Lecture 16: Monday February 10, 2003

today

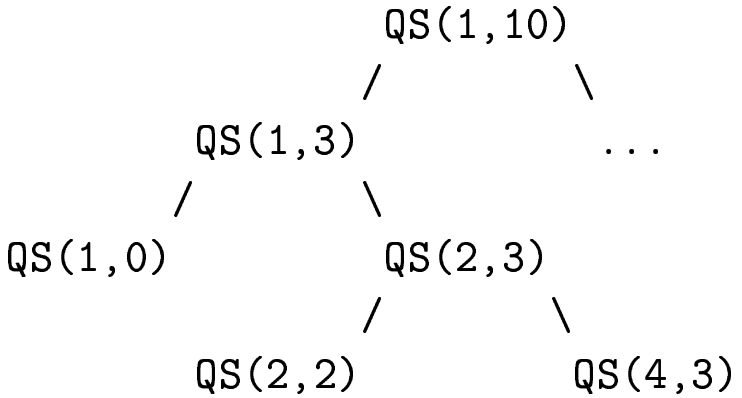
- quicksort (finish)
  - improvement: stack size, tail recursion
  - just for fun: strong concentration
- towards a sorting lower bound [CLRS Ch. 8.1]
  - decision trees

QS space requirements

- not an in place sorting algorithm, because extra space required for all subproblems on the stack
- worst case: can have $\Theta(n)$ subproblems on stack

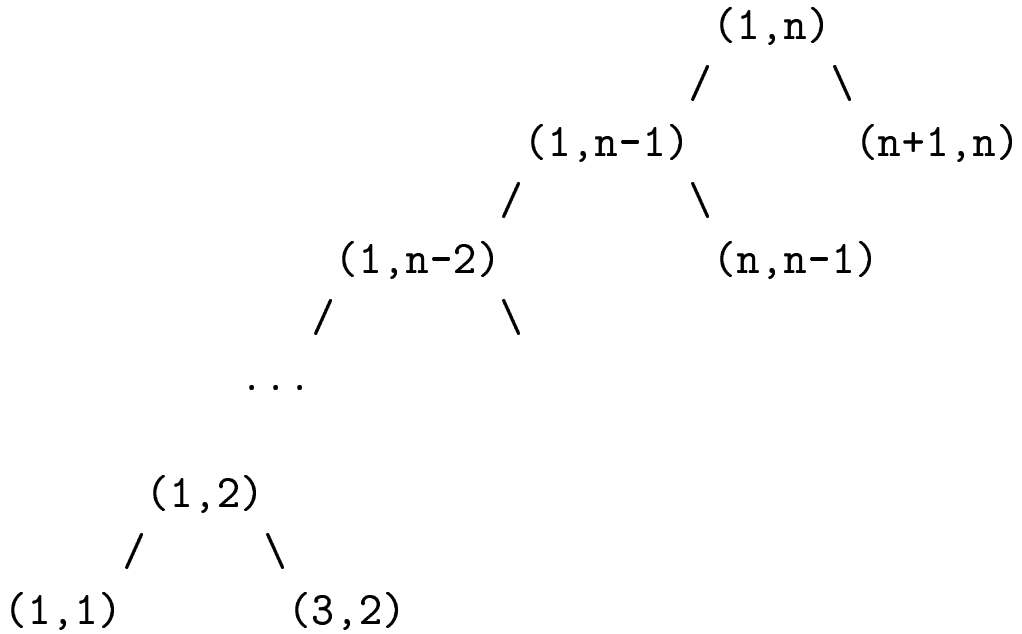
analysis of recursion

- for careful analysis, need to understand implementation
- RAM model implementation?
  - subproblem $\leftrightarrow$ procedure call
  - activation record (AR): info needed to allow interrupted proc. call execution, e.g. code addr, parameter values, resumption addr
  - ARs stored on run-time stack (RTS)
  - simplest impl'n: assume currently active AR stored top of stack
- at procedure call time
  - update caller AR: record resumption instruction
  - initialize callee AR
  - push callee AR, and start callee execution
- at procedure completion time
  - record any values to be returned
  - pop callee AR, and resume caller execution
- trace RTS state for QS recursion tree below



RTS time/space issues

- time
  - need to account for AR pushes (copy time)
  - Θ analysis: usually not significant
  - exact analysis (e.g. real time computing): usually significant; consider avoiding/removing recursion
  - removing tail recursion: a recursive call which is last op'n of proc can be replaced with loop
- space: how big can RTS get?
  - all ancestor calls of active proc on RTS
  - so RTS of any recursion tree with depth n will have n problems on RTS at some point
  - e.g. `QS(1,n)` recursion tree has WC depth n



minimizing quicksort RTS space

- improvement 1: remove tail recursion

```
proc QS(lo,hi)                    proc QSTR(lo,hi)
  if lo<hi do                       while lo<hi do
    partition(lo,split,hi)            partition(lo,split,hi)
    QS(lo,split-1)                    QSTR(lo,split-1)
    QS(split+1,hi)                    lo <- split+1
```

- QSTR still uses WC $\Theta(n)$ extra space 
- improvement 2: make recursive call on smaller subproblem
- now each subproblem in recursion tree $\leq 1/2$ size of parent

