Selection

Chapter 9

The Median Problem

- Given n elements x_1, x_2, ..., x_n, taken from a total order, find the median of this set.
- Of course, we can sort the set in O(n log n) time and then index the (n/2)-th element.

7 4 9 6 2 → 2 4 6 7 9

- Can we solve this problem faster?
- It's easier if we generalize the problem!

The Selection Problem

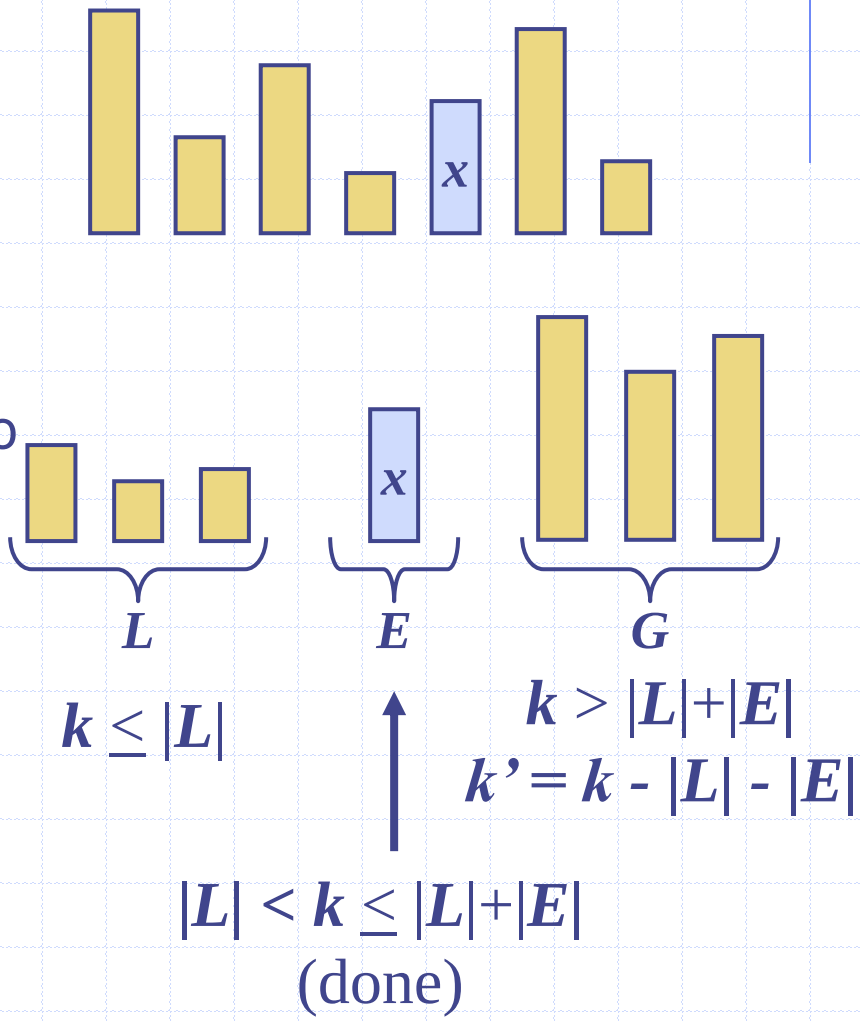
- Given an integer k and n elements x_1, x_2, …, x_n, taken from a total order, find the k-th smallest element in this set.
- Again, we can sort the set in O(n log n) time and then index the k-th element.

k=2 7 4 9 6 2 → 2 4 6 7 9

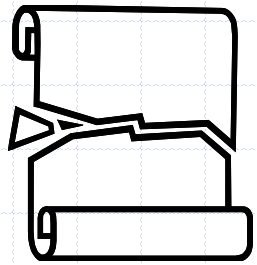
- Adding k to the problem gives us flexibility when doing recursion.

