CIT 596
Quickselect

Penn Engineering

QUICKSELECT

- Goal: select the k^{th} smallest ("rank k") element of an array.
- Option 1:
  - Use quicksort to sort the array A
  - Select the k^{th} smallest element $A[k-1]$
  - Time required: $O(n \log n)$
  - Are we doing unnecessary work?
- Key idea: when we partition A, we only need to recurse on the side of the k^{th} smallest element.

QUICKSELECT

Algorithm Idea: (We are looking for the kth-smallest element.)

1. Choose a "pivot" element x at random.

2. Compare all elements to the pivot

3. Partition all elements into two sets:
   - S (elements smaller than x) and L (elements larger than x)

4. Arrange the elements so that all elements in S come before x and all elements in L come after x, which leaves x in the i^{th} position (for some i)

5. If $k = i$, return x
   If $k < i$, recurse on the elements to the left of x
   If $k > i$, recurse on the elements to the right of x

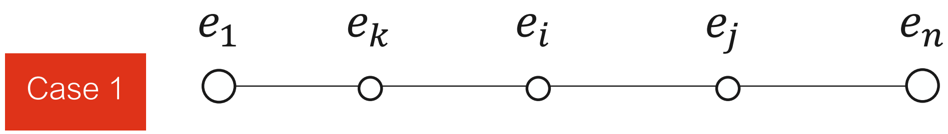
QUICKSELECT – RUNTIME

- Analyze with random variables:
  - Denote the k^{th} smallest element in the array as e_k
  - What is the probability that e_i and e_j are compared when selecting e_k?
  - There are 3 cases.

QUICKSELECT – RUNTIME

- Case 1: $k < i < j$

- e_i and e_j are compared when either e_i or e_j is selected as the pivot.
- e_i and e_j are not compared when any other element between e_k and e_j is selected.
- $\Pr[e_i \text{ and } e_j \text{ compared}] = \frac{2}{j-k+1}$

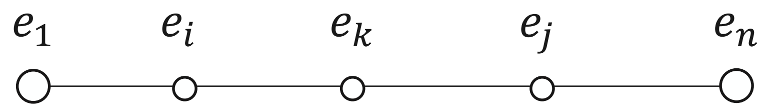


QUICKSELECT – RUNTIME

- Case 2: $i < k < j$

- Similarly, $\Pr[e_i \text{ and } e_j \text{ compared}] = \frac{2}{j-i+1}$

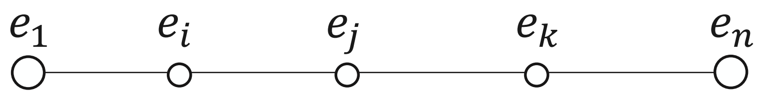
Case 2



- Case 3: $i < j < k$

- Similarly, $\Pr[e_i \text{ and } e_j \text{ compared}] = \frac{2}{k-i+1}$

Case 3



QUICKSELECT – RUNTIME

Runtime:

- Similar to quicksort analysis, we ask: "how many total comparisons are we making?"
- The summation over all pairs of elements e_i and e_j is split among the 3 cases:

$$\mathbf{E}[X] = \sum_{i<j\le k} \frac{2}{k-i+1} + \sum_{i<k<j} \frac{2}{j-i+1} + \sum_{k\le i<j} \frac{2}{j-k+1}$$

The sum is non-trivial to analyze. But using the same techniques as in quicksort analysis, this yields

$$\mathbf{E}[X] = O(n)$$