Order statics

Notes on Order Statistics and Selection Algorithms

---

1. Order Statistics

- **Definition:** The *i-th order statistic* of a set of n elements is the i-th smallest element.
  - Min = 1st order statistic.
  - Max = n-th order statistic.
  - Median = middle element ($\lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$).lecture05_order_statistics

---

2. The Selection Problem

- **Problem statement:** Given a set A of n elements and an integer i, find the element $x \in A$ such that x is greater than exactly $i-1$ elements of A.lecture05_order_statistics
- Special cases:
  - $i=1$: minimum.
  - $i=n$: maximum.
  - $i=(n+1)/2$: median.

---

3. Finding Min and Max

- **Naïve method:** Scan the array once, keep track of current best → O(n)O(n)O(n).lecture05_order_statistics
- **Both min & max:** Can be done in ~3n/23n/23n/2 comparisons by pairing elements.

🔷 4. Quickselect

Idea

- Based on **Quicksort's partition**:
  - Partition around a pivot.
  - If pivot lands at position iii: done.
  - If iii is smaller: recurse left.
  - If iii is larger: recurse right.
- Only recurse into *one side* → less work than Quicksort.lecture05_order_statistics

Why it works

- Each partition places the pivot in its **final correct position**.
- This lets us discard the irrelevant half immediately.

Complexity

- **Worst case:** pivot always smallest/largest → T(n)=T(n−1)+O(n)T(n) = T(n-1) + O(n)T(n)=T(n−1)+O(n) = O(n2)O(n^2)O(n2).lecture05_order_statistics
- **Average case:** random pivot → expected geometric shrinkage.
  - T(n)≤O(n)+T(3n/4)T(n) \leq O(n) + T(3n/4)T(n)≤O(n)+T(3n/4).
  - Expands to n+34n+(34)2n+…=O(n)n + \tfrac{3}{4}n + (\tfrac{3}{4})^2n + \dots = O(n)n+43n+(43)2n+…=O(n).lecture05_order_statistics
- **Space:** In-place, so O(1)O(1)O(1) auxiliary memory.

- **Tail recursion:** Quickselect naturally tail-recurses; can be written iterativelylecture05_order_statistics.

🔷 5. Heapselect

- Put all elements in a heap.
- Extract min iii times → i-th order statistic.
- Complexity: O(n+ilogn)O(n + i \log n)O(n+ilogn).lecture05_order_statistics
  - Efficient for small iii.
  - Worst case: O(nlogn)O(n \log n)O(nlogn).

🔷 6. Introselect

- Hybrid: Quickselect + Heapselect.
- If recursion depth too large (> O(logn)O(\log n)O(logn)), switch to Heapselect.
- **Guarantees:** O(nlogn)O(n \log n)O(nlogn) worst case, but expected O(n)O(n)O(n).lecture05_order_statistics
- Used in C++ `nth_element`.

🔷 7. Improving Quickselect with Pivot Choice

- **Random pivot:** reduces chance of worst-case, expected O(n)O(n)O(n).lecture05_order_statistics
- **Median of medians:** gives deterministic O(n)O(n)O(n).

🔷 8. Median of Medians

Process

1. Split array into groups of 5.
2. Find the median of each group (constant time per group).
3. Recursively select the **median of these medians** → pivot MMM.

4. Partition array using MMM.
5. Recurse into relevant side.lecture05_order_statistics

Why it works (mathematical guarantee)

- At least half of medians ≥ MMM.
- Each such median ≥ 2 other elements in its group → ≥3 elements total ≥ MMM.
- So ≥3n/103n/103n/10 elements ≤ MMM, and ≥3n/103n/103n/10 elements ≥ MMM.lecture05_order_statistics
- Therefore, each partition discards at least 30% of the array.

Complexity

T(n)≤T(n/5)+T(7n/10)+O(n).T(n) \leq T(n/5) + T(7n/10) + O(n).

T(n)≤T(n/5)+T(7n/10)+O(n).

By induction, T(n)≤10cn=O(n)T(n) \leq 10cn = O(n)T(n)≤10cn=O(n).
lecture05_order_statistics

Trade-off

- **Guaranteed O(n)O(n)O(n)** (worst-case).
- But larger constants than randomized Quickselect, so slower in practice unless adversarial input is likely.

🔷 9. Key Comparisons

- **Quickselect (random pivot):** Expected O(n)O(n)O(n), worst O(n2)O(n^2)O(n2).
- **Median of Medians:** Guaranteed O(n)O(n)O(n), but slower constants.
- **Heapselect:** O(n+ilogn)O(n+i \log n)O(n+ilogn), good for small iii.
- **Introselect:** O(n)O(n)O(n) expected, O(nlogn)O(n \log n)O(nlogn) guaranteed.

🔷 10. Core Definitions Recap

- **Order Statistic:** iii-th smallest element.
- **Selection Problem:** Find the iii-th order statistic.
- **Tail recursion:** Recursive call at the end → can be converted to loop.
- **Why correctness holds:** Partitioning guarantees pivot in final place; inductive step ensures recursive side shrinks properly.
- **Why complexities hold:** Cost expansions always form geometric series (average/randomized case) or guaranteed shrinkage (median of medians).

