# Lecture 13: Quick Sort

Agenda:

- Quicksort
  - Algorithm recall
  - Correctness
  - WC running time (KC)
  - BC running time (KC)

Reading:

- Textbook pages 149 – 153

## Another sorting meets divide-and-conquer (recall):

- The ideas:
  - Pick one key (so far, the last key)
  - Compare to others: partition into smaller and greater sublists
  - Recursively sort two sublists
- Pseudocode:

```
procedure Quicksort(A, p, r)      **p 146

   if p < r then
      q ← Partition(A, p, r)
      Quicksort(A, p, q − 1)
      Quicksort(A, q + 1, r)


procedure Partition(A, p, r)      **p 146
   ** A[r] is the key picked to do the partition

   x ← A[r]
   i ← p − 1
   for j ← p to r − 1 do
      if A[j] ≤ x then
         i ← i + 1
         exchange A[i] ↔ A[j]
   exchange A[i + 1] ↔ A[r]
   return i + 1
```

- $A[1..9] = \{3, 6, 4, 7, 1, 2, 5, 9, 8\}$

## Quicksort correctness:

- It follows from the correctness of Partition.
- Partition correctness:
  - Loop invariant:
    At the start of `for` loop:
    1. $A[p..i] \leq A[r]$ — $A[s] \leq A[r], p \leq s \leq i$
    2. $A[(i+1)..(j-1)] > A[r]$
    3. $x = A[r]$
  - Proof of LI: (pages 147 – 148)
    1. Initialization
    2. Maintenance
    3. Termination
  - LI correctness implies Partition correctness

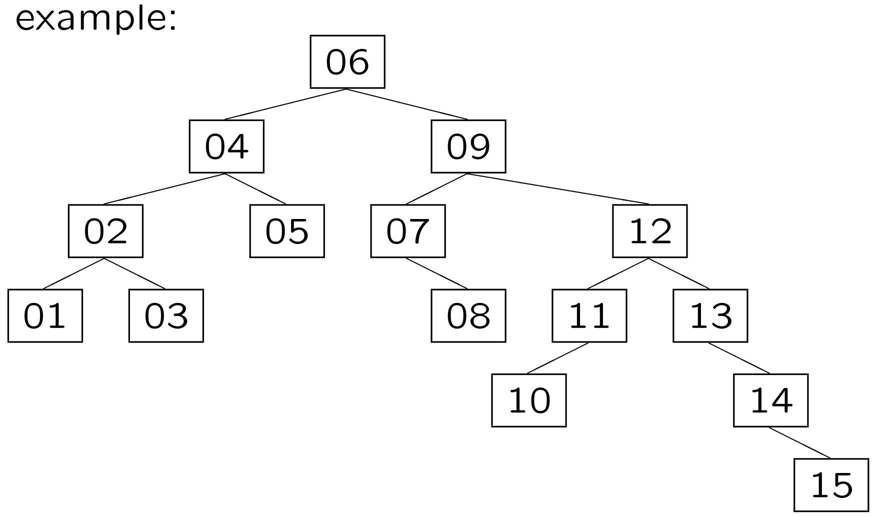
## Quicksort notes:

- Why we study it:
  - 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, etc.

# Quicksort recursion tree:

- Observations:
  - (Again) key comparison is the dominant operation
  - Counting KC
    — *only* need to know (at each call) the rank of the split key

- An example:




- More observations:
  - In the resulting recursion tree, at each node
    (all keys in left subtree) $\leq$ (key in this node) $<$ (all keys in right subtree)
  - 1-1 correspondence:
    quicksort recursion tree $\longleftrightarrow$ binary search tree

# Quicksort WC running time:

- The split key is compared with every other key: $(n-1)$ KC

- Recurrence:

  $$T(n) = T(n_1) + T(n-1-n_1) + (n-1),$$

  where $0 \leq n_1 \leq n-1$

  Base case: $T(0) = 0$, $T(1) = 0$

- Notice that when both subarrays are non-empty, we will be having

  $$(n_1 - 1) + (n - 1 - n_1 - 1) = (n-3)$$

  KC next level ...

- Worst case: one of the subarray is empty !!! needs $(n-2)$ KC next level

- WC recurrence:

  $$T(n) = T(0) + T(n-1) + (n-1) = T(n-1) + (n-1),$$

- Solving the recurrence — Master Theorem does NOT apply

  $$\begin{aligned} T(n) &= T(n-1) + (n-1) = T(n-2) + (n-2) + (n-1) \\ &= \ldots \\ &= T(1) + 1 + 2 + \ldots + (n-1) \\ &= \frac{(n-1)n}{2} \end{aligned}$$

  So, $T(n) \in \Theta(n^2)$

- Therefore, quicksort is *bad* in terms of WC running time !

## Quicksort BC running time:

- Notice that when both subarrays are non-empty, we will be saving 1 KC ...

- Best case: each partition is a bipartition !!!

  Saving as many KC as possible every level ...

  The recursion tree is as short as possible ...

- Recurrence:

$$T(n) = 2 \times T(\frac{n-1}{2}) + (n-1),$$

- Solving the recurrence — apply Master Theorem? not exactly

  $T(n) \in \Theta(n \log n)$

- Question:
  - What is the best case array? for $n = 7$?

- Conclusion:
  - In order to save time, $A[n]$ better BI-partitions the array ...

    — usually it might not bipartition ... we will push it by a technique called *randomization* (future lectures)

## Quicksort BC running time (cont'd):

- In the recursion tree, what is the number of KC at each level?

  Answer:

  - $n - 1$ at the top level
  - at most 2 nodes at the 2nd level, at least $(n_1 - 1) + (n - 1 - n_1 - 1) = n - 3$ KC
  - at most 4 nodes at the 3rd level, at least $(n_1 - 3) + (n - 1 - n_1 - 3) = n - 7$ KC
  - ...
  - at $k$th level, at most $2^{k-1}$ nodes, at least $n - 2^k + 1$ KC

- How many levels are there?

  Answer:

  - At least $\lg n$ levels — binary tree

- So, at least we need $\sum_{i=1}^{\lg n - 1}(n - 2^i + 1)$ KC, and

$$\sum_{i=1}^{\lg n - 1}(n - 2^i + 1) = (n + 1)(\lg n - 1) - (n - 2) \in \Theta(n \log n)$$

- Try $n = 2^k - 1$ to get the closed form for the following recurrence

$$T(n) = \begin{cases} 0, & \text{if } n = 1 \\ (n-1) + T(\lfloor \frac{n-1}{2} \rfloor) + T(\lceil \frac{n-1}{2} \rceil), & \text{if } n \geq 2 \end{cases}$$

## Have you understood the lecture contents?

| `well` | `ok` | `not-at-all` | `topic` |
|---|---|---|---|
| □ | □ | □ | quicksort idea |
| □ | □ | □ | quicksort pseudocode(s), execution |
| □ | □ | □ | correctness of quicksort |
| □ | □ | □ | quicksort WC running time |
| □ | □ | □ | worst case |
| □ | □ | □ | quicksort BC running time |
| □ | □ | □ | best case |