Divide-and-conquer paradigm

Divide-and-conquer.

- Divide up problem into several subproblems.
- Solve each subproblem recursively.
- Combine solutions to subproblems into overall solution.

Most common usage.

- Divide problem of size n into two subproblems of size $n/2$ in linear time.
- Solve two subproblems recursively.
- Combine two solutions into overall solution in linear time.

Consequence.

- Brute force: $\Theta(n^2)$.
- Divide-and-conquer: $\Theta(n \log n)$.

attributed to Julius Caesar

Median and selection problems

مسئله تعیین میانه و مسئله انتخاب آمارهٔ ترتیبی

ترتیب کلی

Selection. Given n elements from a totally ordered universe, find k^{th} smallest.

- Minimum: $k = 1$; maximum: $k = n$.
- Median: $k = \lfloor (n + 1) / 2 \rfloor$.
- $O(n)$ compares for min or max.
- $O(n \log n)$ compares by sorting $\Rightarrow O(1)$
- $O(n \log k)$ compares with a binary heap.

insert /delete

$O(n^2)$ آرایه مرتب نشده

$\leq$

$n=5$

$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5$

$n=6$

$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6$

آمارهٔ ترتیبی k ام

ordered statistics

Applications. Order statistics; find the "top k"; bottleneck paths, …

Q. Can we do it with $O(n)$ compares?

A. Yes! Selection is easier than sorting.

درعمل، آنچه چارکها/دهک ها/صدکهای آماری را بدست می‌آورد همانا محاسبه آمارهٔ ترتیبی متناظر است. مثال وقتی صحبت از چارک سوم می‌کنیم منظور آمارهٔ ترتیبی $k = \lfloor 3n/4 \rfloor$ ام است

Expected Value = Mean = Average = میانگین

$\lfloor \frac{n+1}{2} \rfloor$ آمارهٔ ترتیبی = Median = میانه

$A = \langle 1, 2, 3, 50 \rangle$ pivot = میانگین $\Rightarrow$ افراز $= \langle 1, 2, 3 \mid 50 \rangle$

حالت ایده‌آل زمانی است که $\Rightarrow$ افراز $= \langle$ نصف | نصف $\rangle$

pivot = میانه

یک pivot خوب انتخاب کن تا الگوریتم تعیین میانه کارآمد شود

نیاز به میانه دارد

$\rightarrow$ دور و تسلسل ?

Quickselect

یادآوری. مسئلهٔ میانه و آماره‌های، راه‌حل ساختمان داده‌ای بسیار کارآمدی دارد

الگوریتم افراز در کتاب CLRS معرفی شده

3-way partition array so that:

- Pivot element p is in place.
- Smaller elements in left subarray L.
- Equal elements in middle subarray M.
- Larger elements in right subarray R.

افراز ۳ بخشی

اما در این مبحث ما به دنبال رویکردهای الگوریتمی صرف هستیم

QuickSelect $O(n)$ در میانگین

QuickSort $O(n \lg n)$ در میانگین

Recur in one subarray—the one containing the k^{th} smallest element.

QUICK-SELECT (A, k)

آمارهٔ ترتیبی kام در آرایهٔ A بدست می‌آورد

Pick pivot $p \in A$ uniformly at random.

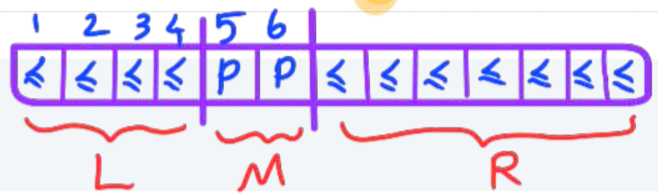
$(L, M, R) \leftarrow$ PARTITION-3-WAY (A, p).

3-way partitioning can be done in-place (using n–1 compares)

IF $k \leq |L|$ RETURN QUICK-SELECT (L, k).

ELSE IF $k > |L| + |M|$ RETURN QUICK-SELECT $(R, k - |L| - |M|)$

ELSE RETURN p.



Quickselect analysis

آنالیز الگوریتم QuickSelect مشابه آنالیز QuickSort است

Intuition. Split candy bar uniformly $\Rightarrow$ expected size of larger piece is ¾.

$$T(n) \le T(\tfrac{3}{4} n) + n \quad \Rightarrow \quad T(n) \le 4n$$

Def. $T(n, k)$ = expected # compares to select k^{th} smallest in an array of size $\le n$.

Def. $T(n) = \max_k T(n, k)$.

$T(n) = T(n-1) + O(n)$

Proposition. $T(n) \le 4n$.

Pf. [by strong induction on n]

- Assume true for $1, 2, \ldots, n-1$.
- $T(n)$ satisfies the following recurrence:

can assume we always recur on largest subarray since T(n) is monotonic and we are trying to get an upper bound

$$\begin{aligned}
T(n) &\le n + 2/n\,[\,T(n/2) + \ldots + T(n-3) + T(n-2) + T(n-1)\,] \\
&\le n + 2/n\,[\,4n/2 + \ldots + 4(n-3) + 4(n-2) + 4(n-1)\,] \\
&= n + 4\,(3/4\,n) \\
&= 4n. \quad \blacksquare
\end{aligned}$$

$\begin{cases} |L| = n-1 \\ |M| = 1 \\ |R| = 0 \end{cases}$ worst case

tiny cheat: sum should start at $T(\lfloor n/2 \rfloor)$

In worst case QuickSelect spends $O(n^2)$ computions.

