Divide and Conquer

Selection

Selection Problem

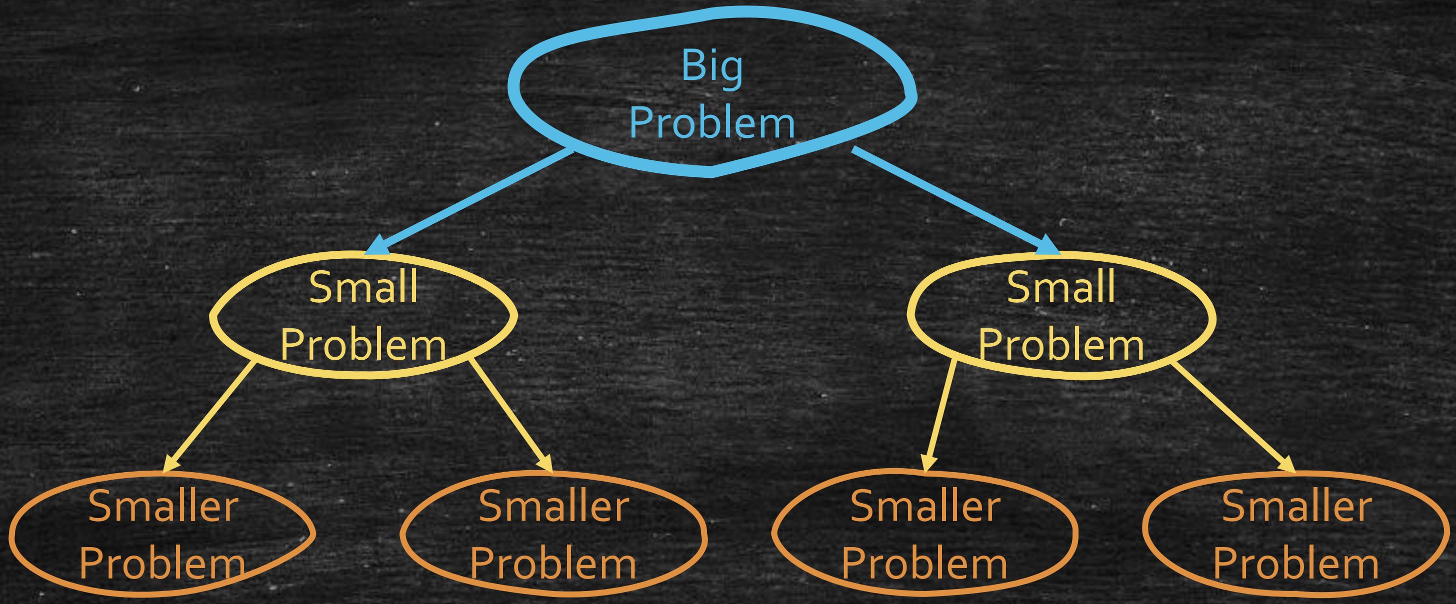
- **Input:** A set S of n integers $x_1, x_2, \ldots, x_n$ and an integer k
- **Output:** The k-th smallest integer x^* among $x_1, x_2, \ldots, x_n$

One-by-one Selection

- **Input:** A set S of n integers $x_1, x_2, \dots, x_n$ and an integer k
- **Output:** The k-th smallest integer x^* among $x_1, x_2, \dots, x_n$
- Plan 1
  - Select the smallest integer. $O(n)$
  - Select the 2nd smallest integer. $O(n-1)$
  - …
  - Select the k-th smallest integer. $O(n-k+1)$
  - Total running time $\boldsymbol{O(nk)}$

Sorting

- **Input:** A set S of n integers $x_1, x_2, \ldots, x_n$ and an integer k
- **Output:** The k-th smallest integer x^* among
- Plan 2
  - $x_1, x_2, \ldots, x_n$ Sort the integers in ascending order. $O(n \log n)$
  - Output the k-th integer. $O(1)$
  - Total running time $\boldsymbol{O(n \log n)}$



Ok! Let's move to divide and conquer!

