Lecture 17: Wednesday February 12, 2003

today

- decision tree sorting lower bound [CLRS Ch. 8.1]

soon

- dynamic programming (start today) [CLRS Ch. 15.1-3]
- union find [CLRS Ch. 21]
- graph algorithms [CLRS Ch. 22,23]

a sorting lower bound [CLRS Ch. 8.1]

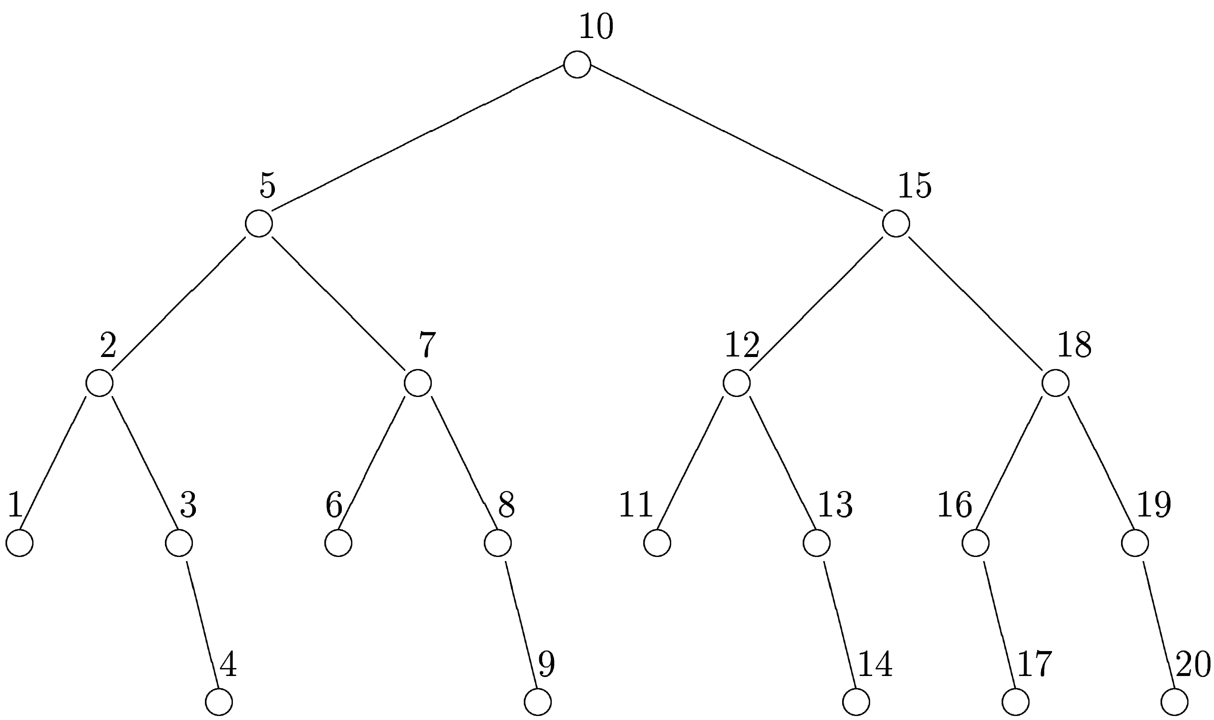
two useful trees in algorithm analysis

- **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: $A[j]$

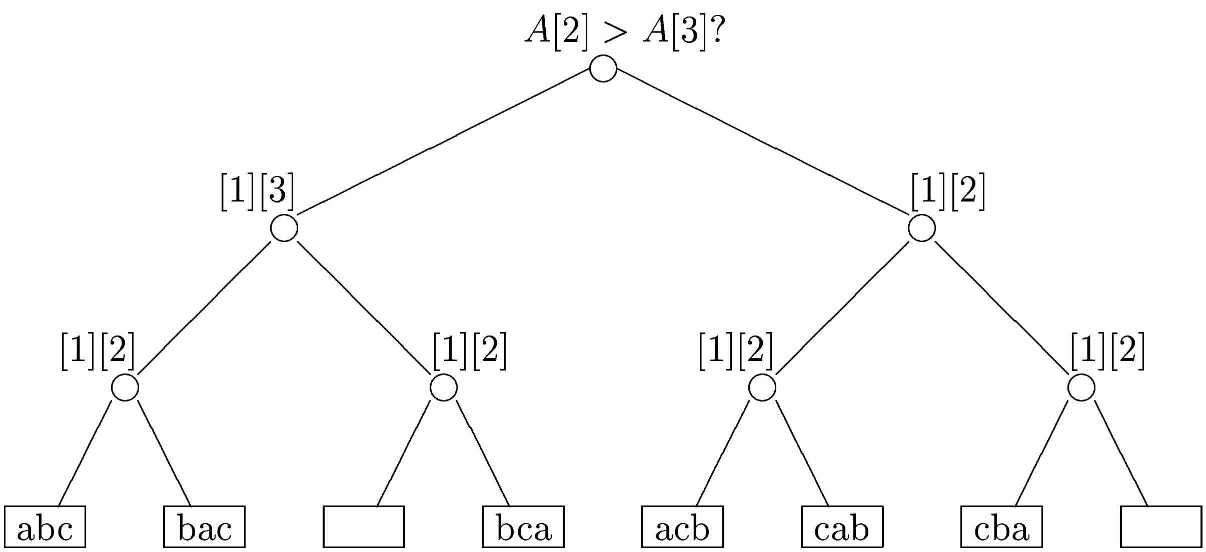


- WC number KC: 5 — in general, $1 + \lfloor \lg n \rfloor$
- AC number KC? — ask: input distribution?
  - target in list, each location equiprobable
    $(2^0 \cdot 1 \ + \ 2^1 \cdot 2 \ + \ 2^2 \cdot 3 \ + \ 2^3 \cdot 4 \ + \ 5 \cdot 5)/20 \ = \ 3.7$
  - target not in list, each 'gap' equiprobable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

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, 3 KC

sorting lower bound

- comparison based sort: keys can be compared **only**
- this argument considers **only** comparison based sort algorithms
- any binary tree with t leaves and k levels:
  - $t \leq 2^{k-1}$
  - $\lg t \leq k-1$
  - $1 + \lg t \leq k$, namely
    binary tree with t leaves has $\geq 1 + \lg t$ levels
- the decision tree of any comparison based sort ...
  - binary
  - has $n!$ leaves
