Linear-time Median

$\lceil n/2 \rceil$

Def: **Median** of elements $A=a_1, a_2, \ldots, a_n$ is the (n/2)-th smallest element in A.

How to find median?

- sort the elements, output the elem. at (n/2)-th position
  - running time ? $O(n \log n)$

Linear-time Median

Def: **Median** of elements $A=a_1, a_2, \ldots, a_n$ is the (n/2)-th smallest element in A.

How to find median?

- sort the elements, output the elem. at (n/2)-th position
  - running time: $\Theta(n \log n)$
- we will see a faster algorithm
  - will solve a more general problem:

    SELECT (A, k): returns the k-th smallest element in A

Linear-time Median

Idea: Suppose A = k = 20

22,5,10,11,23,15,9,8,2,0,4,20,25,1,29,24,3,12,28,14,27,19,17,21,18,6,7,13,16,26

11 8 20 14 19 13 pivot

Arearranged = 5,10,11,9,8,2,0,4,1,3,12,6,7, 13, 26,16,18,21,17,19,27,14,28,24,29,25,20,15,23,22

recurse with k=6 (=20-14)

fnc select(A,k):

1. split the input into groups of 5
2. find the median of each group
3. find the median of the medians → pivot
4. rearrange A to have numbers < pivot on the left, > pivot on the right
5. let p be the position of the pivot

6. if k=p:
   return pivot
7. if k<p:
   return select(Arearranged[1...p-1], k)
8. if k>p:
   return select(Arearranged[p+1,...n], k-p)

Linear-time Median

```
SELECT (A, k)
1. split A into n/5 groups of five elements
2. let b_i be the median of the i-th group
3. let B = [b_1, b_2, …, b_{n/5}]
4. medianB = SELECT (B, B.length/2)
5. rearrange A so that all elements smaller than
   medianB come before medianB, all elements
   larger than medianB come after medianB, and
   elements equal to medianB are next to medianB
6. j = position of medianB in rearranged A
  (if more medianB's, then take the closest
   position to n/2)
7. if (k < j) return SELECT ( A[1…j-1], k )
8. if (k = j) return medianB
9. if (k > j) return SELECT ( A[j+1…n], k-j )
```

T(n)

O(n)

O(n)

∀i

bubble sort
brute force for each group:
O(1) per group

T(n/5)

O(n)

O(n)

T(?)

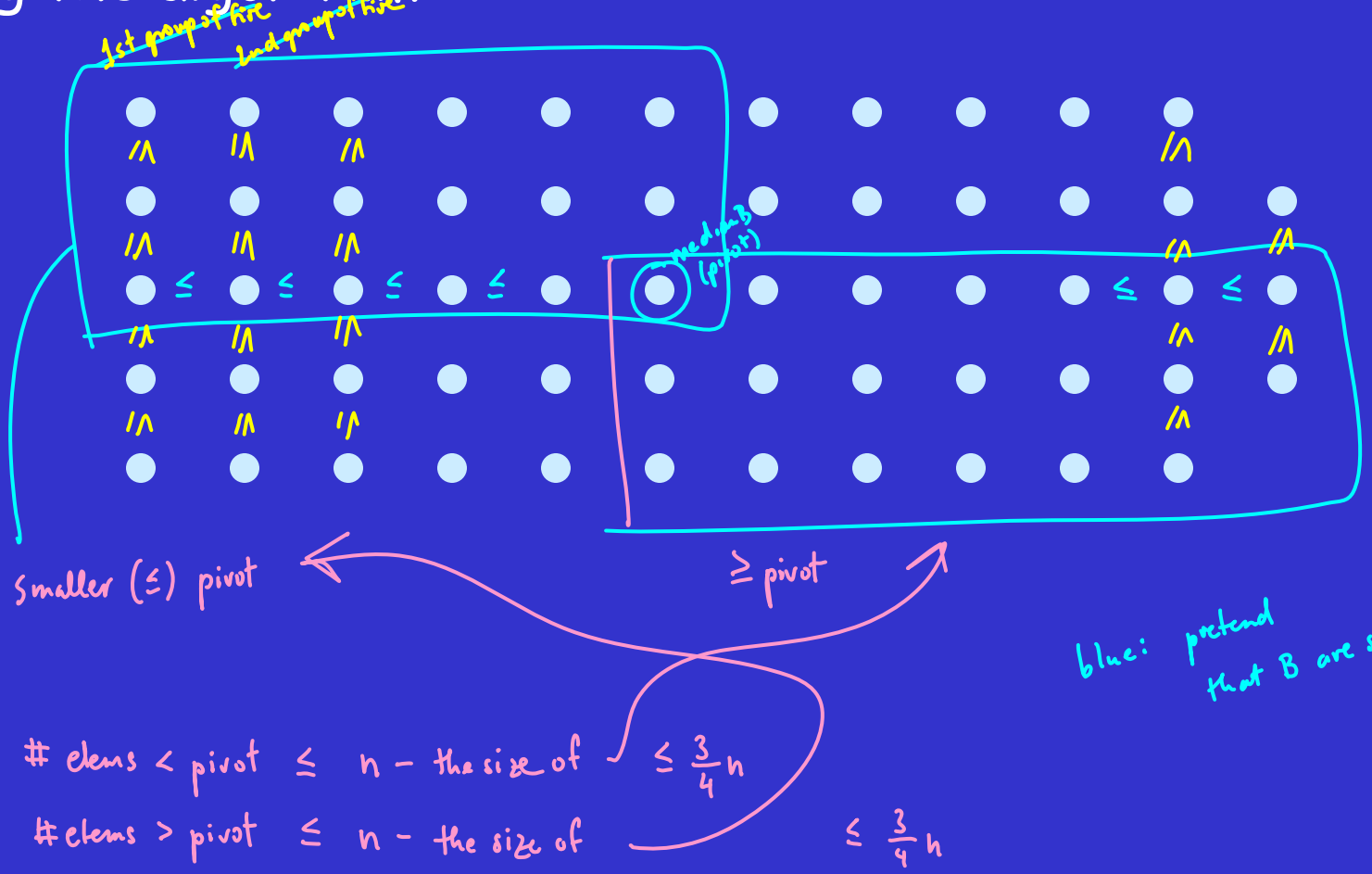
O(1)

