Lecture 11: Wednesday January 29, 2003

today

- mental warm-up exercise
- Ch. 6: heapsort
  - proc Max-Heapify
  - proc BuildHeap
  - proc Heapsort
  - time complexity

mental warm-up from scratch, solve $f(n) = af(n/5) + n^2$, assuming $n = 5^k$, $f(1) = c_1 > 0$, and $a =$ (i) 24 (ii) 25 (iii) 26

$$
\begin{aligned}
(i): \quad f(n) &= af(\frac{n}{5}) + n^2 \\
&= a\left(af(\frac{n/5}{5}) + (n/5)^2 \right) + n^2 \\
&= a^2 f(\frac{n}{5^2}) + n^2\,(\frac{a}{5^2} + 1) \\
&= a^2\left(af(\frac{n/5^2}{5}) + \left(n/5^2\right)^2 \right) + n^2\,\left(\frac{a}{5^2} + 1\right) \\
&= a^3\left(f(\frac{n}{5^3}) \right) + n^2\,\left((\frac{a}{5^2})^2 + (\frac{a}{5^2})^1 + 1 \right) \\
&= \ldots \\
&= a^k f(\frac{n}{5^k}) + n^2 \sum_{j=0}^{k-1} (\frac{a}{5^2})^j \\
&= a^{\log_5 n} f(1) + n^2\,\frac{1 - (a/5^2)^k}{1 - (a/5^2)} \quad \in \Theta(n^2)
\end{aligned}
$$

$$
\begin{aligned}
(ii): \quad f(n) &= a^k f(\frac{n}{5^k}) + n^2 \sum_{j=0}^{k-1} (\frac{a}{5^2})^j \\
&= a^{\log_5 n} f(1) + n^2\,(k) \quad \in \Theta(n^2 \log n)
\end{aligned}
$$

$$
\begin{aligned}
(iii): \quad f(n) &= a^k f(\frac{n}{5^k}) + n^2 \sum_{j=0}^{k-1} (\frac{a}{5^2})^j \\
&= a^{\log_5 n} f(1) + n^2\,\frac{\left[(a/5^2)^{\log_5 n} = n^{\log_5 (a/5^2)}\right] - 1}{(a/5^2) - 1} \\
&= n^{\log_5 a}\left(f(1) + \frac{1}{(a/5^2) - 1} \right) - \frac{n^2}{(a/5^2) - 1} \quad \in \Theta(n^{\log_5 a})
\end{aligned}
$$

Max-Heapify: from almost-heap to heap

- recall heap: array (implicit binary tree), and
  keys satisfy (max-)heap property: parent $\geq$ child
- almost heap: heap, except root may not satisfy h.p.

```
Max-Heapify(A,j)
(* turn almost-heap (root j) into heap via 'trickledown'
(* see previous lecture or text for more detail

node <- A[j]
while node < larger child
  do  interchange
```

heapsort

```
length[A]:   number keys in A                      (never changes)
heapsize[A]: number keys in heap rooted at A[1]    (decreases)

proc. Heapsort(A)      * at start of line 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)
```

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

- each depth d node (bottom) is heap (size 1) root — so
- each depth $d-1$ node is almost-heap root: Max-Heapify — now
- each depth $d-2$ node are almost-heap root: Max-Heapify — now
- ...
- each depth 1 node is almost-heap root: Max-Heapify — now
- the depth 0 node is almost-heap root: Max-Heapify — now
- the depth 0 node is heap root 

```
1 heapsize[A] <- length[A]        * at start of line 2,
2 for j <- length[A]/2 downto 1   * each subtree rooted at
3   do Max-Heapify(A,j)           * position >j is heap
```

coming soon: analysis

- correctness: loop invariants
- space: in place
- running time
  - Max-Heapify $\in O(\lg n)$ why?
  - Build-Heap $\in \Theta(n)$ why?
  - Heapsort (BC,WC) $\in \Theta(n \lg n)$ why?
- improvements
  - Max-Heapify
    - trickledown empty location
    - insert key into (bottom) location
    - bubble-up