Quick-Select

- Quick-select is a randomized selection algorithm based on the prune-and-search paradigm:
  - Prune: pick a random element x (called pivot) and partition S into
    - L: elements less than x
    - E: elements equal x
    - G: elements greater than x
  - Search: depending on k, either answer is in E, or we need to recur in either L or G



Partition

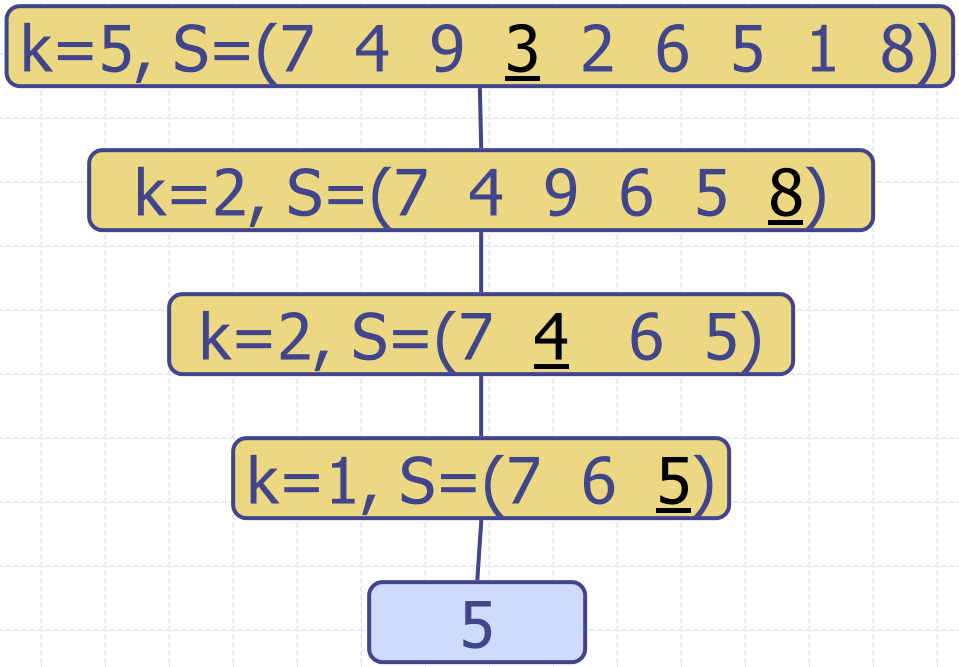


- We partition an input sequence as in the quick-sort algorithm:
  - We remove, in turn, each element y from S and
  - We insert y into L, E or G, depending on the result of the comparison with the pivot x
- Each insertion and removal is at the beginning or at the end of a sequence, and hence takes $O(1)$ time
- Thus, the partition step of quick-select takes $O(n)$ time

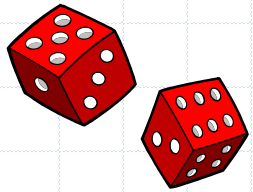
Algorithm *partition*(S, p)
  Input sequence S, position p of pivot
  Output subsequences L, E, G of the elements of S less than, equal to, or greater than the pivot, resp.
  $L, E, G \leftarrow$ empty sequences
  $x \leftarrow S.erase(p)$
  while $\neg S.empty()$
    $y \leftarrow S.eraseFront()$
    if $y < x$
      $L.insertBack(y)$
    else if $y = x$
      $E.insertBack(y)$
    else $\{ y > x \}$
      $G.insertBack(y)$
  return L, E, G

Quick-Select Visualization

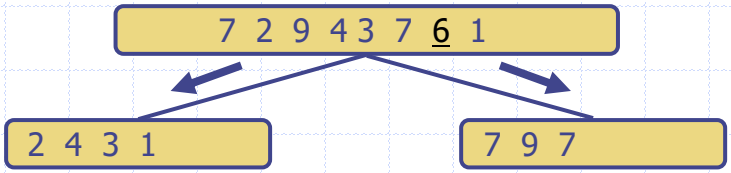
- An execution of quick-select can be visualized by a recursion path
  - Each node represents a recursive call of quick-select, and stores k and the remaining sequence