Selection in worst case linear time

ضمن اینکه از معضل دور و تسلسل اجتناب می‌کنیم

Goal. Find pivot element p that divides list of n elements into two pieces so that each piece is guaranteed to have $\leq 7/10\ n$ elements.

$\max(|L|, |R|) \leqslant \frac{7n}{10}$

L | M | R

Q. How to find approximate median in linear time?

A. Recursively compute median of sample of $\leq 2/10\ n$ elements.

L ≤ R ≤

هزینه بدست آوردن میانه‌ی میانه‌ها

$$T(n) \leqslant \begin{cases} \Theta(1) & \text{if } n = 1 \\ T(7/10\ n) + T(2/10\ n) + \Theta(n) & \text{otherwise} \end{cases}$$

two subproblems
of different sizes!

هزینه افراز
+
هزینه الگوریتم عجیب

بکار گرفتن
قضیه اساسی را نمی‌توان
اما مستقیما با استقرا
اثبات می‌شود

$\Longrightarrow T(n) = \Theta(n)$ در بدترین حالت

Choosing the pivot element

رویکرد رایج در کاربست نگرش تقسیم و غلبه
شکستن مسئله n به تعداد ثابتی (b) زیرمسئله هر یک به ابعاد n/b است

- Divide n elements into $\lfloor n/5 \rfloor$ groups of 5 elements each (plus extra).

این بار از زاویهٔ متفاوتی به فرایند تعریف زیرمسئله‌ها نگاه می‌کنیم

"مسئله با ابعاد n را به n/b زیرمسئله هر یک به ابعاد b می‌شکنیم"

29	10	38	37	2	55	18	24	34	35	36
22	44	52	11	53	12	13	43	20	4	27
28	23	6	26	40	19	1	46	31	49	8
14	9	5	3	54	30	48	47	32	51	21
45	39	50	15	25	16	41	17	22	7	

N = 54

5 5 5 . . . . . . 5 4

Choosing the pivot element

Step 1
- Divide n elements into $\lfloor n/5 \rfloor$ groups of 5 elements each (plus extra).

Step 2
- Find median of each group (except extra).

درواقع در گام غلبه، مستقیم یک مسئله ساده (حالت adhoc) را داریم:

هر زیرآرایهٔ 5 تایی با یک الگوریتم سادهٔ مرتب سازی مرتب می کنیم و عنصر وسط را بعنوان میانه تعیین می کنیم

medians

29	10	**38**	37	2	55	**18**	24	34	**35**	36
22	44	52	11	53	12	13	**43**	20	4	27
28	**23**	6	26	**40**	**19**	1	46	**31**	49	8
14	9	5	3	54	30	48	47	32	51	21
45	39	50	**15**	25	16	41	17	22	7	