📘 Notes on Order Statistics and Selection Algorithms

🔷 1. Order Statistics

- **Definition:** The *i-th order statistic* of a set of nnn elements is the iii-th smallest element.
  - Min = 1st order statistic.
  - Max = nnn-th order statistic.
  - Median = middle element (⌊(n+1)/2(n+1)/2(n+1)/2⌋ or ⌈(n+1)/2(n+1)/2(n+1)/2⌉).lecture05_order_statistics

---

🔷 2. The Selection Problem

- **Problem statement:** Given a set AAA of nnn elements and an integer iii, find the element x∈Ax \in Ax∈A such that xxx is greater than exactly i−1i-1i−1 elements of AAA.lecture05_order_statistics
- Special cases:
  - i=1i=1i=1: minimum.
  - i=ni=ni=n: maximum.
  - i=(n+1)/2i=(n+1)/2i=(n+1)/2: median.

🔷 3. Finding Min and Max

- **Naïve method:** Scan the array once, keep track of current best → O(n)O(n)O(n).lecture05_order_statistics
- **Both min & max:** Can be done in ~3n/23n/23n/2 comparisons by pairing elements.

🔷 4. Quickselect

Idea

- Based on **Quicksort's partition**:
  - Partition around a pivot.
  - If pivot lands at position iii: done.
  - If iii is smaller: recurse left.
  - If iii is larger: recurse right.
- Only recurse into *one side* → less work than Quicksort.lecture05_order_statistics

Why it works

- Each partition places the pivot in its **final correct position**.
- This lets us discard the irrelevant half immediately.

Complexity

- **Worst case:** pivot always smallest/largest → T(n)=T(n−1)+O(n)T(n) = T(n-1) + O(n)T(n)=T(n−1)+O(n) = O(n2)O(n^2)O(n2).lecture05_order_statistics
- **Average case:** random pivot → expected geometric shrinkage.
  - T(n)≤O(n)+T(3n/4)T(n) \leq O(n) + T(3n/4)T(n)≤O(n)+T(3n/4).
  - Expands to n+34n+(34)2n+…=O(n)n + \tfrac{3}{4}n + (\tfrac{3}{4})^2n + \dots = O(n)n+43n+(43)2n+…=O(n).lecture05_order_statistics
- **Space:** In-place, so O(1)O(1)O(1) auxiliary memory.

- **Tail recursion:** Quickselect naturally tail-recurses; can be written iterativelylecture05_order_statistics.

🔷 5. Heapselect

- Put all elements in a heap.
- Extract min iii times → i-th order statistic.
- Complexity: O(n+ilogn)O(n + i \log n)O(n+ilogn).lecture05_order_statistics
  - Efficient for small iii.
  - Worst case: O(nlogn)O(n \log n)O(nlogn).

🔷 6. Introselect

- Hybrid: Quickselect + Heapselect.
- If recursion depth too large (> O(logn)O(\log n)O(logn)), switch to Heapselect.
- **Guarantees:** O(nlogn)O(n \log n)O(nlogn) worst case, but expected O(n)O(n)O(n).lecture05_order_statistics
- Used in C++ `nth_element`.

🔷 7. Improving Quickselect with Pivot Choice

- **Random pivot:** reduces chance of worst-case, expected O(n)O(n)O(n).lecture05_order_statistics
- **Median of medians:** gives deterministic O(n)O(n)O(n).

🔷 8. Median of Medians

Process

1. Split array into groups of 5.
2. Find the median of each group (constant time per group).
3. Recursively select the **median of these medians** → pivot MMM.

4. Partition array using MMM.
5. Recurse into relevant side.lecture05_order_statistics

Why it works (mathematical guarantee)

- At least half of medians ≥ MMM.
- Each such median ≥ 2 other elements in its group → ≥3 elements total ≥ MMM.
- So ≥3n/103n/103n/10 elements ≤ MMM, and ≥3n/103n/103n/10 elements ≥ MMM.lecture05_order_statistics
- Therefore, each partition discards at least 30% of the array.

Complexity

T(n)≤T(n/5)+T(7n/10)+O(n).T(n) \leq T(n/5) + T(7n/10) + O(n).

$T(n) \leq T(n/5) + T(7n/10) + O(n).$

By induction, T(n)≤10cn=O(n)T(n) \leq 10cn = O(n)T(n)≤10cn=O(n). lecture05_order_statistics

Trade-off

- **Guaranteed O(n)O(n)O(n)** (worst-case).
- But larger constants than randomized Quickselect, so slower in practice unless adversarial input is likely.

🔷 9. Key Comparisons

- **Quickselect (random pivot):** Expected O(n)O(n)O(n), worst O(n2)O(n^2)O(n2).
- **Median of Medians:** Guaranteed O(n)O(n)O(n), but slower constants.
- **Heapselect:** O(n+ilogn)O(n+i \log n)O(n+ilogn), good for small iii.
- **Introselect:** O(n)O(n)O(n) expected, O(nlogn)O(n \log n)O(nlogn) guaranteed.

🔷 10. Core Definitions Recap

- **Order Statistic:** iii-th smallest element.
- **Selection Problem:** Find the iii-th order statistic.
- **Tail recursion:** Recursive call at the end → can be converted to loop.
- **Why correctness holds:** Partitioning guarantees pivot in final place; inductive step ensures recursive side shrinks properly.
- **Why complexities hold:** Cost expansions always form geometric series (average/randomized case) or guaranteed shrinkage (median of medians).