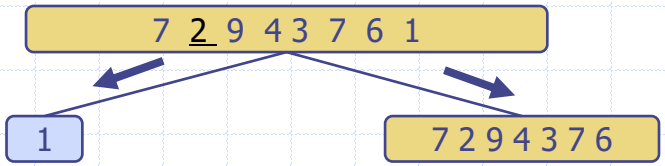
Expected Running Time



- Consider a recursive call of quick-select on a sequence of size s
  - Good call: the sizes of L and G are each less than $3s/4$
  - Bad call: one of L and G has size greater than $3s/4$

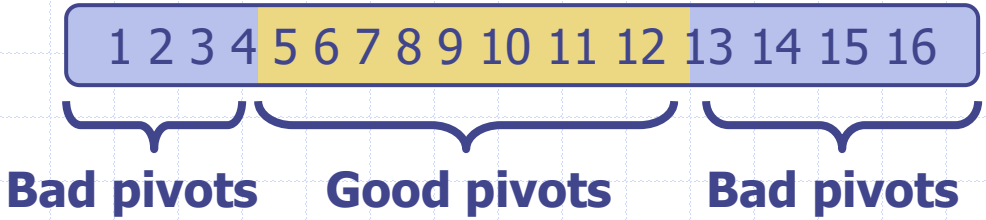


Good call

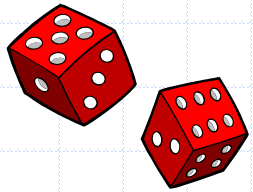


Bad call

- A call is good with probability $1/2$
  - 1/2 of the possible pivots cause good calls:



