Design and Analysis of Algorithms

Median and Order statistics

Reference:
CLRS Chapter 9

Topics:

- **Order statistics**
- **Expected linear time selection**
- **Worst-case linear time selection**

Today

- Median and order statistics
- Two $O(n)$ time algorithms:
  - Randomized: similar to Quicksort
  - Deterministic: quite tricky
- Both are examples of divide and conquer

Order Statistics

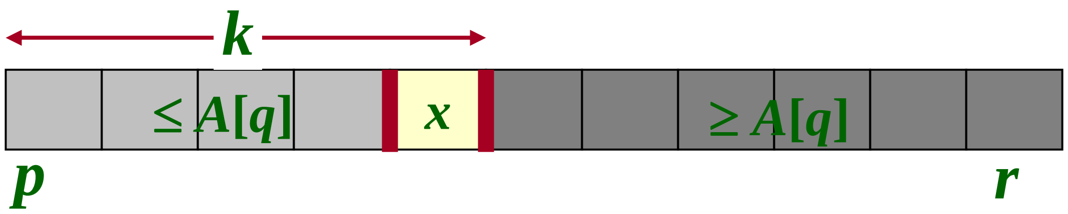
- Select the ith smallest of n elements (the element with rank i).
  - Minimum: $i = 1$
  - Maximum: $i = n$
  - Median: $i = \lfloor (n+1) / 2 \rfloor$ or $\lceil (n+1)/2 \rceil$

- How fast can we solve the problem?
  - $O(n)$ for min or max.
  - $O(n \lg n)$ by sorting for general i.
  - $O(n \lg i)$ with heaps.

- Next we will see how to do it in $O(n)$ time.

Randomized algorithm for finding the *i*th element

- Divide and conquer approach
- Main idea: PARTITION

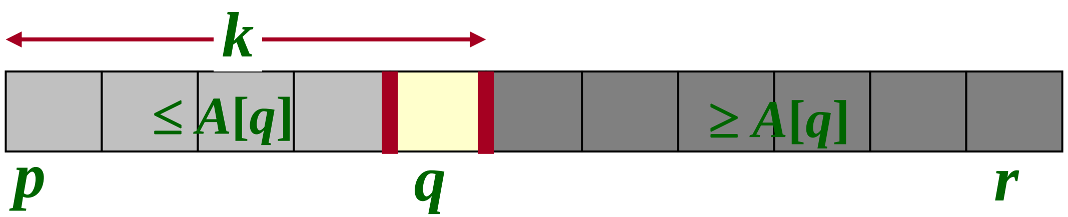


- If $i < k$, recurse on the left
- If $i > k$. recurse on the right
- Otherwise, output x

RANDOMIZD Divide-and-conquer

RANDOMIZED SELECT

```
RANDOMIZED-SELECT(A, p, r, i)
if p = r
    then return A[p]
q ← RANDOMIZED-PARTITION(A, p, r)    // as used in quicksort
k ← q–p+1                             // k = rank(A[q])
if i = k                              // the pivot value is the answer
    then return A[q]
if i < k
    then return RANDOMIZED-SELECT(A, p, q–1, i)
    else return RANDOMIZED-SELECT(A, q+1, r, i–k)
```



Example

- Select the *i* = 7th smallest:

6	10	13	5	8	3	2	11

pivot

$i = 7$

- Partition:

2	5	3	6	8	13	10	11

$k = 4$

Select the 7- 4 = 3rd smallest recursively

Analysis

- What is the worst-case running time?

  Unlucky:

  $$T(n) = T(n-1) + \Theta(n) = \Theta(n^2)$$

- Recall that a lucky partition splits into arrays with size ratio at most $9:1$
- What if all partitions are lucky?

  lucky:

  $$T(n) = T(9n/10) + \Theta(n) = \Theta(n)$$

  $$n^{\log_{10/9} 1} = n^0 = 1$$

  CASE 3

Expected running time

- The probability that a random pivot induces lucky partition is at least 8/10
- Let t_i be the number of partitions performed between the (i–1)-th and the i-th lucky partition
- The total time is at most…

$$T = t_1 n + t_2(9/10)n + t_3(9/10)^2 n + \ldots$$

- The total expected time is at most:
- $E[T] = E[t_1]\, n + E[t_2]\, (9/10)n + E[t_3]\, (9/10)^2 n + \ldots$