تعداد دسته ها $\lfloor n/5 \rfloor$ = تعداد میانه ها

N = 54

B مجموعهٔ میانه ها

28	23	38	15	40	---

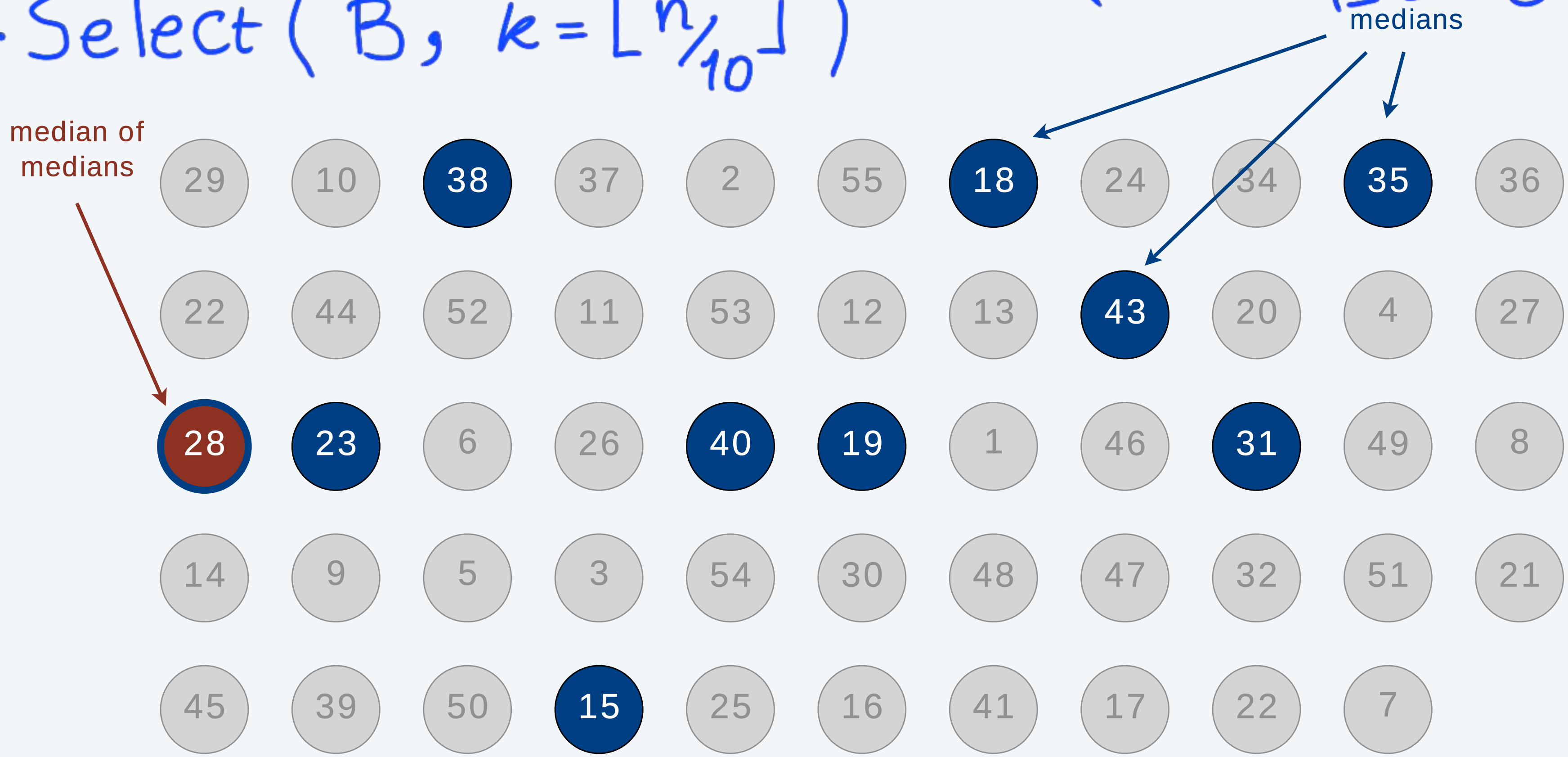
Choosing the pivot element

محاسبهٔ میانهٔ میانه‌ها

- Divide n elements into $\lfloor n/5 \rfloor$ groups of 5 elements each (plus extra).
- Find median of each group (except extra).
- Find median of $\lfloor n/5 \rfloor$ medians recursively.
- Use median-of-medians as pivot element.

(در ادامهٔ گام غلبه، میانهٔ مجموعهٔ میانه‌ها
را با بازفراخوانی الگوریتم Select)

$$p = \text{Select}\left(B,\ k = \lfloor n/10 \rfloor\right)$$



N = 54

Median-of-medians selection algorithm

MOM-SELECT (A, k)

$n \leftarrow |A|$.

IF $n < 50$ RETURN k^{th} smallest of element of A via mergesort.

Group A into $\lfloor n / 5 \rfloor$ groups of 5 elements each (plus extra).

$B \leftarrow$ median of each group of 5. $\rightarrow O(n)$

$p \leftarrow$ MOM-SELECT$(B, \lfloor n / 10 \rfloor)$ ← median of medians $\rightarrow T\left(\frac{n}{5}\right)$

$\frac{2n}{10}$

محاسبات لازم برای بدست آوردن pivot "خوب"

$(L, M, R) \leftarrow$ PARTITION-3-WAY (A, p).

IF $k \le |L|$ RETURN MOM-SELECT (L, k).