T(?)

Linear-time Median

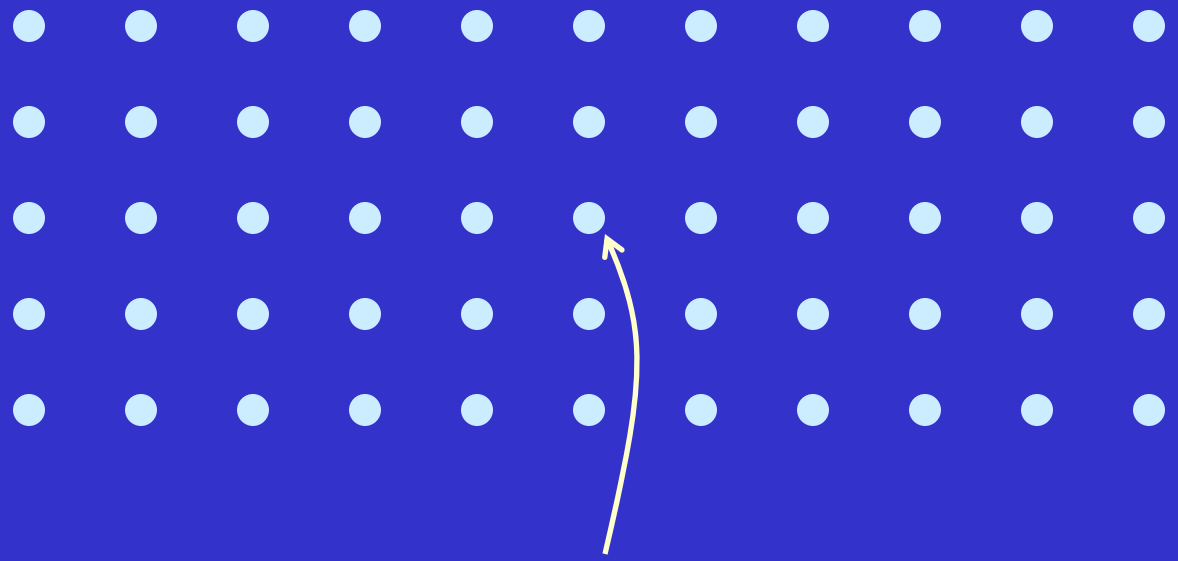
$A = [a_1, \ldots, a_n]$

Running the algorithm:



Linear-time Median

Running the algorithm:



Rearrange columns so that medianB in the "middle."

Recurrence: $T(n) \leq T(n/5) + T(\frac{3}{4}n) + c \cdot n$

Linear-time Median

Recurrence: $T(n) \leq T(n/5) + T(3n/4) + cn$ if $n > 5$

$T(n) \leq c$ if $n \leq 5$

Claim: There exists a constant d such that $T(n) \leq dn$.