$= 10/8 * n * [1 + p + p^2 \ldots]$ Geometric series $p = (9/10) < 1$

$= 10/8 * n * [1/(1-p)]$

$= O(n)$

Digression: 9 to 1

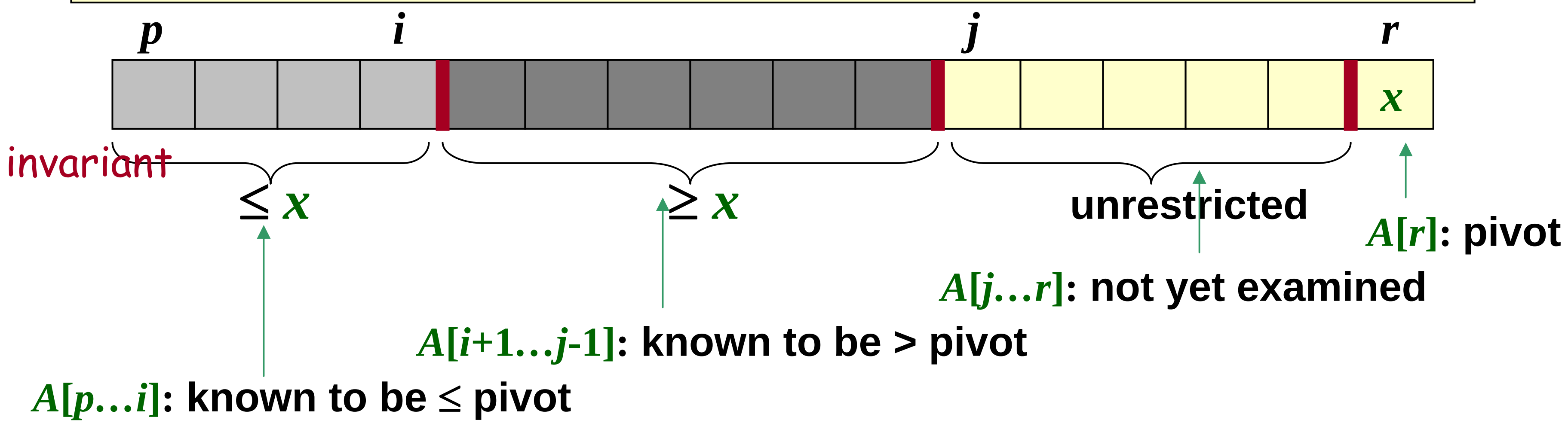
- Do we need to define the lucky partition as $9:1$ balanced?
- No. Suffices to say that both sides have size $\geq \alpha n$, for $0 < \alpha < ½$
- Need constant fraction of n, regardless of what the fraction is. But not constant number.
- Probability of getting a lucky partition is $1–2\alpha$

Partitioning subroutine

PARTITION

```
PARTITION(A, p, r)        //A[p..r]
  x ← A[r]                //the rightmost element as pivot
  i ← p-1
for j ← p to r-1
      do if A[j] ≤ x
            then i ← i+1
                 exchange A[i]↔A[j]
exchange A[i+1]↔A[r]
return i+1
```