Divide and Conquer

- **Input:** A set S of n integers $x_1, x_2, \ldots, x_n$ and an integer k
- **Output:** The k-th smallest integer x^* among $x_1, x_2, \ldots, x_n$
- Plan 2: Divide and Conquer
  - Divide:
    - Pick an arbitrary value v among $x_1, x_2, x_3, \ldots$.
    - Divide $x_1, x_2, x_3, \ldots$ into three subsets:
      - L : $x < v$
      - M : $x = v$
      - R : $x > v$
  - Recurse: find x^* in the subset contains x^*.
  - Combine: we already have x^*!

Divide

- Choose $v = 3$.

1	2	5	3	1	3	4	0

- What is L , M, and R?

1	2	5	3	1	3	4	0

- L:

1	2	1	0

- M:

3	3

- R:

5	4

Divide

- L:

1	2	1	0

- M:

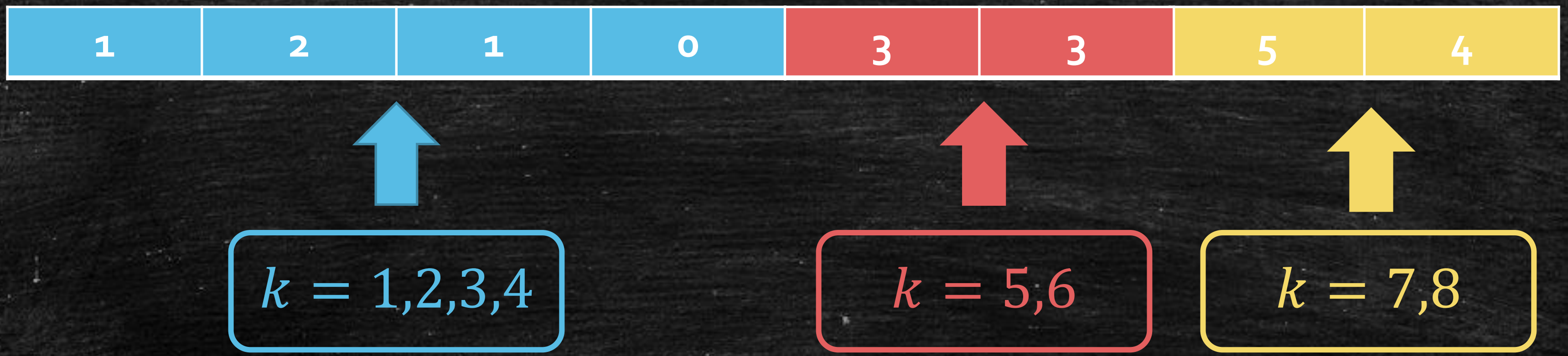
3	3

- R:

5	4

Recurse

- Roughly sorted list



Recurse

- How to find x^* in L,M,R?
  - Recall x^* is the k-th smallest integer in S.
- L:

1	2	1	0

  - x^* is the k-th integer in L
- M:

3	3

  - $x^* = 3$
- R:

5	4

  - x^* is the $(k - |\mathrm{L}| - |\mathrm{M}|)$-th integer in R

Example: $k = 4$

1	2	5	3	1	3	4	0

1	2	1	0

3	3

5	4

Example: $k = 4$

1	2	5	3	1	3	4	0

1	2	1	0

Example: $k = 4$

1	2	5	3	1	3	4	0

1	2	1	0

0	1	1	2

Example: $k = 4$

1	2	5	3	1	3	4	0

1	2	1	0

2

Example: $k = 4$

1	2	5	3	1	3	4	0

1	2	1	0

2

2

Output 2

Formalize

Function Select(S,k)

Is it correct?

- Divide:
  - Pick an arbitrary value v among $x_1, x_2, x_3, \ldots$.
  - Divide $x_1, x_2, x_3, \ldots$ into three subsets:
    - L : $x < v$,
    - M : $x = v$,
    - R : $x > v$.
- Recurse:
  - Recurse the subset contains x^*.
    - If $k \leq |L|$, output Select(L,k);
    - If $|L| < k \leq |L| + |M|$, output v;
    - If $|L| + |M| < k$, output Select($R, \mathrm{k} - |\mathrm{L}| - |\mathrm{M}|$).