  - ...so has at least $1 + \lg(n!) = 1 + \sum_{j=1}^{n} \lg j \in \Theta(n \lg n)$ levels
- conclusion: any comparison based sort has
  - worst case number of key comparisons in $\Omega(n \lg n)$
  - worst case run time in $\Omega(n \lg n)$
- comparison based: selectsort, insertsort, heapsort, quicksort
- **not** comparison based: radix sort, bucket sort
- see Ch. 8 for extra reading

end of material for Section B1 Midterm

dynamic programming

- an algorithm design technique
- DP: avoiding recomputation of repeated subproblems by storing subproblem answers in tables/arrays

1st example problem: Fibonacci numbers

- $f(n) \; = \; \begin{cases} n & \text{if } \; n = 0, 1 \\ f(n-1) + f(n-2) & \text{if } \; n \geq 2 \end{cases}$

- 1st Fibonacci implementation: recursion

```
proc f(n)
  if n<2 then return n
  else return f(n-1)+f(n-2) endif
endf
```

```
                 -- f5 --
                /        \
             f4            f3
            /  \          /  \
          f3    f2      f2   f1
         / \   / \     / \
       f2  f1 f1  f0  f1  f0
      / \
    f1  f0
```

- ☹ repeated function calls
- time $$T(n) = \begin{cases} c_1 & \text{if } n = 0, 1 \\ c_2 + T(n-1) + T(n-2) & \text{if } n \geq 2 \end{cases}$$
- $T(n) > f(n)$ so $T(n) \in \Omega(\left(\frac{1+\sqrt{5}}{2}\right)^n)$ ☹