Running time = O(n) for n elements



Summary of randomized order-statistics selection

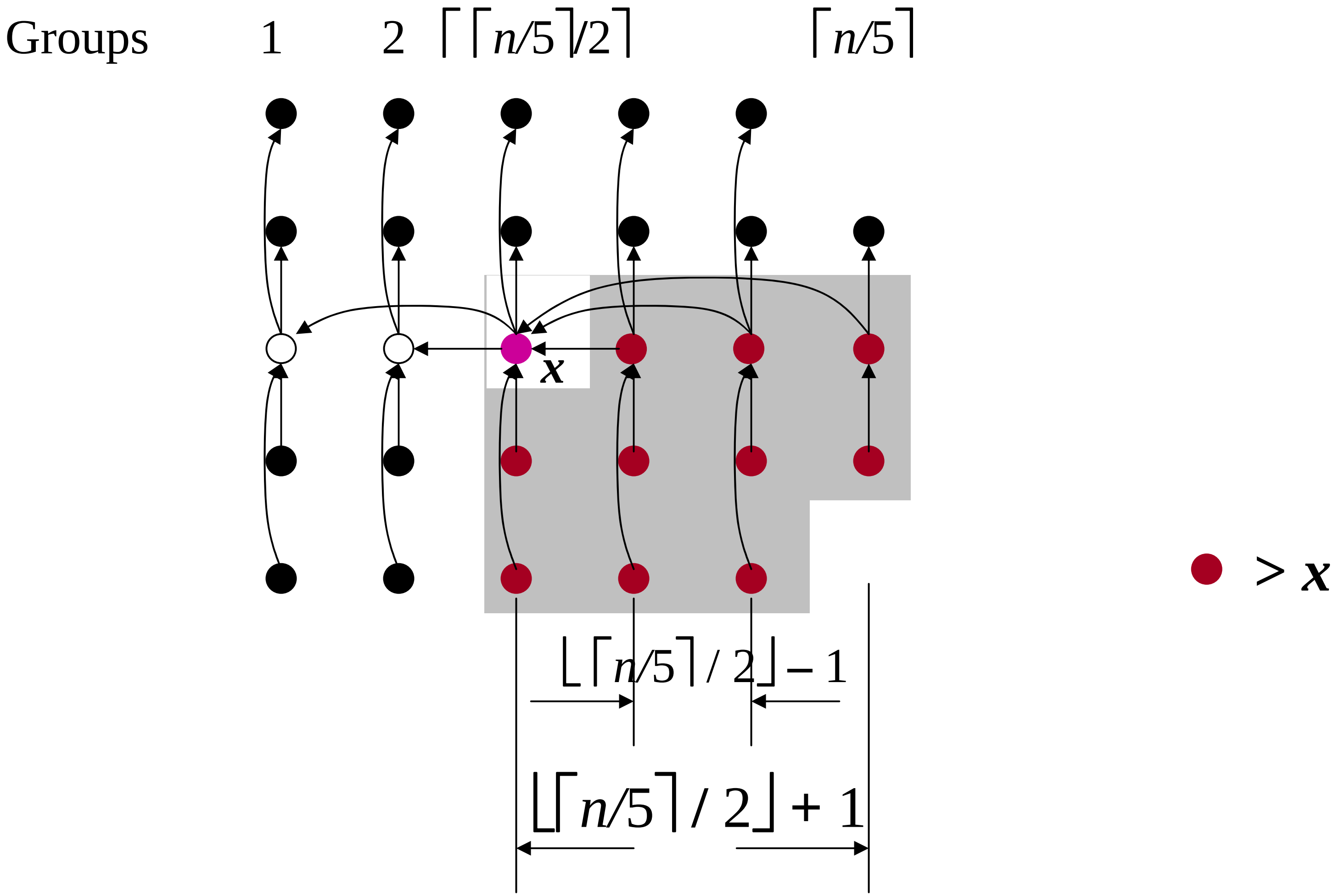
- Works fast: linear expected time.
- Excellent algorithm in practice.
- But, the worst case is very bad: $\Theta(n^2)$.

Q. Is there an algorithm that runs in linear time in the worst case?

A. Yes, due to [Blum-Floyd-Pratt-Rivest-Tarjan'1973]

- IDEA: Generate a good pivot recursively.

Pictorial Analysis of Select



Worst-Case Linear-Time Selection

SELECT(A, i)

1. Divide n elements into $\lfloor n/5 \rfloor$ groups of 5, plus extra
2. Find medians of each group by insertion sort.
3. Find median x of the $\lfloor n/5 \rfloor$ medians by calling SELECT() recursively
4. Partition the n elements around pivot x. Let k = rank(x)
5. **if** ($i = k$) **then** return x

   if ($i < k$) **then** call SELECT() recursively to find i-th smallest element in left partition

   else call SELECT() recursively to find (i-k)-th smallest element in right partition

Partition

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

29	10	38	37	02	55	18	24	34	35	36
22	44	52	11	53	12	13	43	20	04	27
28	23	06	26	40	19	01	46	31	49	08
14	09	05	03	54	30	48	47	32	51	21
45	39	50	15	25	16	41	17	33	07	

$n = 54$

Partition

- **`Partition()`.**
  - Divide n elements into $\lfloor n/5 \rfloor$ groups of 5 elements each, plus extra.
  - Find medians of each group by insertion sort.

