6.006 Review

9/3

Way of solving a problem

time, space, energy

Document distance

How to review
- Class notes ← started
- Final notes
- Textbook ← prob best

Input + Output

Correct = halts w/ correct output

Dual purpose of review

Google

6.046

Data structure

2

Insertion Sort

Start w/ empty left hand
+ cards face down

remove one card at a time
+ insert in proper position

Officially in place
- just swaps order
- but can also push over

remember what sudo code is saying

↑ ↑ ↑

3rd 2nd moves 1st

← so alg goes that way till it works

Really understand what is going on

③

Worst Case running time

Look at each step

$\Theta(n^2)$ since for each # each could be moved

So like 9 8 7 6

¿So log n

No for some reason n^2

∴ Since the first one is n at worse

Con:

9 8 7 6

6 9 8 7

6 7 9 8

(4)

I did it wrong

9 8 7 6   1

8 9 7 6   2

7 8 9 6   3   ← This is n

6 7 8 9

ln is split in 2 each time - which it is not

; So I guess n^2

Grows quadratically

So input size 4 worst case is $3+2+1=6$

5    $4+3+2+1=10$

6    $5+10=15$

but that is growing at n

5

Yeah n would grow at 1

n^2 grows at n each new step

? Correct ?

$4^2 = 16$
$5^2 = 25$? doesn't match

but $\frac{n}{2} \cdot n$ is still n^2

Good!

Divide + Conquer

Divide

Conquer

Combine

ie Merge sort

splits up all the way
, then combines
in recurrence

$$T(n) = \begin{cases} \Theta(1) & n=1 \\ 2T(n/2) + \Theta(n) & n > 1 \end{cases}$$

So $\ln(n) + 1$ levels each costing cn

So $\Theta(n \lg n)$

⑥

$\Theta(g(n)) = \quad 0 \le c_1 g(n) \le f(n) \le c_2 g(n)$

for $n \ge n_0$

So middle
- asy tight

$O(g(n))$ = asy upper bound

Ω = asy lower bound

o = upper

ω = lower

Of course

$a^0 = 1$

$a^1 = a$

$a^{-1} = 1/a$

$(a^m)^n = a^{m \cdot n} = (a^n)^m$

$a^m a^n = a^{m+n}$

at least I know that now

7

Master Theorm

$\cancel{T(n)}$ $T(n) = aT(n/b) + f(n)$

1. $f(n) = O(n^{\lg_b a - \epsilon})$

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

2. $f(n) = \Theta(n^{\lg_b a}) \to T(n) = \Theta(n^{\lg_b a})$

3. $\Omega(n^{\lg_b a + \epsilon})$ and $af(\frac{n}{b}) \leq cf(n)$

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

what was the easy version

1. $9T(n/3) + n$

   $n^{\lg_3 9} = n^2$ so $\epsilon = 1$ so $T(n) = \Theta(n^2)$

$\cancel{\lg}$ $\lg_2 16 = 4$

$\underbrace{4^2}_{\text{what is this}} = 16$

8

2. $T(n) = 3\,T(n/4) + n\lg n$

$n^{\log_4 3} = \Theta(n^{.793})$

So Case 3

$3\left(\frac{n}{4}\right)\lg\left(\frac{n}{4}\right) \le \left(\frac{3}{4}\right) n\lg n = c\,f(n)$