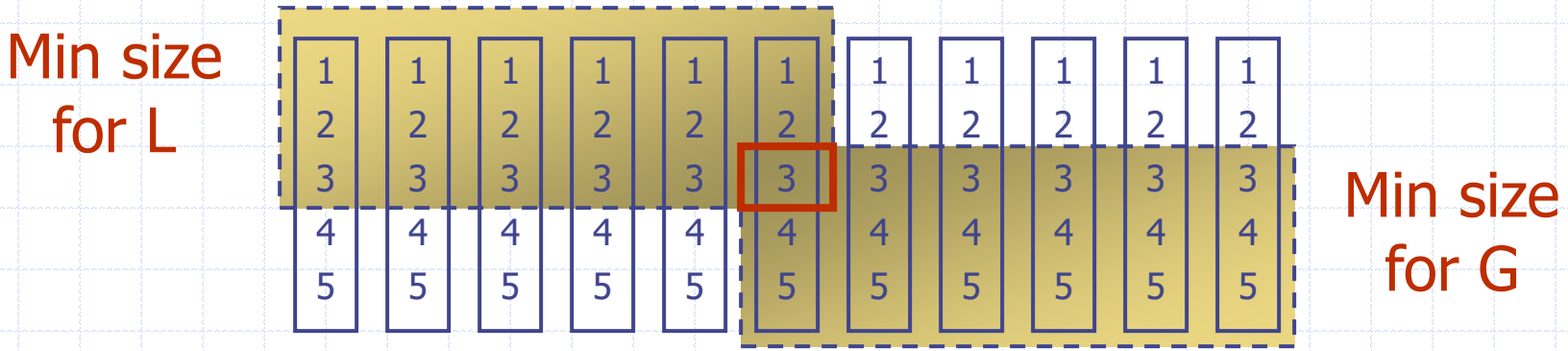
Expected Running Time, Part 2



- Probabilistic Fact #1: The expected number of coin tosses required in order to get one head is two
- Probabilistic Fact #2: Expectation is a linear function:
  - $E(X + Y) = E(X) + E(Y)$
  - $E(cX) = cE(X)$
- Let T(n) denote the expected running time of quick-select.
- By Fact #2,
  - $T(n) \leq T(3n/4) + bn$ * (expected # of calls before a good call)
- By Fact #1,
  - $T(n) \leq T(3n/4) + 2bn$
- That is, by plug-and-chug, T(n) is a geometric series:
  - $T(n) \leq 2bn + 2b(3/4)n + 2b(3/4)^2n + 2b(3/4)^3n + \ldots$
- So T(n) is O(n).
- QuickSelect solves the selection problem in O(n) expected time.

Deterministic Selection

- We can do selection in O(n) worst-case time.
- Main idea: recursively use the selection algorithm itself to find a good pivot for quick-select:
  - Divide S into n/5 sets of 5 each
  - Find a median in each set
  - Recursively find the median of the "baby" medians.



More slowly…

- We want to select the k-th element of a set S.
- If S is small (say, less than 40 elements), sort it and choose the kth element in the sorted order.

Otherwise:

- Divide S into $\left\lceil \frac{n}{5} \right\rceil$ sets of at most 5 elements each.

19	31	63	7	29	56	47	83	91	43	58
41	55	27	82	83	26	88	24	64	17	35
4	62	81	49	66	10	35	22	44	56	41
80	69	54	64	41	52	14	70	32	61	
37	5	55	35	38	42	59	92	6	75	

This can be done in linear time.

More slowly…

- Find the median of each set, by sorting each set.

4	5	27	7	29	10	14	22	6	17	35
19	31	54	35	38	26	35	24	32	43	41
37	55	55	49	41	42	47	70	44	56	58
41	62	63	64	66	52	59	83	64	61	
80	69	81	82	83	56	88	92	91	75	

We'll call these medians the representative of their respective sets.

Sorting up to five elements takes O(1) time, so in total this step is $\left\lceil \frac{n}{5} \right\rceil * O(1) = O(n)$ time.

More slowly…

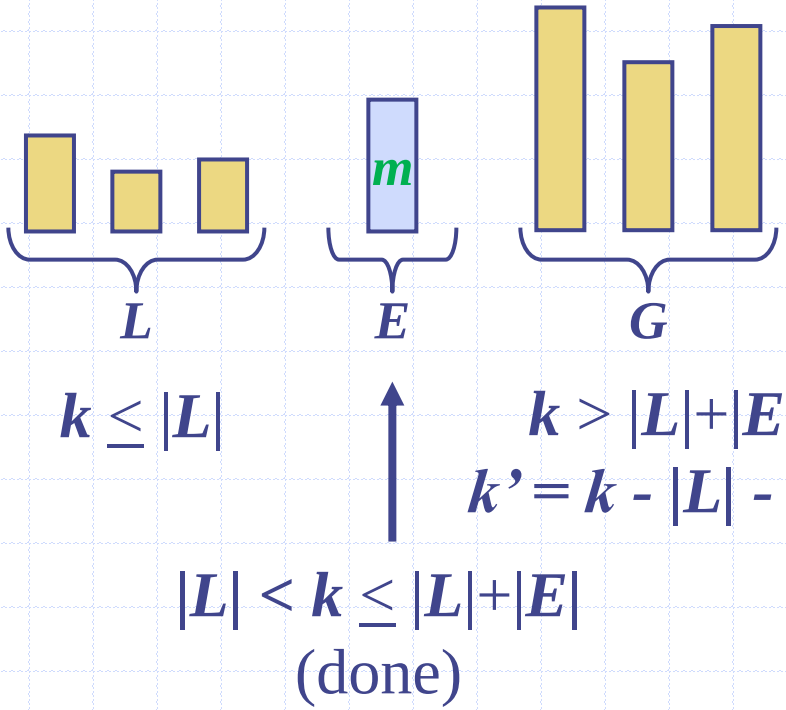
- Recursively find the median m of the representatives.

