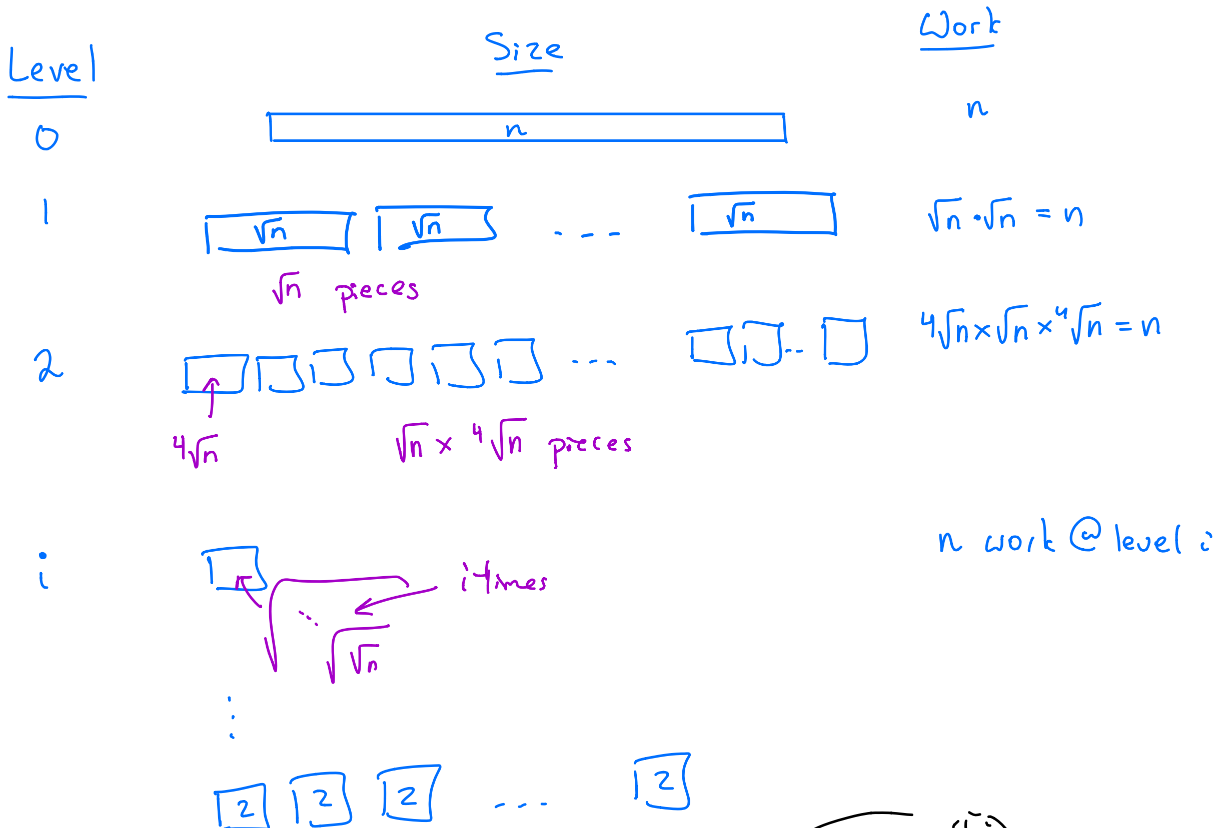
Assume $\sqrt{n}$ is an integer

- Consder the recurrence $T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n$ with ~~$T(1) = 1$~~. Solve using a recursion tree.

  $T(2) = 1$

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

$T(n) = \sqrt{n}\, T(\sqrt{n}) + n$



$\underbrace{\sqrt{\cdots \sqrt{\sqrt{n}}}}_{i \text{ times}} = 2$

$\log_2\left(n^{\frac{1}{2^i}}\right)$

$\frac{1}{2^i} \cdot \log_2 n$

$n^{1/2^i} = 2$

$\frac{1}{2^i} \cdot \log_2 n = 1$

$i = \log_2 \log_2 n$

(# of levels is $\log_2 \log_2 n$)

(n work per level)

$= n \log_2 \log_2 n$

Topics: Stable Matching

- Stable Matching
  - Definition of a stable matching
  - The Gale-Shapley algorithm
  - Consequences of the Gale-Shapley algorithm

- Good way to study:
  - Kleinberg-Tardos

Practice Question: Stable Matching

- Give an example of 3 doctors and 3 hospitals such that there exists a stable matching in which every hospital gets it last choice of doctor

Hospitals

	1st	2nd	3rd
A	Z	y	X
B			
C			

Doctors

	1st	2nd	3rd
X	A	C	B
y			
Z			

Topics: Divide-and-Conquer

- Divide-and-Conquer
  - Writing pseudocode
  - Proving correctness by induction
  - Analyzing running time via recurrences
- Examples we've studied:
  - Mergesort, Binary Search, Karatsuba's, Selection
- Good way to study:
  - Example problems from Kleinberg-Tardos or Erickson
  - Practice, practice, practice!