ELSE IF $k > |L| + |M|$ RETURN MOM-SELECT $(R, k - |L| - |M|)$

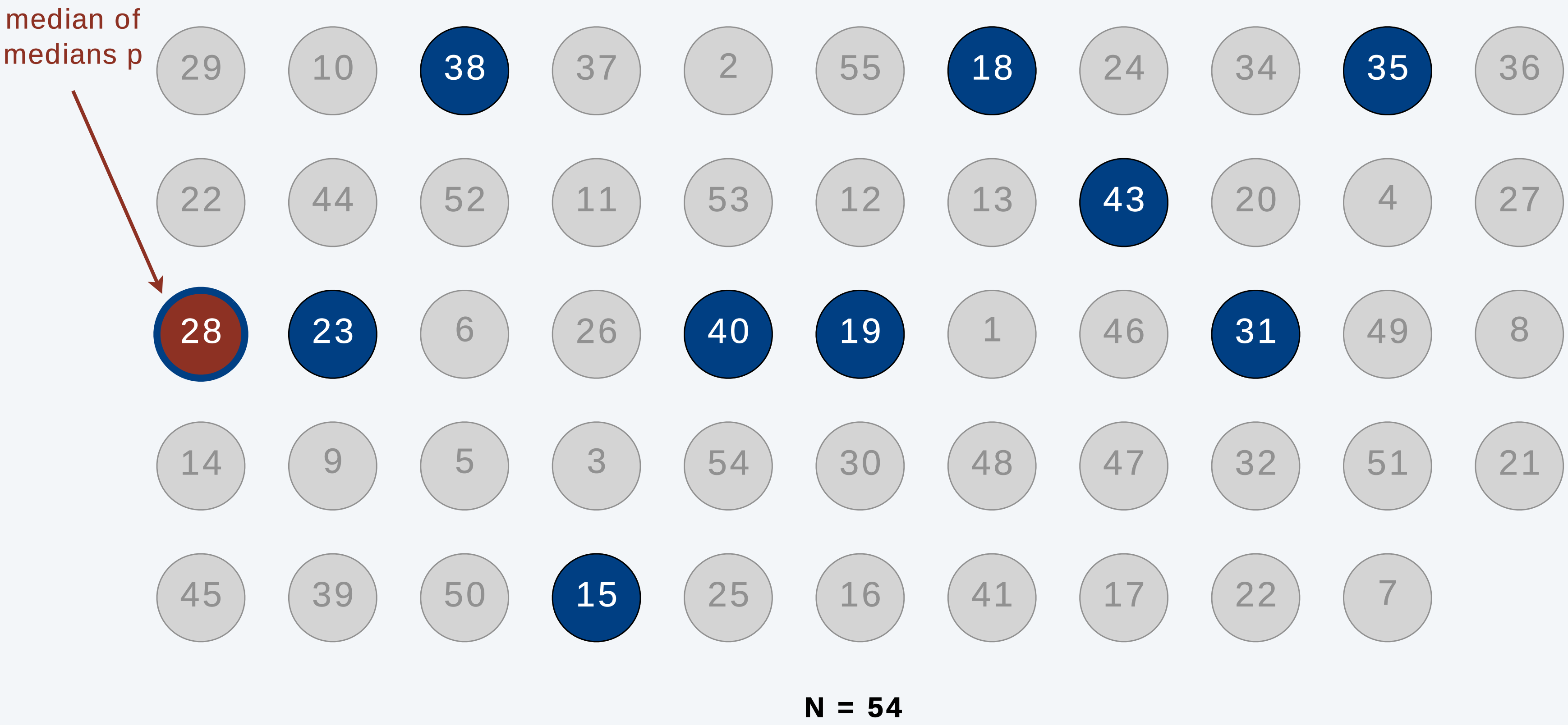
ELSE RETURN p.

$T\left(\frac{7n}{10}\right)$

ادامه کار شبیه به الگوریتم QuickSelect

Analysis of median-of-medians selection algorithm

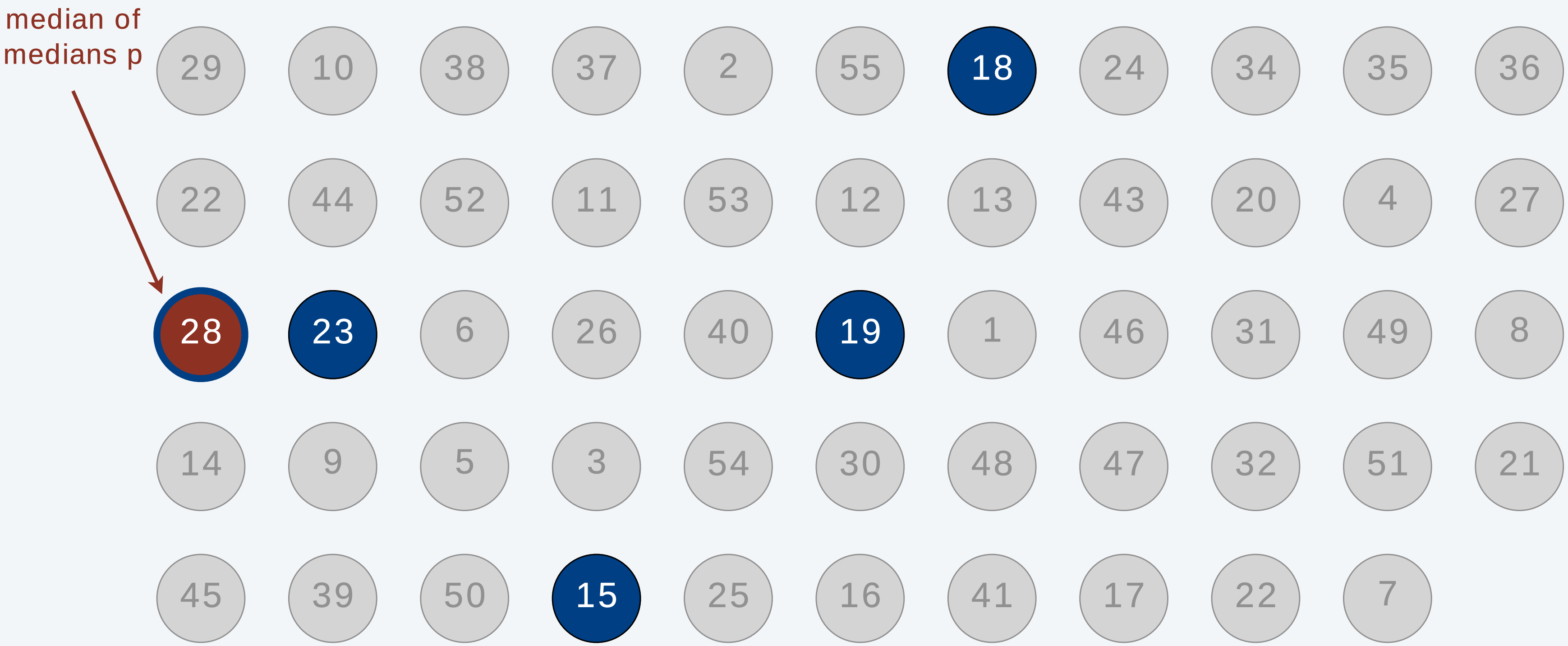
- At least half of 5-element medians $\leq$ p.