$c = \frac{3}{4}$

So $\Theta(n\lg n)$

3. $T(n) = 2\,T(n+2) + n\lg n$

$n^{\lg_2 2} = n$

n is not polonomically larger

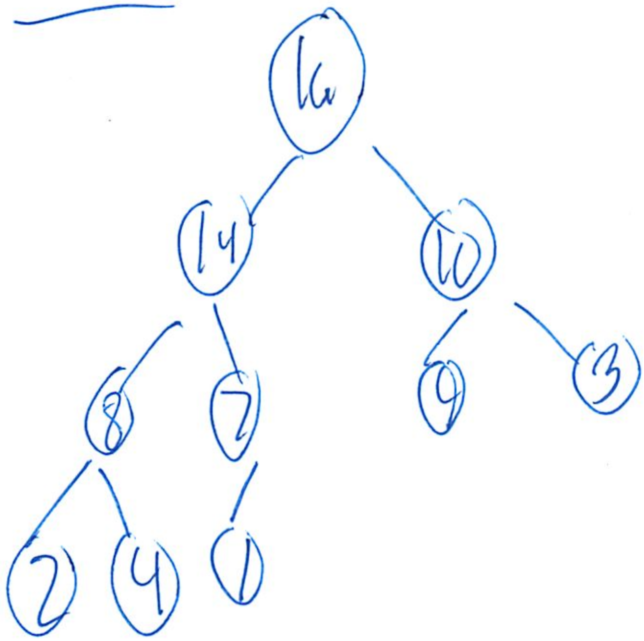
I don't get what is happening

Need to find the simple version

Do later

9

Heaps



Parent (i)

