Lecture 17: Wednesday October 17, 2001

today

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

announcements

- assignment 5: due Monday October 29

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

```
                QS(1,10)
               /        \
        QS(1,3)           ...
       /       \
QS(1,0)         QS(2,3)
               /       \
        QS(2,2)         QS(4,3)
```

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

```
                            (1,n)
                           /     \
                    (1,n-1)       (n+1,n)
                   /       \
             (1,n-2)        (n,n-1)
            /       \
        ...

      (1,2)
     /     \
(1,1)       (3,2)
```

minimizing quicksort RTS space

- improvement 1: remove tail recursion

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

```
proc QSTR(lo,hi)
  while lo<hi do
    partition(lo,split,hi)
    QSTR(lo,split-1)
    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 ☺

a sorting lower bound [CLRS Ch. 8.1]

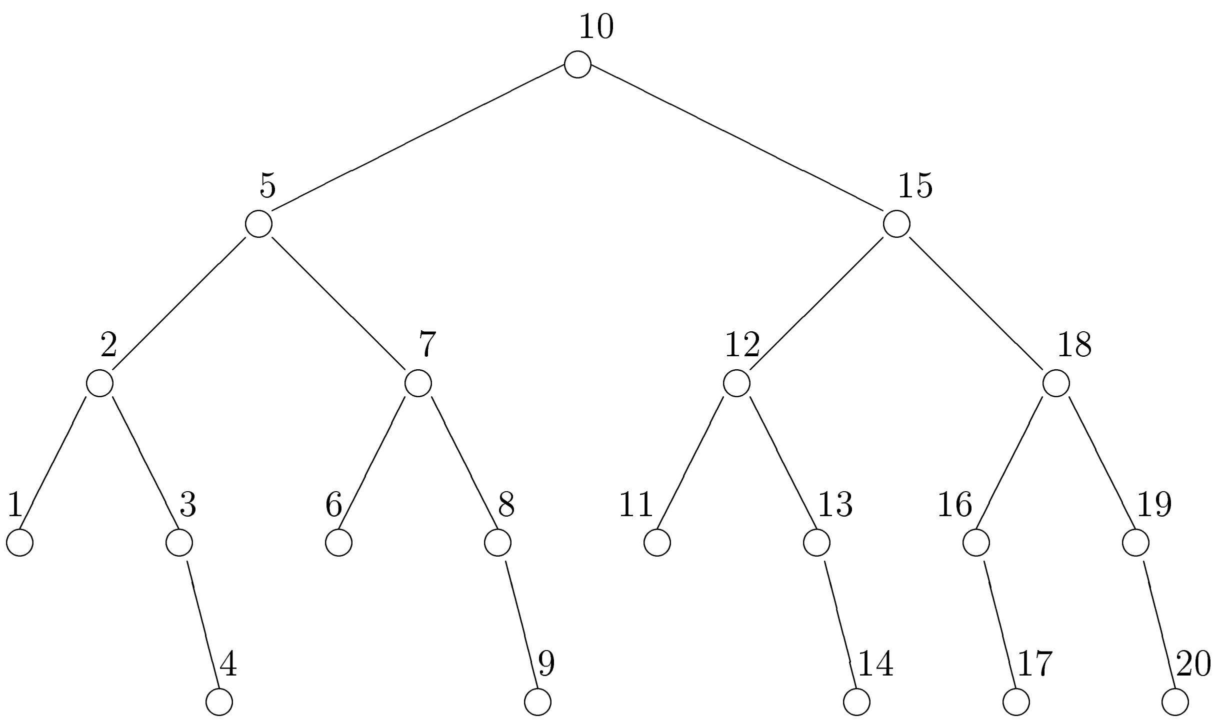
two useful trees in algorithm analysis

- **recursion tree**
  - node $\leftrightarrow$ recursion call
  - describes algorithm execution for one particular input, by showing all calls made
  - one algorithm execution $\leftrightarrow$ all nodes