Running Time

We want to know $T(n)$

Function Select(S,k)

- Divide:
  - Pick an arbitrary value v among $x_1, x_2, x_3, \ldots$.
  - Divide $x_1, x_2, x_3, \ldots$ into three subsets:
    - L : $x < v$,
    - M : $x = v$,
    - R : $x > v$.

Divide: $O(n)$

- Recurse:
  - Recurse the subset contains x^*.
    - If $k \le |L|$, output Select(L,k);
    - If $|L| < k \le |L| + |M|$, output v;
    - If $|L| + |M| < k$, output Select($R, \mathrm{k} - |\mathrm{L}| - |\mathrm{M}|$).

$T(|L|)$

$O(1)$

$T(|R|)$

Running Time

- $T(n) \leq O(n) + \max\{T(|L|), T(|R|)\}$

$$\leq O(n) + T(n-1)$$

$$\leq O(n) + \overbrace{O(n-1) + T(n-2)} \leq \cdots$$

$$= O(n) + O(n-1) + O(n-2) + \cdots + O(1) = \boldsymbol{O(n^2)}$$

- Very Bad!
  - One-by-one: $O(nk)$
  - Sorting: $O(n\log n)$

Fact

- $|L| + |M| + |R| = |S| = n$
- $|M| \geq 1$
- $|L|, |R| \leq n-1$

Is it really that bad?

- Yes, the unluckiest case:
  - $k = 1$
  - Each time, v is the largest integer.
  - $T(n) = O(n) + T(n-1) = O(n) + O(n-1) + T(n-2) = \cdots = O(n^2)$
- What if we are **super lucky**?
  - Each time, v is in the middle.
  - $T(n) = T\left(\frac{n}{2}\right) + O(n) = T\left(\frac{n}{4}\right) + O\left(\frac{n}{2}\right) + O(n) = \cdots = O(n).$
- What if we are lucky?
  - Each time, v is in the middle range $[\frac{1}{3}n, \frac{2}{3}n]$.
  - $T(n) = T\left(\frac{2}{3}n\right) + O(n) = T\left(\frac{4}{9}n\right) + O\left(\frac{2}{3}n\right) + O(n) = \cdots = O(n).$
- Idea: to make us reasonably lucky in average by randomness.

What is the next?

- Improving the running time **with** randomness.
- Improving the running time **without** randomness.

Using Randomness!

Function Select(S,k)

- Divide:
  - Pick an arbitrary value v among $x_1, x_2, x_3, \ldots$.
  - Divide $x_1, x_2, x_3, \ldots$ into three subsets:
    - L : $x < v$,
    - M : $x = v$,
    - R : $x > v$.
- Recurse:
  - Recurse the subset contains x^*.
    - If $k \leq |L|$, output Select(L,k);
    - If $|L| < k \leq |L| + |M|$, output v;
    - If $|L| + |M| < k$, output Select(R, $\mathrm{k} - |\mathrm{L}| - |\mathrm{M}|$).

Using Randomness!

Function Select(S,k)