return $\lfloor i/2 \rfloor$

left (i) return $2i$

right (i) return $2i+1$

min or max heap

$A[\text{Parent}(i)] \leq A[i]$

Parent $\lfloor i/2 \rfloor$ must be less

$A[\text{Parent}(i)] \geq A[i]$

parent $\lfloor i/2 \rfloor$ must be greater

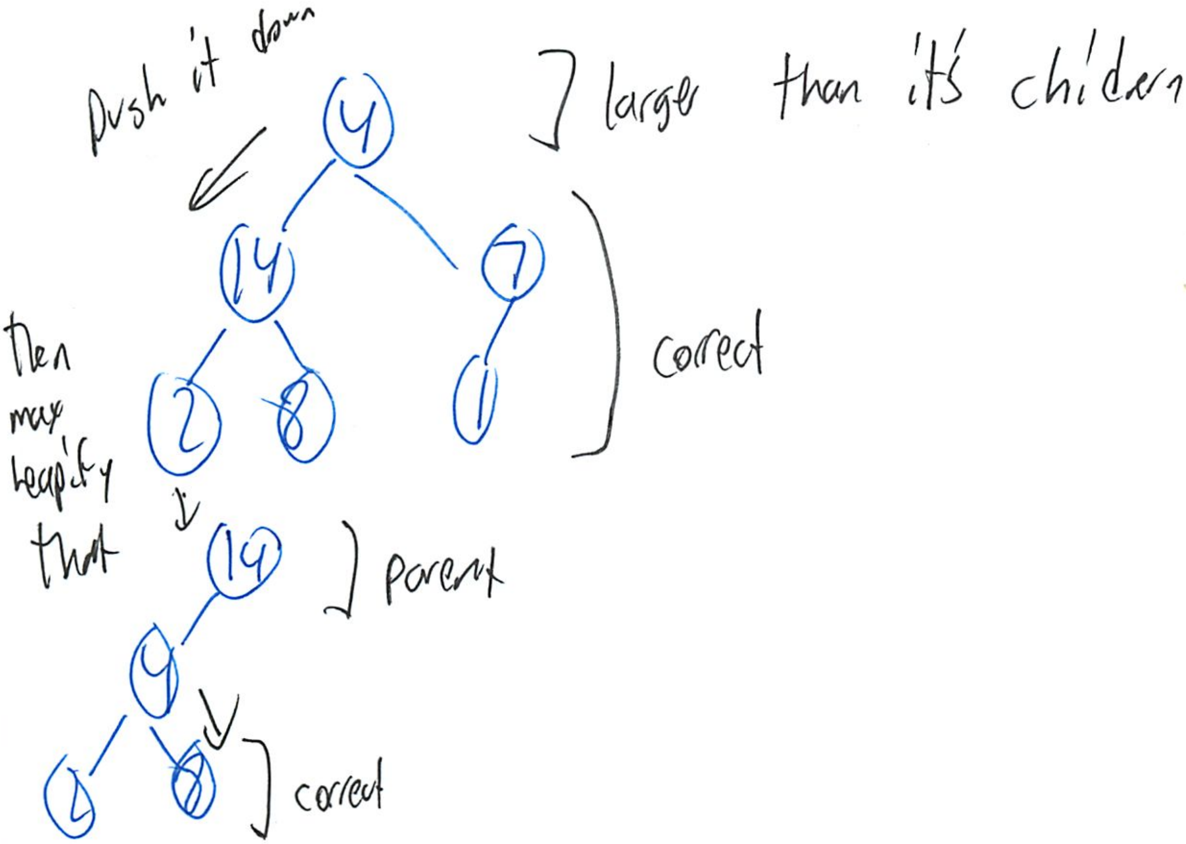
10

Max Heapify

Need to mantain property

assumes left(i) right(i) is correct

but A[i] might be smaller than children



$$T(n) = O(\lg n)$$

$$= O(h)$$

since h is lg n

(11)

Build max heap

build from bottom up

run max heapify on each

from $A[(\lfloor n/2 \rfloor + 1) \ldots n]$

(all the bottom level

Heap sort

builds max heap $A[1 \ldots n]$

So $O(n \lg n)$

Quicksort usually beats though

Priority queue

data structure

insert()

max() – in $O(1)$

extract max()

increase key ()

extract - returns max
and then removes it
and then max heapifies

increase key
exchanges w/ parent

heap insert
calls heap increase key

Quick sort

Often the best pratical choice $\Theta(n \lg n)$
though worst case $\Theta(n^2)$

quick sort half after partioning

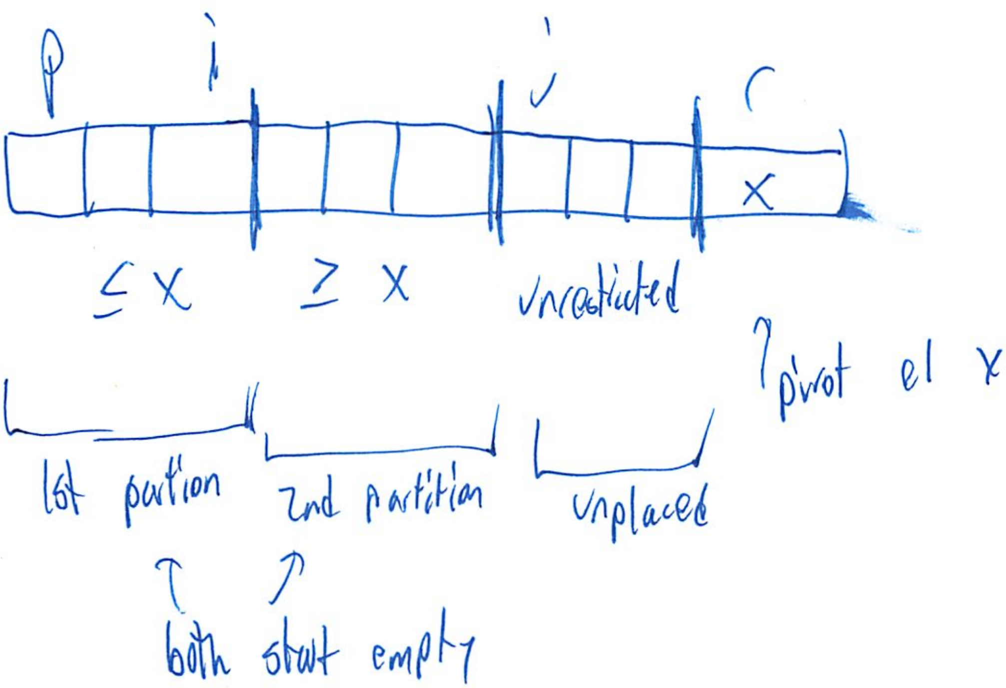
(13)