Analysis of median-of-medians selection algorithm

- At least half of 5-element medians $\leq\ p$.
- At least $\lfloor \lfloor n/5 \rfloor / 2 \rfloor = \lfloor n/10 \rfloor$ medians $\leq\ p$.

دست کم نیمی از گروه های پنج تایی
میانه ای دارند که کوچکتر یا مساوی p است.

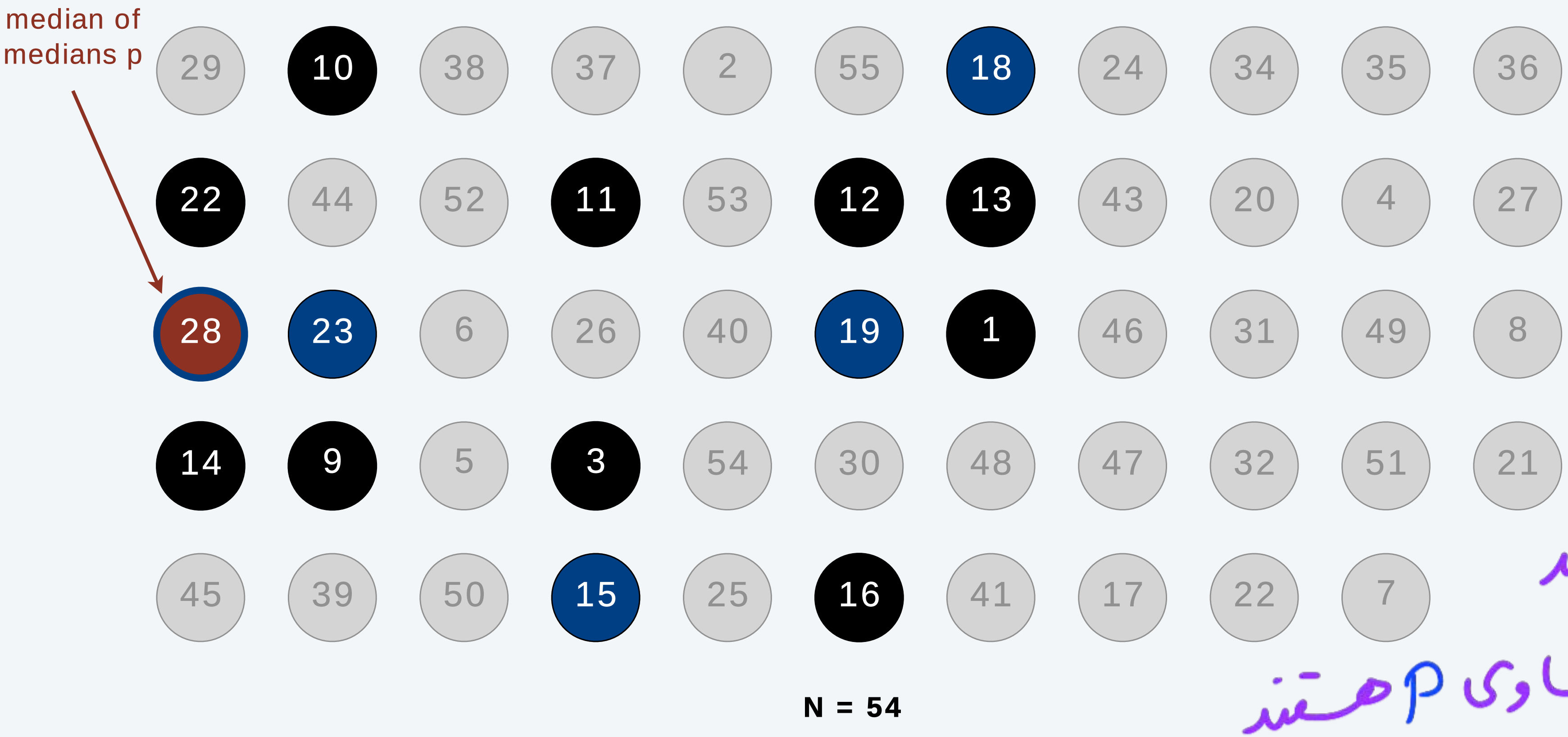


N = 54

Analysis of median-of-medians selection algorithm

در هر یک از
آن گروه های پنج تایی که میانه‌ای
کوچکتر یا مساوی P دارند، ۳ عدد
وجود دارد که کوچکتر یا مساوی میانهٔ همان گروه است (با احتساب خود میانه آن گروه)

- At least half of 5-element medians $\leq p$.
- At least $\lfloor \lfloor n/5 \rfloor / 2 \rfloor = \lfloor n/10 \rfloor$ medians $\leq p$.
- At least $3 \lfloor n/10 \rfloor$ elements $\leq p$.



