Recitation 8

Overhead:

- Psets:
  - look @ solns
  - stats **A** mean: median:
    B mean: median:
  - handout @ end
- QUIZ:
  - 10/15 7:30-9:30 PM
  - conflict 10/16 8-10AM
    - email staff w/ reason
    - if previously emailed - confirm.
- Feedback Form

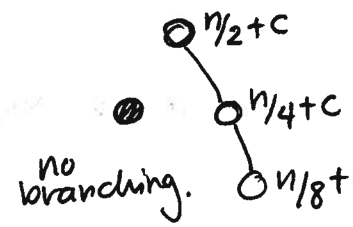
Agenda
- overhead
- exercises
- coding
- (?) shell sort
- psets

= Questions from lecture / anything you want reviewed.

= Pset

- nlogn bound on $T(n) = T(n/2) + n/2 + c$ is not tight.
  - yes you can prove $O(n\lg n)$
  - missing proof of $\Omega(n\lg n)$ ← impossible.

  real runtime $\Theta(n)$
  - why? b/c $n/2$ does not add n per step!



- OOPS - my apologies.

  $\log(f(n)) = \Theta(\lg(g(n))) \Rightarrow f(n) = \Theta(g(n))$ is FALSE.

  $\log(2^n)$ vs. $\log(3^n)$

  $n\log(2)$ $\quad$ $n\log(3)$

  yes Θ ... BUT you need to convert back. you can't cancel const values b/c they are actually e^c.

Shell sort

idea: some sorts are very fast for <u>nearly</u> sorted data.
we can nearly sort data really fast then pass off the data

sort every nth element, n decreases each time.
- lets elements make bigger "jumps" toward correct position

visually: use columns

1	2	3	4
5	6	...	

then sort the columns.

gap sequence matters!

1 4 10 23 57 132 301 ...

algorithm on wikipedia

worst case $\Theta(n^2)$ w/ gap seq 1 2 4 8 ...
best $\Theta(n \log^2 n)$

works well on real data.

Selection Sorts

idea: select an element, remove it, repeat.
if you select in a logical manner (max/min) you can sort!

variables/options: storage data structure
selection criteria.

struct	array	array	balanced BST	heap
selection	min	max	min/max	min/max
time per select	$O(n)$	$O(n)$	$O(\lg n)$	$O(\lg n)$
total	$O(n^2)$	$O(n^2)$	$O(n\lg n)$	$O(n\lg n)$

EXERCISES

1. state a situation for which insertion sort would be a good choice of sort.

2. what is the worst possible sort you can think of? each operation must be productive (i.e. no while (1){})

3. analyze the runtimes of insertion + merge sort on lists that are:
   a. sorted

   b. reverse-sorted

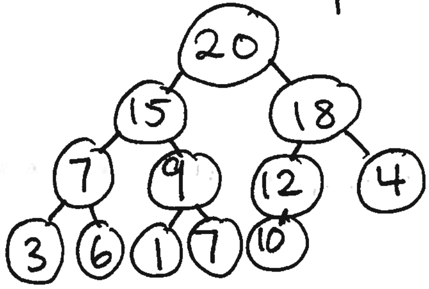
4. how does insertion sort's runtime change if you use a linked list?

5. what is the runtime of the following code? $\Theta(\quad)$
   then change the code to improve the runtime.
   you may assume all elements are ≥ 0.

```
def mergesort (A):
    if len(A) == 1 : return A
    m= len(A)/2
    B= len(A) * [-1]
    C= len(A) * [-1]
    B[: mid] = mergesort(A[: mid])
    C[: len(A) - mid] = merge sort(A[mid+1 :])
    return merge (A B C)
```

```
def merge (A B C):
    a=0
    b=0
    c=0
    while b< len(B) and
          c< len(C):
        if B[b]< C[c]:
            A[a]=B[b]
            b+=1
        else: A[a]=C[c]
            c+=1
        a+=1
```

6. convert the heap into its array representation



=

7.

		x	y					
19	17	13	5	3	11	7	2	1

parent of x :

left (x) :

right (x) :

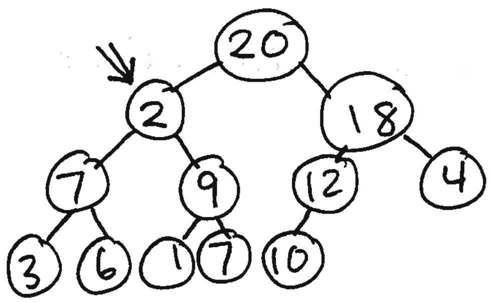
depth of heap :

parent(y) :

left (y) :

right (y) :

8. restore the max-heap property



9. get code heaps.py from website (or take a handout)
   a. implement left(i), right(i) and parent(i)
   b. complete the missing code in heapify
      optional: what code would go here for a min-heap?
   c. modify heapify to take a length argument.
   d. implement delete-max
   e. implement heap sort.
   f. challenge: implement build which builds a heap in place from an arbitrary array.

if you change the method signatures — make sure to alter the tests accordingly!