partioning selects an element $x = A[r]$ as a pivot

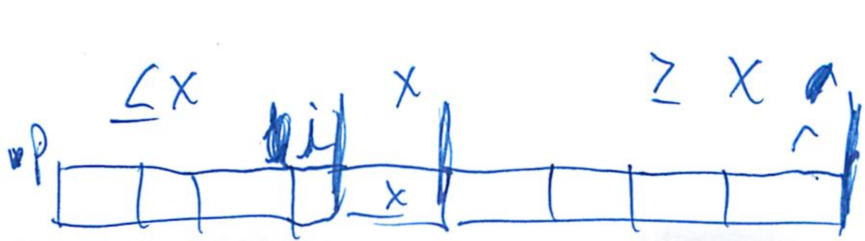
partions into 4 (empty) regions

1. If $p \le k \le i$ then $A[k] \le x$
2. If $i+1 \le k \le j-1$ then $A[k] > x$
3. If $k = r$ then $A[k] = x$

did we ever learn this?



ends w/



(14)

~~Idea~~ behind the scenes

1. if $A[j] > x \rightarrow$ increment j

2. if $A[j] \leq x \rightarrow$ increment i

   swap $A[i]$ and $A[j]$

   increment ~~i~~ j

(I think I get it now!)

~~Worst case~~

Randomized

pick $A[r]$ at random

(15)

Linear time sorting

Others were not linear
Since decion tree

Counting support

The # in each bin

Radix support

From old machines
Least - sig digit 1st!

Bucket sort

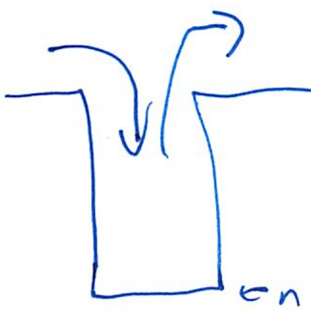
Sorts stuff into buckets
Pointers within buckets

(16)

dynamic sets = can add + subtract

Stacks

push + delete



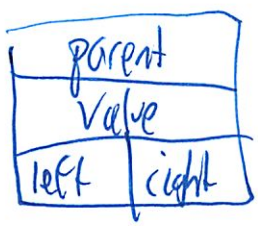
Underflow = pop empty stack

S.top > n = overflow

queue

linked list

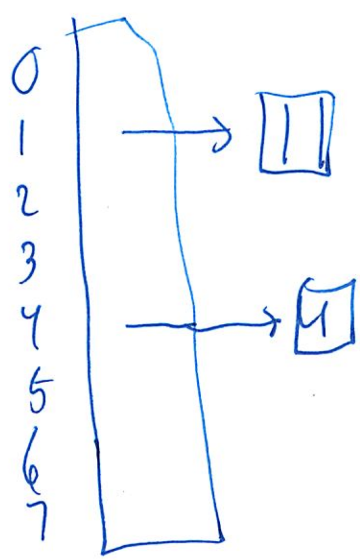
Binary trees



Hash tables

insert
search $= O(1)$
delete

most basic → direct address

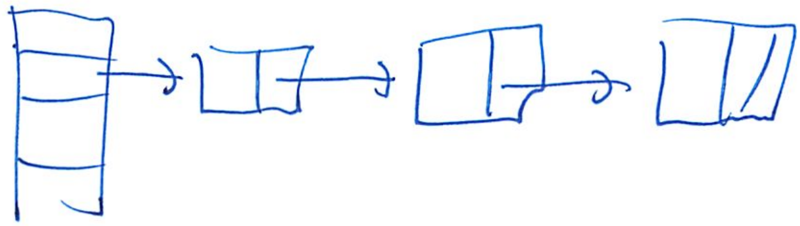


(18)

Or use a <u>hash function</u>

- division
- multiplication
- universal – choose fn randomly

Can <u>chain</u>



but this ruins the worst case

<u>Open addressing</u> keep going till find something

<u>Binary Search tree</u>

left: $y.key \leq x.key$

right: $y.key \geq x.key$

of x

(19)

In-Order tree walk ( )

(what I f-ed up last time at Google)

In-order tree walk (x.left)
print x.key
I.O T W (x.right)

Search(x, k)

search x for k

if = return

~~go left~~

if k < x.key
go left

else
go right

min () go all the way left

max() " " " " right

20

Insert

(this is the complicated one)
↳ no delete is

? should I know?