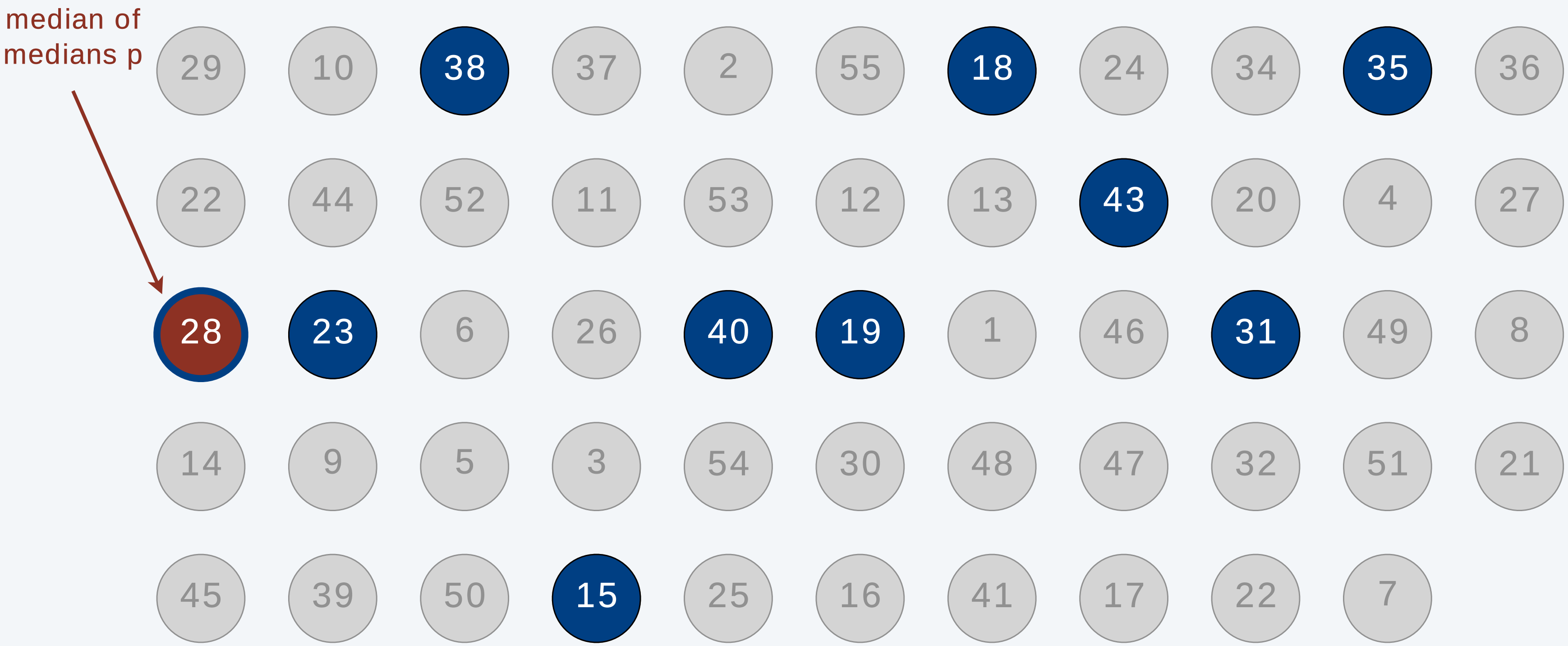
و در نتیجه
$3 \lfloor n/10 \rfloor$
عدد دست کم
در کل وجود دارند
که کوچکتر یا مساوی P هستند

Analysis of median-of-medians selection algorithm

- At least half of 5-element medians ≥ p.