  - useful in analysis: sum number of operations over all nodes

- **decision tree**
  - node $\leftrightarrow$ algorithm decision
  - describes algorithm execution for all possible inputs, by showing all possible algorithm decisions
  - one algorithm execution $\leftrightarrow$ one root-to-leaf path
  - useful in analysis: sum number of operations over one path

binary search decision tree, $n = 20$

- assume input keys in array $A[1 \ldots 20]$
- tree node: $\leftrightarrow$ '3-way' key comparision $<?$ $=?$ $>?$
- node label: $A[j]$

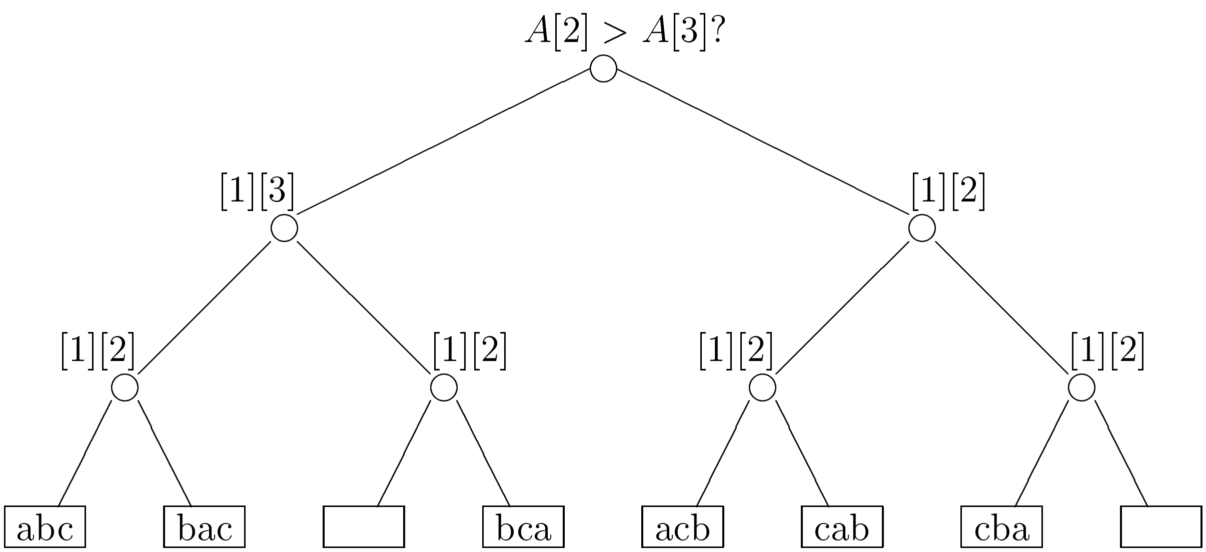


- WC number KC: 5 — in general, $1 + \lfloor \lg n \rfloor$
- AC number KC? — ask: input distribution?
  - target in list, each location equiprobable
    $(2^0 \cdot 1 \ + \ 2^1 \cdot 2 \ + \ 2^2 \cdot 3 \ + \ 2^3 \cdot 4 \ + \ 5 \cdot 5)/20 \ = \ 3.7$
  - target not in list, each 'gap' equiprobable
    $(11 \cdot 4 + 10 \cdot 5)/21 \approx 4.5$
  - both dist'ns: AC$(n = 2^k - 1) \approx \lfloor \lg n \rfloor + 1/2$

selection sort decision tree, $n = 3$

- assume input keys in array $A[1 \ldots 3]$, initially `[a b c]`
- tree node: `A[k]>A[j]?` key comparision
- node label: $A[j]$

```
for j <- n downto 2 do
  psn <- j    (*index of max*)
  for k <- j-1 downto 1 do
    if A[k] > A[psn] then psn <- k
  exchange A[j] <-> [psn]
```



- notice: every case, 4 KC