Lecture 14: Wednesday October 10, 2001

today

- quicksort [CLRS Ch. 7]
  - the algorithm
  - the analysis: WC, BC, AC
  - the improvements
  - in practice?

announcements

- assignment 4: due Wednesday October 17

quicksort

```
QS(A,lo,hi)                    (*sort A[lo ... hi]
if lo < hi then
  spI <- partition(A,lo,hi)  (*split index
  QS(A, lo ,spI-1)
  QS(A,spI+1, hi )
```

```
partition(A,lo,hi)          (*upon completion,
         A[lo ... lastSmallI]    smaller  keys
         A[1+lastSmallI]           splitter
         A[2+lastSmallI ... hi]  larger   keys
spK <- A[hi]
lastSmallI <- lo-1
for next <- lo to hi-1 do
  if A[next] <= spK then
    lastSmallI <- lastSmallI + 1
    exchange A[lastSmallI] <-> A[next]
exchange A[1+lastSmallI] <-> A[hi]
return 1+lastSmallI
```

```
 * * * * * * * * *                ? ? ? ? ? ?  ^
 lo     lastSmallI                next         hi
--------------------------------------------------
|    <= spK       |     > spK     |           | spK
--------------------------------------------------
```

```
*   ?   ?   ?   ?   ?   ?   ?   ^
    2   8   7   1   3   5   6   4

    *   ?   ?   ?   ?   ?   ?   ^
    2   8   7   1   3   5   6   4

    *       ?   ?   ?   ?   ?   ^
    2   8   7   1   3   5   6   4

    *           ?   ?   ?   ?   ^
    2   8   7   1   3   5   6   4

    *   *           ?   ?   ?   ^
    2   1   7   8   3   5   6   4

    *   *   *           ?   ?   ^
    2   1   3   8   7   5   6   4

    *   *   *               ?   ^
    2   1   3   8   7   5   6   4

    *   *   *                   ^
    2   1   3   8   7   5   6   4

    *   *   *   ^
    2   1   3   4   7   5   6   8
```

quicksort correctness

- follows from partition correctness
- partition loop invariant: at start of loop at 3,

```
 * * * * * * * * *                  ? ? ? ? ?  ^
 lo     lastSmallI                  next       hi
--------------------------------------------------
|    <= spK       |    > spK      |          | spK
--------------------------------------------------
```

  - `A[lo ... lastSmall]` $\leq$ `spK`
  - `A[1+lastSmall ... next-1]` $>$ `spK`
  - `A[hi]` $=$ `spK`

- proof: see text

why we study quicksort

- very efficient, in use
- divide and conquer, randomization
- huge literature; a model for analysis of algorithms

history

- Hoare 1961: conception
- Knuth 1973: first analysis
- Sedgewick 1980: more analysis
- McDiarmid & Hayward: strong concentration ☺

QS: a recursion tree

- QS: run time proportional to number key comparisons
- counting KC: only need know (at each call) rank of split key

```
               06
         04             09
   02       05    07                         16
01    03             08          13
                           10          14
                                 12       15
                              11
```

- observe: in resulting tree, for each node,
  all keys(left subtree) $\leq$ node's key $\leq$ all keys(right subtree)
- 1-1 correspondence: quicksort recursion tree $\leftrightarrow$ binary search tree

QS run time: worst case

`a bad case`

```
01

   02

      03

         04

            05

               06

                  ...
```

- number of KC in above case?
- $W(1) = 0$
- $W(n) = n - 1 + W(0) + W(n-1)$ why?
- $W(n) \;=\; \sum_{j=n-1}^{1} j \;=\; n(n-1)/2$
- can show $W(n)$ is worst case number of KC, so
- QS worst case run time in $\Theta(n^2)$

QS run time: best case

`a good case`

```
                                    08

                    04                                      12

          02                  06                  10                  14

01        03        05        07        09        11        13        15
```

- number of KC in above case?
- $B(1) = 0$
- $B(n) = n - 1 + B(\lfloor n - 1/2 \lfloor) + B(\lceil n - 1/2 \rceil)$ why?
- $B(n) \approx n + 2B(n/2)$
- can show $B(n) \in \Theta(n \lg n)$
- can show $B(n)$ is best case number of KC, so
- QS best case run time in $\Theta(n \lg n)$