Lecture 12: Monday October 1, 2001

today

- Ch. 6: heapsort
  - analysis
    - time: $\Theta(n \lg n)$
    - space: in place (input plus $Theta(1)$)
    - correctness: loop invariants
- midterm format

announcements

- assignment 3
  - due now (late: until 3pm)
  - solutions posted today after 3pm
- midterm Wednesday, in class
- no assignment this week

heapsort run time

```
proc. Heapsort(A)       invariant: start of 3, A[1..j] is heap
1  Build-Max-Heap(A)
2  for j <- length[A] downto 2
3    do exchange A[1] <-> A[j]
4       heapsize[A] <- heapsize[A]-1
5       Max-Heapify(A,1)
```

heapsize versus number of levels

- for heap with $\updownarrow$ levels and s keys

$$2^{\updownarrow-1} \;\leq\; s \;\; 2^{\updownarrow} - 1$$

$$\updownarrow \;=\; \lceil \lg(s+1) \rceil$$

Max-Heapify run time

- worst case
  - trickledown: 2 KC, 1 swap for each level (except bottom)
  - Max-Heapify of s keys $\in \Theta(s)$
  - time for MH(n-1) + MH(n-2) + . . . + MH(2) is in

$$\sum_{s=2}^{n-1} \Theta(\lceil \lg(s+1) \rceil) \;=\; \Theta(\sum \lg s) \;=\; \Theta(n \lg n)$$

- time for BuildHeap?

Build-Heap (bottom-up; Floyd's alg'm)

- level -1 nodes (bottom) are heaps (size 1) so
- level -2 nodes are almost-heap roots: Max-Heapify now
- level -3 nodes are almost-heap roots: Max-Heapify …

```
heapsize[A] <- length[A]
for j <- length[A]/2 downto 1
  do Max-Heapify(A,j)
```

- assume $n = 2^k - 1$ (so heap with n keys has levels $0, 1, \ldots, k-1$)
- then Build-Heap:
  - at level t: Max-Heapify 2^t heaps with height $k - t$
  - worst case total number of KC =

$$2\sum_{t=0}^{k-1} 2^t(k-(t+1) \;=\; 2(2^k - k - 1) \;\approx\; 2n - 2\lg n \;\in\; \Theta(n)$$

Other-Build-Heap (top-down; Williams' alg'm)

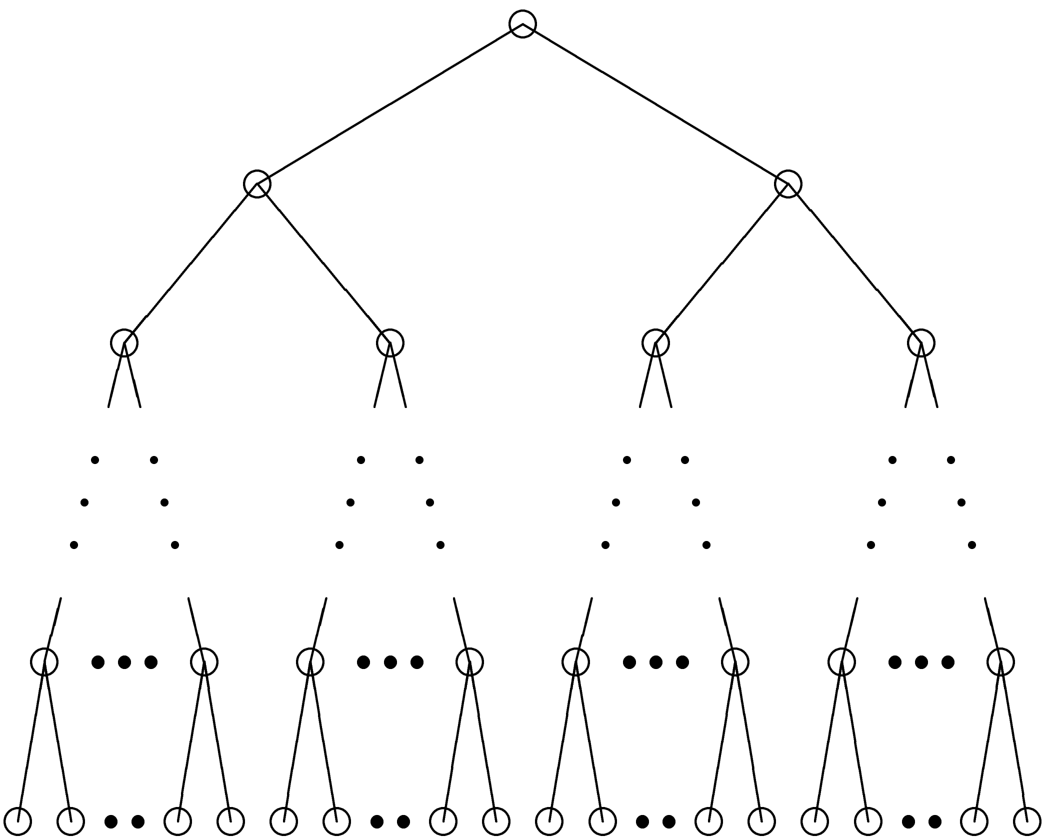
- A[1] is heap (size 1) so
- 'bubble-up' A[2], now A[1..2] heap so
- 'bubble-up' A[3], now A[1..3] heap ...

```
Other-Build-Heap(A)
for j <- 2 to length[A]
  do Bubble-Up-Insert(A,j)
```

- assume $n = 2^k - 1$ (so heap with n keys has levels $0, 1, \ldots, k-1$)
- then Other-Build-Heap:
  - at level t: 2^t bubble-ups from depth t
  - worst case total number of KC =

$$\sum_{t=0}^{k-1} 2^t k \ = \ 2^k(k-2) + 2 \ \approx \ n \lg n - 2n \ \in \ \Theta(n \lg n)$$

heap construction
worst case key comparisons, $n = 2^k - 1$

Williams		Floyd ($\times$ 2)
$0\ 2^0$		$(k-1)2^0$
$1\ 2^1$		$(k-2)2^1$
$2\ 2^2$		$(k-3)2^2$
$\vdots$		$\vdots$
$(k-2)2^{k-2}$		$1\ 2^{k-2}$
$(k-1)2^{k-1}$		$0\ 2^{k-1}$

$2^k(k-2)+2$	$2^k - k - 1$
$\approx n \lg n - 2n$	$\approx n - \lg n$
$\in\ \Theta(n \lg n)$	$\in\ \Theta(n)$

heapsort run time: conclusion

- worst case: $\in \Theta(n) + \Theta(n \lg n) = \Theta(n \lg n)$
- best case: can show also in $\Theta(n \lg n)$
- space: in place, namely space for input plus $O(1)$

heapsort correctness

- use loop invariants (see text)