Practice Question: Divide-and-Conquer

Problem 3. *Divide-and-Conquer*

Suppose you have two sorted arrays $A[1,\ldots,n]$ and $B[1,\ldots,n]$ of equal length, containing $2n$ numbers in total. Design an $O(\log n)$ time divide-and-conquer algorithm that finds the *median* of these $2n$ numbers in A and B. We use the convention that the median of an *odd*-length sorted list $C[1,\ldots,2m+1]$ is $C[m+1]$ and the median of an *even*-length sorted list $C[1,\ldots,2m]$ is $C[m]$.

Example: Suppose $n = 5$, $A = [1,4,7,9,19]$, and $B = [2,5,12,18,20]$, then the combined sorted list is $C = [1,2,4,5,7,9,12,18,19,20]$, whose median is $C[5] = 7$.

Practice Question: Divide-and-Conquer

- Describe your algorithm in pseudocode

Practice Question: Divide-and-Conquer

- Prove by induction that the algorithm is correct

Practice Question: Divide-and-Conquer

- Analyze your algorithm's running time by writing a recurrence

Topics: Dynamic Programming

Interval Sched, Segmented LS, Knapsack, Edit Dist, RNA Folding

- Dynamic Programming
  - Identify sub-problems
  - Write a recurrence, $OPT(n) = \max\{v_n + OPT(n-6), OPT(n-1)\}$
  - Fill the dynamic programming table
  - Find the optimal solution
  - Analyze running time

- Good way to study:
  - Example problems from Kleinberg-Tardos or Erickson
  - Practice, practice, practice!

Practice Question: DP

Problem 2. *Dynamic Programming*

The dark lord Sauron loves to destroy the kingdoms of Middle Earth. But he just can't catch a break, and is always eventually defeated. After a defeat, he requires three epochs to rebuild his strength and once again rise to destroy the kingdoms of Middle Earth. In this problem, you will help Sauron decide in which epochs to rise and destroy the kingdoms of Middle Earth.

The input to the algorithm consists of the numbers $x_1, \ldots, x_n$ representing the number of kingdoms in each epoch. If Sauron rises in epoch i then he will destroy all x_i kingdoms, but will not be able to rise again during epochs $i+1, i+2$, or $i+3$. We call a set $S \subseteq \{1, \ldots, n\}$ of epochs *valid* if it satisfies this constraint that $|i-j| \geq 4$ for all $i, j \in S$, and its *value* is $\sum_{i \in S} x_i$. You will design an algorithm that outputs a valid set of epochs with the maximum possible value.

Example: Suppose there are $(1, 7, 8, 2, 6, 3)$ kingdoms of Middle Earth in epochs $1, \ldots, 6$. Then the optimal set of epochs for Sauron to rise up and destroy the kingdoms of Middle Earth is $S = \{2, 6\}$, during which he destroys 10 kingdoms, 7 in the 2nd epoch and 3 in the 6th epoch.

Practice Question: Divide-and-Conquer

- What is the set of subproblems you will use?

- Give a recurrence for this problem.

Practice Question: Divide-and-Conquer

- Give a pseudocode description of your algorithm

Practice Question: Divide-and-Conquer

- Analyze the running time and space usage

Practice Question

- Design an $O(n)$-time algorithm that takes an array $A[1:n]$ and returns a sorted array containing the smallest $\sqrt{n}$ elements of A

Practice Question

- Consider the following sorting algorithm

```
A[1:n] is a global array
SillySort(1,n):
  if (n <= 3): put A in order
  else:
    SillySort(1,2n/3)
    SillySort(n/3,n)
    SillySort(1,2n/3)
```

- Prove that it is correct
- Analyze its running time

Practice Question

```
A[1:n] is a global array
SillySort(1,n):
  if (n <= 3): put A in order
  else:
    SillySort(1,2n/3)
    SillySort(n/3,n)
    SillySort(1,2n/3)
```

Practice Question

- Consider the following sorting algorithm

```
A[1:n] is a global array
SillySort(1,n):
  if (n <= 3): put A in order
  else:
    SillySort(1,2n/3)
    SillySort(n/3,n)
    SillySort(1,2n/3)
```

- Prove that it is correct
- Analyze its running time