4	35	6	10	29	14	7	22	27	17	5
19	41	32	26	38	35	35	24	54	43	31
37	58	44	42	41	47	49	70	55	56	55
41		64	52	66	59	64	83	63	61	62
80		91	56	83	88	82	92	81	75	69

Since there are $\left\lceil \frac{n}{5} \right\rceil$ representatives, this takes time T($\left\lceil \frac{n}{5} \right\rceil$).

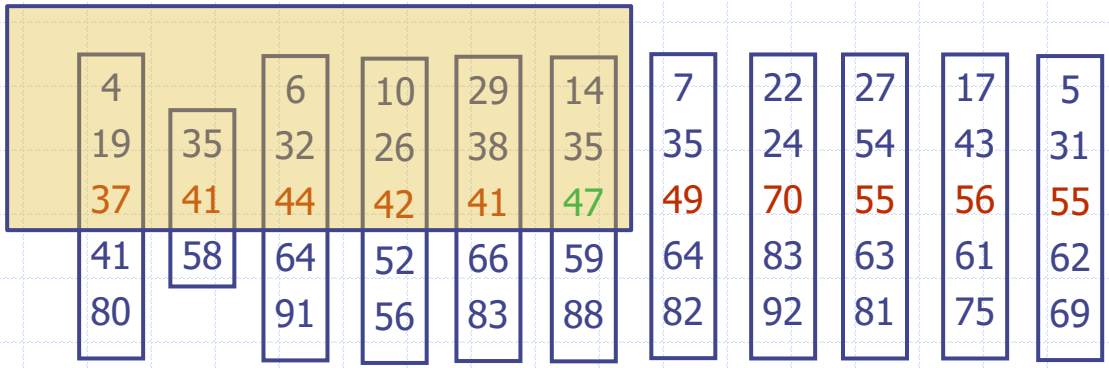
More slowly…

- Partition the entire set S into sets L, E, and G, using m as the pivot. This takes linear time.
- Then:
  - If k ≤ |L|, recurse to find the k-th element of L
  - If |L| < k ≤ |L|+|E|, m is the k-th element, and we are done.
  - If |L| + |E| < k, recurse to find the (k - |L| - |E|)-th element of G.



More slowly…

- The recursion takes time T(|L|) or T(|G|).
- But we can bound |L| and |G|:



- The $> \frac{n}{4}$ elements highlighted here must all be less than or equal to m. In the partition, these go into L or E. Thus $|G| \leq \frac{3n}{4}$.
- By a similar argument, $|L| \leq \frac{3n}{4}$.

More slowly…

- So the final recursion step takes time at most $T(\frac{3n}{4})$.

Adding up the work, we get:

$$T(n) \leq \begin{cases} b & n < 40 \\ T(\lceil n/5 \rceil) + T(3n/4) + bn & \text{otherwise} \end{cases}$$

This is a cool recurrence. We'll solve it by the guess-and-verify method. I'm going to guess that $T(n) \leq cn$. My induction hypothesis is that that is true for $n' \leq n$. The basis implies that my c must be at least b.

By the induction hypothesis,

$$T(n) \leq c\lceil n/5 \rceil + c(3n/4) + bn$$

More slowly…

- Since $\lceil n/5 \rceil < (n/5) + 1$, we have

$$T(n) \le c(n/5) + c + c(3n/4) + bn$$

or

$$\begin{aligned} T(n) &\le c(4n/20) + c(15n/20) + c + bn \\ &= c(19n/20) + c + bn \\ &\le c(19n/20) + (n/40)c + bn \\ &= c(39n/40) + bn \\ &\le cn, \quad \text{provided } b \le (1/40)c, \text{ or } c \ge 40b. \end{aligned}$$

Since we can choose our c to be equal to 40b, we have just shown that $T(n) \le cn$, by induction.

Thus, selection can be done deterministically in linear time.