مشابه استدلال‌های فوق

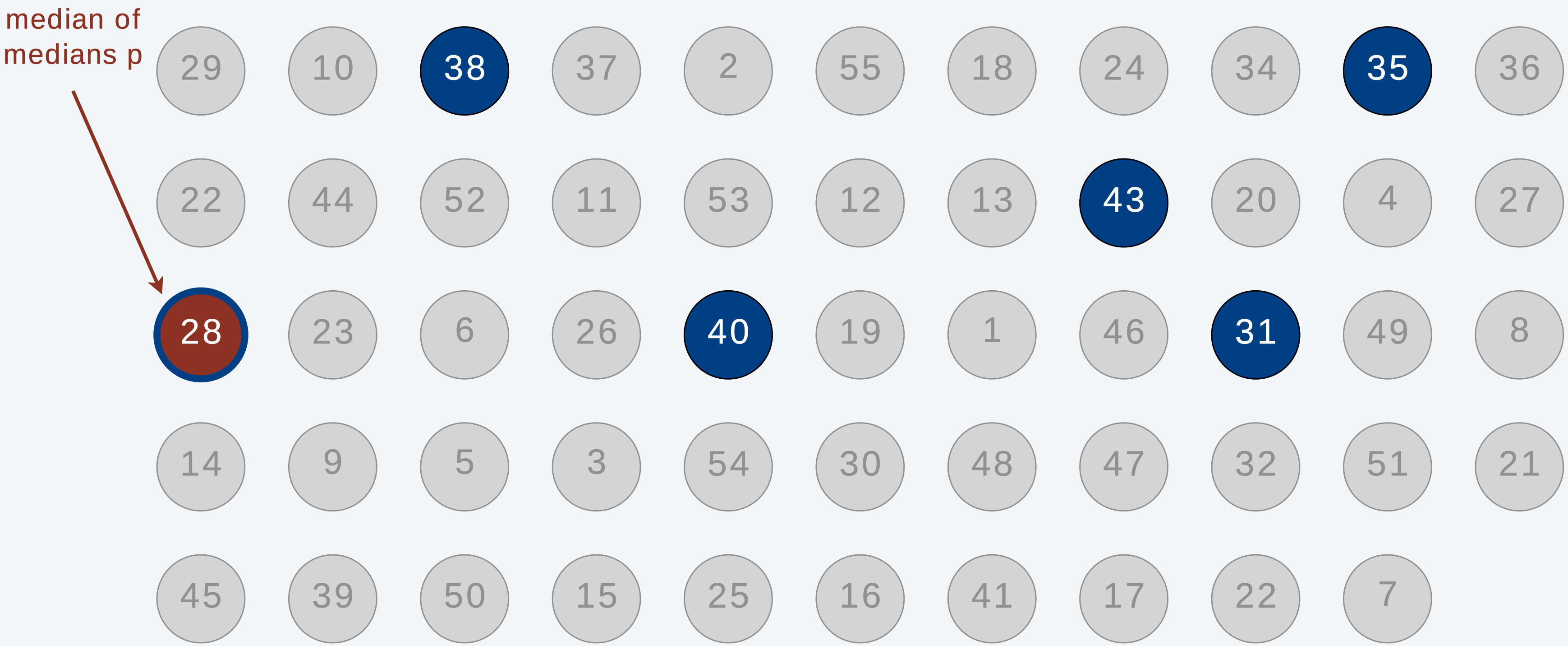
را می‌توان برای گروه‌هایی که میانه‌ای بزرگتر یا مساوی P دارند آورد



N = 54

Analysis of median-of-medians selection algorithm

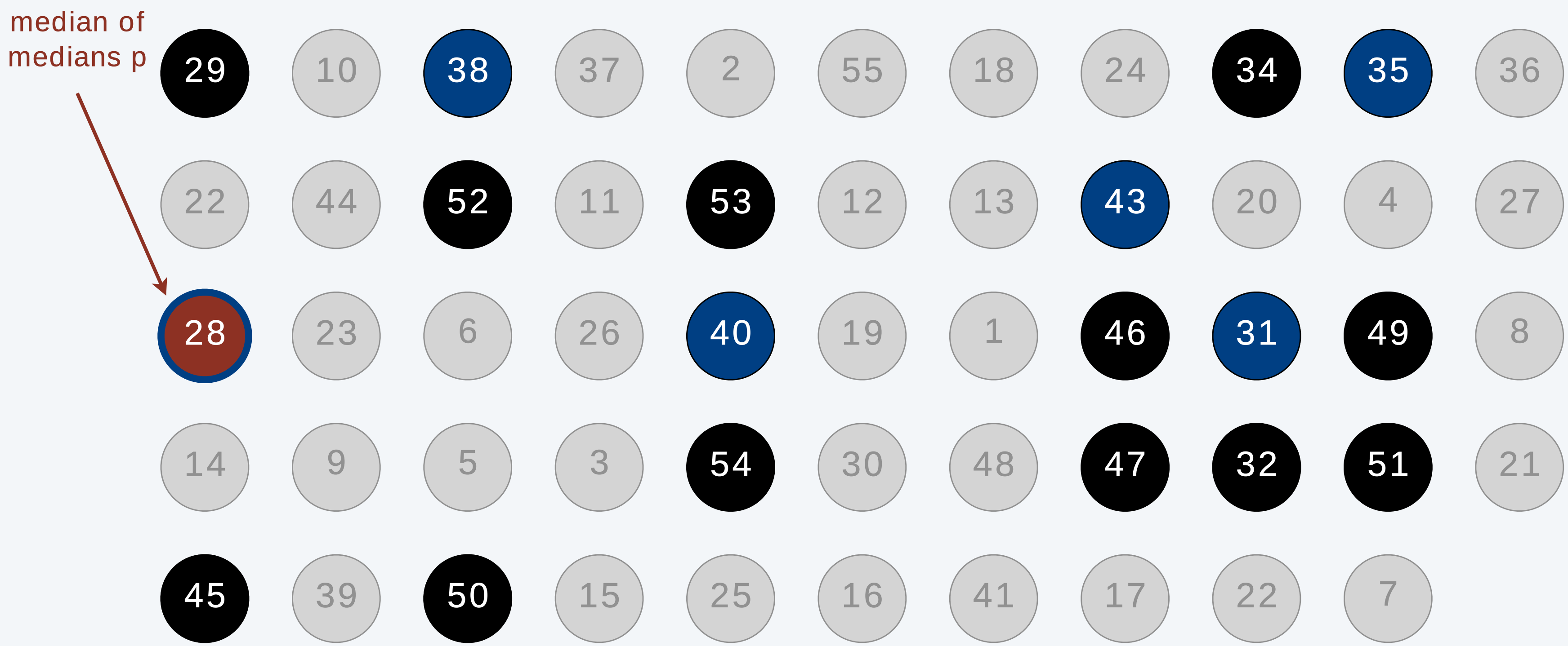
- At least half of 5-element medians ≥ p.
- Symmetrically, at least $\lfloor n / 10 \rfloor$ medians ≥ p.