Takes node z for which $z.key = v$

$z.left = nil$

$z.right = nil$ / duh

Looks in tree
going left or right

? always will be a leaf I think
↳ did I ever realize that before?

Delete

This one is complex

if no children - just delete

if 1 child - replace it

if 2 children - find z's sucessor y
↳ in z's right subtree

have y takes z's place
rest of z's original right subtree is y's new right tree

21

z's left subtree becomes y's new left subtree

Height

random $h = \lg n$

worst case $h = n$

Red Black

; What did we do in class?

Balanced tree

Like when you add - be extra careful to keep it balanced

AVL = height balanced

left + right differ by at least 1

balance w/ rotations

Augmented structures

Can add extra info

must keep it up to date though

like <u>interval trees</u>

? study
6.006 did cover — but in a diff way

Can have both endpoints (closed)
or just 1 (half open)

good for time

Rod <u>cutting</u>

Another I didn't use book for

<u>Memoization</u>

Ah <u>Dynamic Programming</u> - Using previous results
- Smaller subproblems
- Combine answers to those

Can do bottom up - solve pieces + combine
top down - split up into little pieces + solve

23

Skipping other examples
- even though don't really know
- review it later

Greedy - ~~picking up a lot~~
- trying to take as much as possible up front
- Making ~~the~~ globally optimal sol from locally optimal options

knapsack project
(why didn't I use the book?)

Amortized Analysis

average the time it takes to perform the action over all operations performed

guarentees avg peformance in the worst case

In class this was making the array bigger/smaller

B-Trees

Balanced search trees designed to work on disks
Like red-black trees but minimize I/O

But can have many children
 L branching factors

but each node has $O(\lg n)$

$x.n$ keys → $x.n + 1$ children

Make a $(x.n + 1)$-way decision

height grows log
Optimized for disk
 how in depth — want save pg
 should I review?

25

disk-read()

disk-write()

node is ~1 pg large

branching factor ~ 50 - 2000

wider = less disk accesses

So each node has pointers to its children

But how does it know where to go?

Like in BST normally in order

Oh $k_1 \leq x.key_1 \leq k_2 \leq x.key_2 \ldots$

Height $h \leq \lg_t \frac{n+1}{2}$

Search

Start at root

Returns (y, i) such that $y.key_i = k$

Skipping rest

Merging Heaps

Fib Heaps

data structure ~~an~~ made of trees

find min $O(1)$

Others in constant amortized time

~~Un~~like binomial trees

which is like binary trees

↳ "our normal trees"

I think we talked about these - but not in detail...

Graphs

Sparse vs dense

representations

adj - list

adj - matrix

Some weighted / some not

BFS

I should really have this down!

It looks at all neighbors

Adds to the ~~start~~ end

Investigates the next one

This one officially does the colors thing

white = undiscovered

gray = queued

black = explored

28

Finds the shotest ~~past~~ path distance

BF Trees
Builds a predisessor sub graph

Depth First

Go deep
Then back track
Discovered + Finished time stamps
Insert to start
Forms parenthesee structure
Edge Types
Tree
Back
Forward
Cross - all other

Topo sort
DFS by finishing time
~~most~~ last finished first

Strongly Connected Components

Single Source shortest path

- when neg

~~All pairs~~

Bellman Ford

- Can be neg

$O(VE)$ relax edges

Dij

Faster

but must be all (+)

min queue w/ fib heap

DAG shortest path

All pairs

Floyd - Warshall can be neg

$O(V^3)$

finds all possible paths

Johnson

for sparse