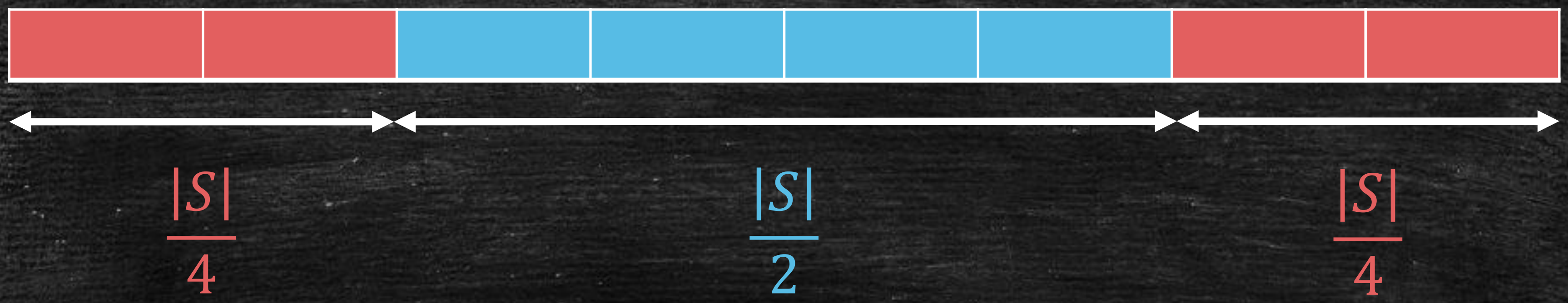
- Divide:
  - Pick an ~~arbitrary~~ value v among $x_1, x_2, x_3, \ldots$.
  - Divide $x_1, x_2, x_3, \ldots$ into three subsets:
    - L : $x < v$,
    - M : $x = v$,
    - R : $x > v$.
- Recurse:
  - Recurse the subset contains x^*.
    - If $k \le |L|$, output Select(L,k);
    - If $|L| < k \le |L| + |M|$, output v;
    - If $|L| + |M| < k$, output Select(R, $\mathrm{k} - |\mathrm{L}| - |\mathrm{M}|$).

Quick Selection

Function Select(S,k)

- Divide:
  - Pick a **random** value v among $x_1, x_2, x_3, \ldots$.
  - Divide $x_1, x_2, x_3, \ldots$ into three subsets:
    - L : $x < v$,
    - M : $x = v$,
    - R : $x > v$.
- Recurse:
  - Recurse the subset contains x^*.
    - If $k \leq |L|$, output Select(L,k);
    - If $|L| < k \leq |L| + |M|$, output v;
    - If $|L| + |M| < k$, output Select($R, \mathrm{k} - |\mathrm{L}| - |\mathrm{M}|$).

When we are lucky



- : Lucky pivot area
- : Bad pivot area
- Fact 1: With $\frac{1}{2}$ probability, we are lucky!
- Fact 2: If we are always lucky, $T(n) = T\left(\frac{3n}{4}\right) + O(n) = O(n)$

Analysis

- $\tau(n)$: Time we reduce n to $\frac{3n}{4}$
- $T(n) = \tau(n) + T(\frac{3n}{4})$
- $E[\tau(n)]$: The expected time we reduce n to $\frac{3n}{4}$
- $E[T(n)] = E\left[\tau(n) + T\left(\frac{3n}{4}\right)\right]$

  $= E[\tau(n)] + E\left[T\left(\frac{3n}{4}\right)\right]$
- $E[\tau(n)] = O(n)$
- $E[T(n)] = O(n) + E\left[T\left(\frac{3n}{4}\right)\right] = \boldsymbol{O(n)}$

Fact

Since we are lucy with probably $\frac{1}{2}$, so the expected number of rounds it takes to become lucky is 2.

Evaluate Random Algorithm by Expected Running Time!

Other Viewpoints

- Worst Case Running Time
  - $O(n^2)$
- The Probability it runs in $O(n)$?

What if we do not want randomness?

Throw Randomness!

Function Select(S,k)

- Divide:
  - Pick a **random** value v among $x_1, x_2, x_3, \ldots$.
  - Divide $x_1, x_2, x_3, \ldots$ into three subsets:
    - L : $x < v$,
    - M : $x = v$,
    - R : $x > v$.
- Recurse:
  - Recurse the subset contains x^*.
    - If $k \leq |L|$, output Select(L,k);
    - If $|L| < k \leq |L| + |M|$, output v;
    - If $|L| + |M| < k$, output Select($R, \mathrm{k} - |\mathrm{L}| - |\mathrm{M}|$).

Throw Randomness!

