COE428 Lecture Notes Week 5 (Feb 6—10, 2017)

Table of Contents

Announcements

- **Repeat**: Counselling hours (ENG-449): Mon 1:00pm—2:00 pm, Fri 1pm—2:00pm.
- **Repeat:** Midterm: Wednesday, March 8, 2017

Topics (from course outline)

The following table shows the topics for this course week by week.

The topics in **bold** is for **this** week.

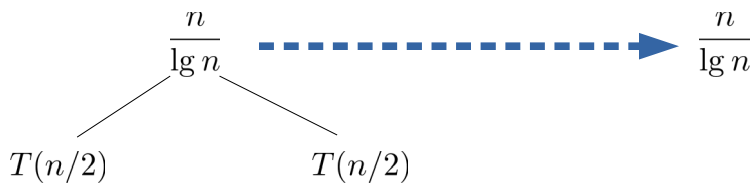
The topics in grey have been covered.

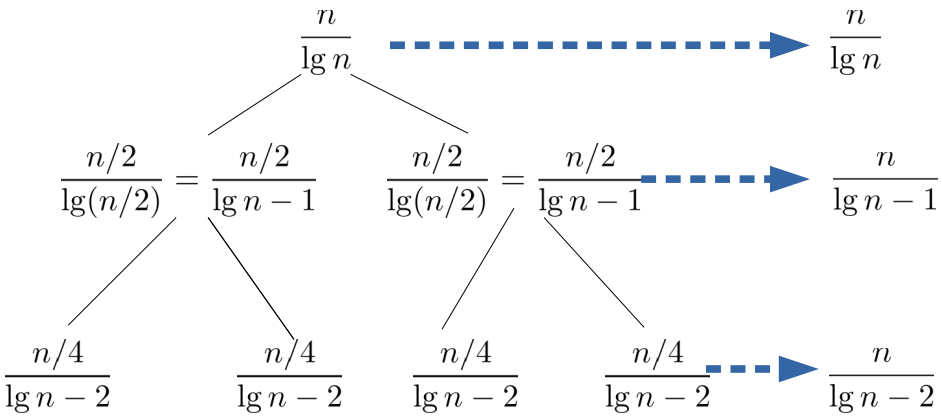
Other topics are for the future....

Week	Date	Topics
1	Jan 9	Introduction. Course overview. Intro to algorithms.
2	Jan 16	Analyzing and designing algorithms. Recursion.
3	Jan 23	Complexity analysis.
4	Jan 30	Recurrence equations. *Data Structures.*
5	Feb 6	**Stacks and Queues.**
6	Feb 13	Heapsort. Hashing.
	Feb 20	*Study week.*
7	Feb 27	Trees and Priority Queues.
8	March 6	Binary Search Trees (BST).
9	March13	Balanced BSTs (including Red-Black Trees)
10	March 20	Graphs.
11	March 27	Elementary graph algorithms.
12	April 3	Elementary graph algorithms. (continued)
13	April 20	Review

Answers to last week's questions

1. Draw a recursion tree for $T(n) = 2T(n/2) + \frac{n}{\lg n}$ for $n > 1$. Find a tight Big-O upper bound.





- Continuing, we get: $T(n) = n\left(\frac{1}{\lg n} + \frac{1}{\lg n - 1} + \frac{1}{\lg n - 2} + \cdots + \frac{1}{1}\right)$
- Noting that the depth of the tree is $\lg n$, we get $T(n) = n \sum_{i=0}^{\lg n} \frac{1}{\lg n - i}$
- The summation is the *Harmonic Series* which which approaches $\ln \lg n$ for large n.
- Consequently, $T(n) = \Theta(n \log \log n)$.

2. Draw a recursion tree for $T(n) = \sqrt{n}T(\sqrt{n}) + n$ for $n \geq 2$ and determine its $\Theta()$ complexity. (Hint: assume $n = 2^{2^m}$.)

n		n
$\sqrt{n}$	($\sqrt{n}$ times)	n
$n^{1/4}$	($n^{3/4}$ times)	n

- Each level on the recursion tree contributes *n*. Since there are $\lg \lg n$ levels, we obtain :
- $T(n) = \Theta(n \log \log n)$

Stacks and Queues

Some basic data structures support only 2 operations:

- *add* and *remove* (aka *delete*)

The most important data structures like this are:

1. Stacks
2. Queues
3. Priority Queues

Stacks

- A stack is a "Last In, First Out" (LIFO) data structure.
- Adding an element to a stack is usually called "push".
- Removing an element is usually called "pop"
- The "top of the stack" is where elements are pushed to and popped from.
- For example, if we perform the operations push(5) followed by push(3), the top of the stack contains the last value pushed, i.e. 3 in this case. Below it on the stack is the value 5.
- Uses of stacks:
  - Reversing
  - Verifying "balance"
  - Almost all computers implement a Stack in hardware. Local variables are placed on the stack and parameters and passed on the Stack.

Queues

- Queues are "First In, First Out" (FIFO) data structures.
- Unlike a Stack where all operations are at the "top of stack", a queue has two ends called the "tail" (or "rear") and the "head" (or "front").
- Items are added at the tail and removed at the head.
- Adding to a queue is usually called "enqueue" and removing from a queue is called "dequeue".

Priority Queues

- In a “Priority Queue” each item has a *priority* arranged such that:
  - deleting removes the item with the highest priority for a MaxQueue (or the minimum value for a MinQueue)
  - adding modifies the data structure so that deletion is efficient.
- Priority Queues will be examined next week.

Implementation of Stacks and Queues

Stack using an array

Stack using a linked list

Queue using an array

Queue using a linked list

Exercises

1.

Questions

1.

Suggested problems

CLRS: 10-1

My book: 6.2, 6.5, 6.9

References (text book and online)

- *CLRS:* Chapter 10.1
- kclowes book: Chapter 6