N = 54

Analysis of median-of-medians selection algorithm

- At least half of 5-element medians $\geq p$.
- Symmetrically, at least $\lfloor n / 10 \rfloor$ medians $\geq p$.
- At least $3 \lfloor n / 10 \rfloor$ elements $\geq p$.



N = 54

Median-of-medians selection algorithm recurrence

درسته

Median-of-medians selection algorithm recurrence.

- Select called recursively with $\lfloor n / 5 \rfloor$ elements to compute MOM p.
- At least $3 \lfloor n / 10 \rfloor$ elements $\leq p$.
- At least $3 \lfloor n / 10 \rfloor$ elements $\geq p$.
- Select called recursively with at most $n - 3 \lfloor n / 10 \rfloor$ elements.

Def. $C(n)$ = max # compares on an array of n elements.

$$C(n) \leq \underbrace{C(\lfloor n/5 \rfloor)}_{\text{median of medians}} + \underbrace{C(n - 3\lfloor n/10 \rfloor)}_{\text{recursive select}} + \tfrac{11}{5}\, n$$

median of medians

recursive select

computing median of 5 (6 compares per group)

partitioning (n compares)

Now, solve recurrence.

- Assume n is both a power of 5 and a power of 10?
- Assume $C(n)$ is monotone nondecreasing?

Median-of-medians selection algorithm recurrence

Analysis of selection algorithm recurrence.

- $T(n)$ = max # compares on an array of $\leq n$ elements.
- $T(n)$ is monotone, but $C(n)$ is not!

$$T(n) \leq \begin{cases} 6n & \text{if } n < 50 \\ T(\lfloor n/5 \rfloor) + T(n - 3\lfloor n/10 \rfloor) + \tfrac{11}{5}\, n & \text{otherwise} \end{cases}$$

Claim. $T(n) \leq 44\, n$.

- Base case: $T(n) \leq 6\, n$ for $n < 50$ (mergesort).
- Inductive hypothesis: assume true for $1, 2, \ldots, n-1$.
- Induction step: for $n \geq 50$, we have:

$$\begin{aligned} T(n) &\leq T(\lfloor n/5 \rfloor) + T(n - 3\lfloor n/10 \rfloor) + 11/5\, n \\ &\leq 44\,(\lfloor n/5 \rfloor) + 44\,(n - 3\lfloor n/10 \rfloor) + 11/5\, n \\ &\leq 44\,(n/5) + 44\, n - 44\,(n/4) + 11/5\, n \quad \longleftarrow \quad \text{for } n \geq 50,\ 3\lfloor n/10 \rfloor \geq n/4 \\ &= 44\, n. \quad \blacksquare \end{aligned}$$

Linear-time selection postmortem

Proposition. [Blum-Floyd-Pratt-Rivest-Tarjan 1973] There exists a compare-based selection algorithm whose worst-case running time is $O(n)$.

Time Bounds for Selection

.

Manuel Robert W. Floyd, Vaughan
Ronald and Robert E.

<u>Abstract</u>

The number of comparisons required to select the i-th smallest of n numbers is shown to be at most a linear function of n by analysis of a new selection algorithm -- PICK. Specifically, no more than n comparisons are ever required. This bound is improved for

Theory.

- Optimized version of BFPRT: $\leq 5.4305\, n$ compares.
- Best known upper bound [Dor-Zwick 1995]: $\leq 2.95\, n$ compares.
- Best known lower bound [Dor-Zwick 1999]: $\geq (2 + \varepsilon)\, n$ compares.

Linear-time selection postmortem

Proposition. [Blum-Floyd-Pratt-Rivest-Tarjan 1973] There exists a compare-based selection algorithm whose worst-case running time is $O(n)$.

Time Bounds for Selection

.

Manuel Robert W. Floyd, Vaughan

Ronald and Robert E.

Abstract

The number of comparisons required to select the i-th smallest of n numbers is shown to be at most a linear function of n by analysis of a new selection algorithm -- PICK. Specifically, no more than n comparisons are ever required. This bound is improved for

Practice. Constant and overhead (currently) too large to be useful.

Open. Practical selection algorithm whose worst-case running time is $O(n)$.