Function Select(S,k)

- Divide:
  - Pick a ~~random~~ value v among $x_1, x_2, x_3, \dots$.
  - Divide $x_1, x_2, x_3, \dots$ into three subsets:
    - L : $x < v$,
    - M : $x = v$,
    - R : $x > v$.
- Recurse:
  - Recurse the subset contains x^*.
    - If $k \leq |L|$, output Select(L,k);
    - If $|L| < k \leq |L| + |M|$, output v;
    - If $|L| + |M| < k$, output Select($R, \mathrm{k} - |\mathrm{L}| - |\mathrm{M}|$).

Median of medians (1973)

Blum, M.; Floyd, R. W.; Pratt, V. R.; Rivest, R. L.; Tarjan, R. E.

Function Select(S,k)

- Divide:
  - Pick a **good pivot** value v among $x_1, x_2, x_3, \ldots$.
  - Divide $x_1, x_2, x_3, \ldots$ into three subsets:
    - L : $x < v$,
    - M : $x = v$,
    - R : $x > v$.
- Recurse:
  - Recurse the subset contains x^*.
    - If $k \le |L|$, output Select(L,k);
    - If $|L| < k \le |L| + |M|$, output v;
    - If $|L| + |M| < k$, output Select($R, \mathrm{k} - |\mathrm{L}| - |\mathrm{M}|$).

Trade-off

- The time of finding a good pivot.
- The quality of the pivot.
- We can find the median by sorting the array, but it takes too much time.
- We can find an arbitrary pivot in $O(1)$ but it may be too bad.
- Look at the recursive running time:
  - $T(n) = T(c \cdot n) + findPivot + O(n)$.

How to select a good pivot?

- Partition S into subsets with size 5.

12	23	53	84	90	32	4	5	32	63	13	14	8	2	42

5 5 5

How to select a good pivot?

- Partition S into subsets with size 5.

12	23	53	84	90

32	4	5	32	63

13	14	8	2	42

How to select a good pivot?

- Partition S into subsets with size 5.

12	23	53	84	90

32	4	5	32	63

13	14	8	2	42

- Find the medians of them: v_1, v_2, v_3
  - 53, 32, 13

How to select a good pivot?

- Partition S into subsets with size 5.

12	23	53	84	90

32	4	5	32	63

13	14	8	2	42

- Find the medians of them: v_1, v_2, v_3
  - 53, 32, 13
- Fix v to be the median of v_1, v_2, v_3
  - $v = 32$

How long it takes?

- Partition S into subsets with size 5.

$O(n)$

12	23	53	84	90
32	4	5	32	63
13	14	8	2	42

- Find the medians of them: v_1, v_2, v_3 — $O(n)$
  - 53, 32, 13
- Fix v to be the median of v_1, v_2, v_3 — $T(n/5)$
  - $v = 32$

Why it is good?

- It should be in the middle range
- Why? Two questions
  - How many integers should be no greater than v?
  - How many integers should be no less than v?