BASE CASE: $n \leq 5$: $T(n) \leq c$ we want to show $T(n) \leq d \cdot n$ for some d

we need: $d \geq c$

IND. CASE: $n > 5$:

$T(n) \overset{\text{know}}{\leq} T(n/5) + T(3n/4) + c \cdot n$ want: $T(n) \leq dn$ for some d

by IH: $T(n/5) \leq d \cdot n/5$

$T(3/4 \cdot n) \leq d \cdot 3n/4$

$\hookrightarrow \leq d \cdot \frac{n}{5} + d \cdot \frac{3n}{4} + c \cdot n = \left(\frac{19}{20} \cdot d + c\right) \cdot n \overset{\text{want}}{\leq} d \cdot n$

want: $\frac{19}{20} d + c \leq d \Rightarrow \boxed{d \geq 20c}$

Randomized Linear-time Median

Idea:
Instead of finding medianB, take a random element from A.

```
SELECT-RAND (A, k)
1. x = a_i where i = a random number from {1,…,n}
2. rearrange A so that all elements smaller than
   x come before x, all elements larger than x
   come after x, and elements equal to x are
   next to x
3. j = position of x in rearranged A (if more
   x's, then take the closest position to n/2)
4. if (k < j) return SELECT-RAND ( A[1…j-1], k )
5. if (k = j) return medianB
6. if (k > j) return SELECT-RAND ( A[j+1…n], k-j)
```

Randomized Linear-time Median

Worst case running time: $O(n^2)$.

```
SELECT-RAND (A, k)
1. x = a_i where i = a random number from {1,…,n}
2. rearrange A so that all elements smaller than
   x come before x, all elements larger than x
   come after x, and elements equal to x are
   next to x
3. j = position of x in rearranged A (if more
   x's, then take the closest position to n/2)
4. if (k < j) return SELECT-RAND ( A[1…j-1], k )
5. if (k = j) return medianB
6. if (k > j) return SELECT-RAND ( A[j+1…n], k-j)
```

Randomized Linear-time Median

Worst case running time: $O(n^2)$.

Claim: **Expected** running time is $O(n)$.

Master Theorem

Let $a \geq 1$ and $b>1$ be constants, $f(n)$ be a function and for positive integers we have a recurrence for T of the form

$$T(n) = a\,T(n/b) + f(n),$$

where n/b is rounded either way.

E.g. $T(n) = T(\frac{n}{2}) + c \cdot n$ $\quad a=1 \quad b=2 \quad f(n)=c \cdot n$ $\quad \log a/\log b = 0$

E.g. $T(n) = T(\frac{n}{2}) + c$ $\quad a=1 \quad b=2 \quad f(n)=c$ $\quad \log a/\log b = 0$ ↘ bin. search, $T(n) = \Theta(\log n)$

E.g.: $T(n) = 2 \cdot T(n/2) + c \cdot n$ $\quad a=2 \quad b=2 \quad f(n)=c \cdot n$ $\quad \log a/\log b = 1$ $\quad f(n) = \Theta(n)$ $\quad T(n) = \Theta(n^1 \cdot \log n)$

Then,

- If $f(n) = O(n^{\log a/\log b - e})$ for some constant $e > 0$, then $T(n) = \Theta(n^{\log a/\log b})$.

  $T(n) = T(n-1) + c \cdot n$ ← the Master Thm does not apply!

  $T(n) = 4\,T(n/2) + c \cdot n$ $\quad a=4 \quad b=2 \quad f(n)=c \cdot n \quad \log a/\log b = 2$

  $f(n) = O(n^{2-\varepsilon})$ e.g. $\varepsilon = \frac{1}{2}$ → $T(n) = \Theta(n^2)$

- If $f(n) = \Theta(n^{\log a/\log b})$ then $T(n) = \Theta(n^{\log a/\log b} \log n)$.

- If $f(n) = \Omega(n^{\log a/\log b + e})$ for some constant $e > 0$, and if $af(n/b) \leq c_2 f(n)$ for some constant $c_2 < 1$ (and all sufficiently large n), then $T(n) = \Theta(f(n))$.

  for ε (e) $= 1/2$ (e.g.)

  ↘ $1 \cdot c \cdot \frac{n}{2} \leq c_2 \cdot c \cdot n$

  e.g. $c_2 = \frac{1}{2}$

  $T(n) = \Theta(n)$