  $\Rightarrow$ recursion tree depth $\leq \lg n$

  $\Rightarrow$ QSTRO(n) uses WC $\Theta(\lg n)$ extra space 

```
proc QSTRO(lo,hi)       (*tail removal optimization*)
  while lo<hi do
    partition(lo,split,hi)
    if (split-lo) <= (hi-split)
      QSTRO(lo,split-1)
      lo <- split+1
    else
      QSTRO(split+1,hi)
      hi <- split-1
```

quicksort: strong concentration

Theorem. For any $\varepsilon > 0$, as $n \to \infty$,

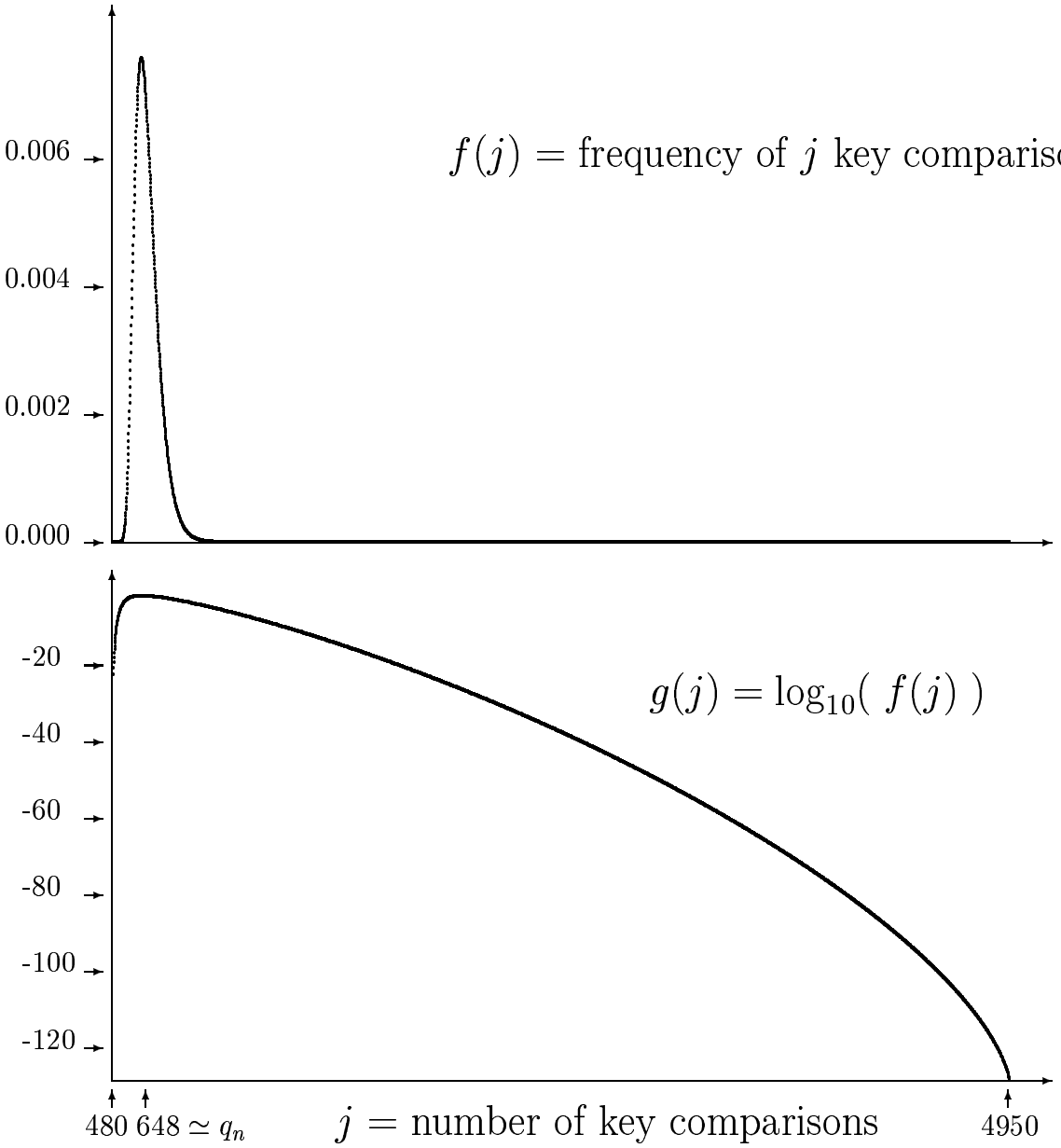
$$\Pr\left[\left|\frac{Q_n}{q_n} - 1\right| > \varepsilon\right] \in n^{-2\varepsilon(\ln\ln n + O(\ln\ln\ln n))}.$$

Proof. for homework just kidding 

computing QS KC distribution exactly

- $q_{n,k}$: probabililty an equiprobable n-key QS does exactly k KC
- can compute $q_{n,k}$ exactly (iteration) 
- $q_{100,k}$ shown on next slide 

QS KC distribution $q_{100,k}$



QS space requirements

- not an in place sorting algorithm, because extra space required for all subproblems on the stack
- worst case: can have $\Theta(n)$ subproblems on stack

QS improvements

- small sublists: use insertion sort
  - can determine best crossover size (about 20)
- split selection
  - sampling
    - use A[mid] instead of A[hi] mid = (lo+hi)/2
    - better: median of { A[lo], A[mid], A[hi] }
  - randomization
    - select split using pseudo-random number generator
- space! stack size
  - remove tail recursion
  - make sure smaller subproblem executed first
  - only need $\Theta(\log n)$ extra space