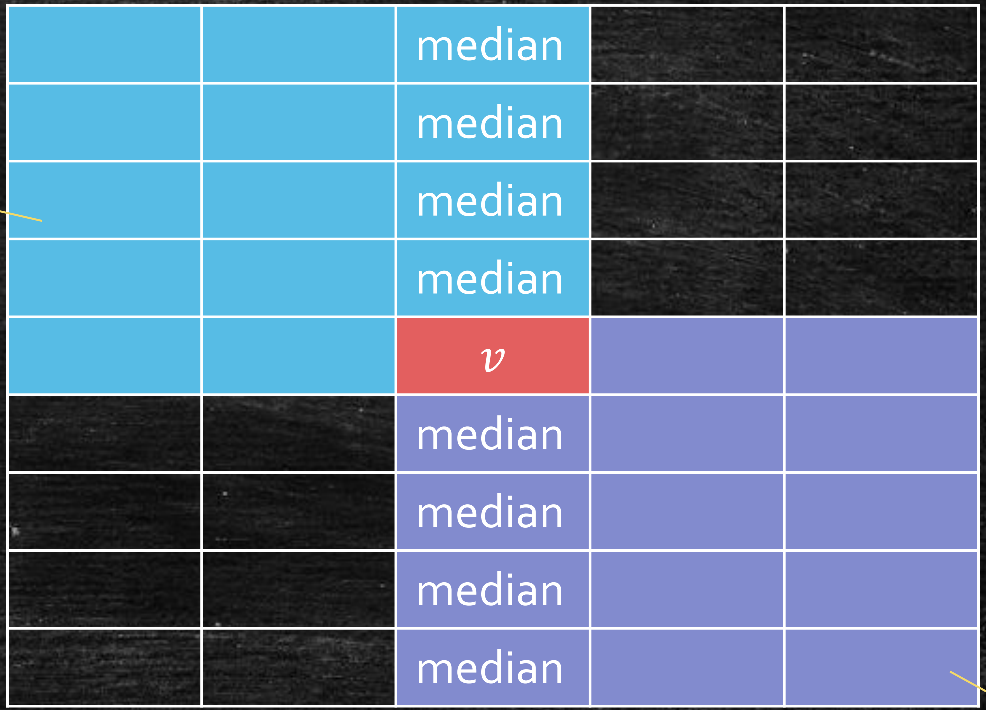
Answer them step by step

- Partition S into subsets

12	23	53	84	90

32	4	5	32	63

13	14	8	2	42



- Answer
  - We have $\frac{n}{5}$ groups, so $\frac{n}{5}$ medians.
  - v is no greater than $n/10$ medians, no less than $n/10$ medians.
  - Each median is no greater than 2 integers, no less than 2 integers.
  - v is no greater than $\frac{3n}{10}$ integers, no less than $3n/10$ integers.

The running time

Function Select(S,k)

- Divide:
  - Pick a **good pivot** value v among $x_1, x_2, x_3, \ldots$.
  - Divide $x_1, x_2, x_3, \ldots$ into three subsets:
    - L : $x < v$,
    - M : $x = v$,
    - R : $x > v$.
- Recurse:
  - Recurse the subset contains x^*.
    - If $k \leq |L|$, output Select(L,k);
    - If $|L| < k \leq |L| + |M|$, output v;
    - If $|L| + |M| < k$, output Select($R, \mathrm{k} - |\mathrm{L}| - |\mathrm{M}|$).

$\boldsymbol{T\left(\frac{n}{5}\right) + O(n)}$

$\boldsymbol{O(n)}$

$\boldsymbol{T(|L|) \leq T(n - \frac{3}{10}n)}$

$\boldsymbol{T(|R|) \leq T(n - \frac{3}{10}n)}$

Guess time!

$$\boldsymbol{T(n)=T(0.2n)+T(0.7n)+O(n)}$$

Observation: Comparing to Master Theorem

Level 0 — n — $O(n)$

Level 1 — $0.2n$ $0.7n$ — $O(0.9n)$

Level 2 — $0.04n$ $0.14n$ …… $0.14n$ $0.49n$ — $O(0.81n)$

Level k — …… — $O(0.9^k n)$

Level $\log_{\frac{10}{7}} n$ — 1 1 …… 1 1 — $O(0.9^{\log_{\frac{10}{7}} n} n)$

We allow some problem with size≤ 1.

Make a guess

$$T(n) = T(0.2n) + T(0.7n) + O(n)$$

Assume
$O(n) \leq Cn$

- Guess: $T(n) \leq Bn$!
- Try to prove it inductively
  - Basic step: $T(1) = 1 \leq Bn$
  - Inductive step:

$$\begin{aligned} T(n) &\leq T(0.2\text{n}) + T(0.7\text{n}) + Cn \\ &\leq 0.9Bn + Cn \\ &\leq Bn \end{aligned}$$

It holds when
$B \geq 10C$

- We have $T(n) \leq 10Cn = O(n)$

Remember not to use induction with Big O notations!

One more Question

What if we group them by 2,3,4,5,…?

Today’s goal

- Learn the quick selection algorithm
- Learn to make it polynomial by randomness (in expectation) **analytically**
- Learn to make it polynomial by median of medians **analytically**
- Remember to try to group by 2,3,4,5,6…