14	09	05	03	02	12	01	17	20	04	08
22	10	06	11	25	16	13	24	31	07	21
28	23	38	15	40	19	18	43	32	35	27
29	39	50	26	53	30	41	46	33	49	36
45	44	52	37	54	53	48	47	34	51	

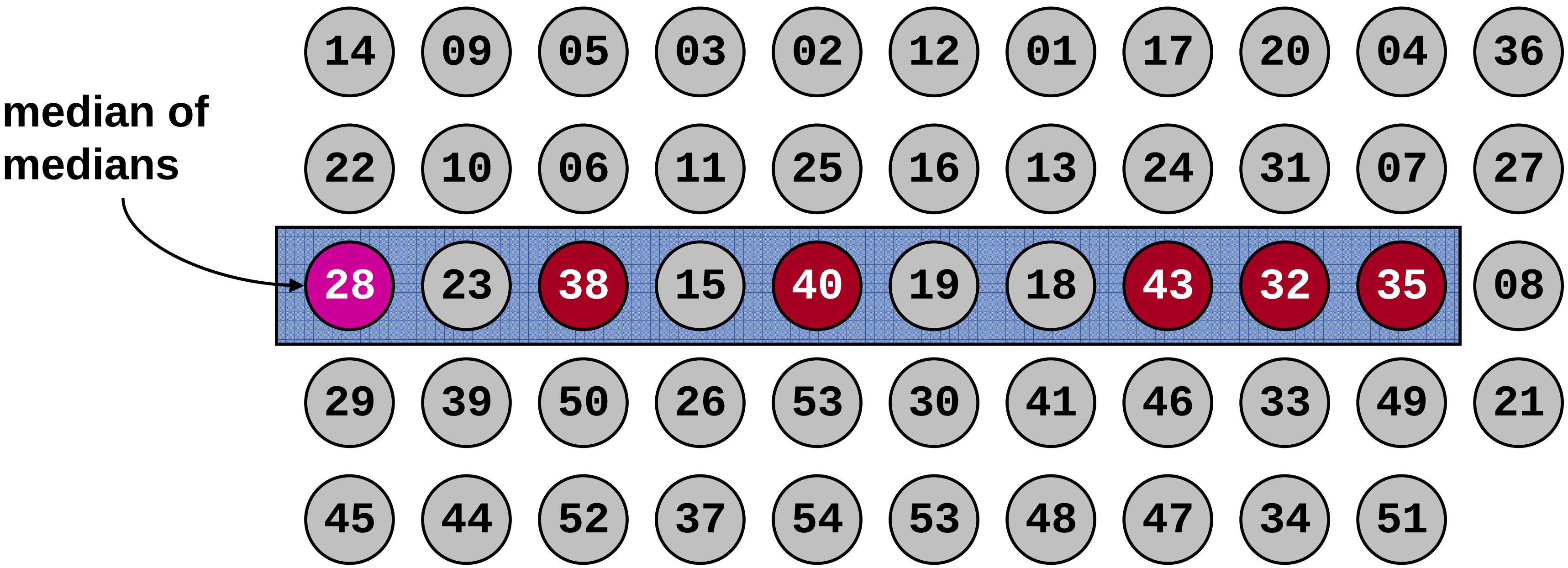
Partition

- **`Partition()`.**
  - **Divide n elements into $\lfloor n/5 \rfloor$ groups of 5 elements each, plus extra.**
  - **Find medians of each group by insertion sort.**
  - **Find x = "median of medians" by `SELECT()` on $\lfloor n/5 \rfloor$ medians.**

14	09	05	03	02	12	01	17	20	04	08
22	10	06	11	25	16	13	24	31	07	21
28	23	38	15	40	19	18	43	32	35	27
29	39	50	26	53	30	41	46	33	49	36
45	44	52	37	54	53	48	47	34	51	

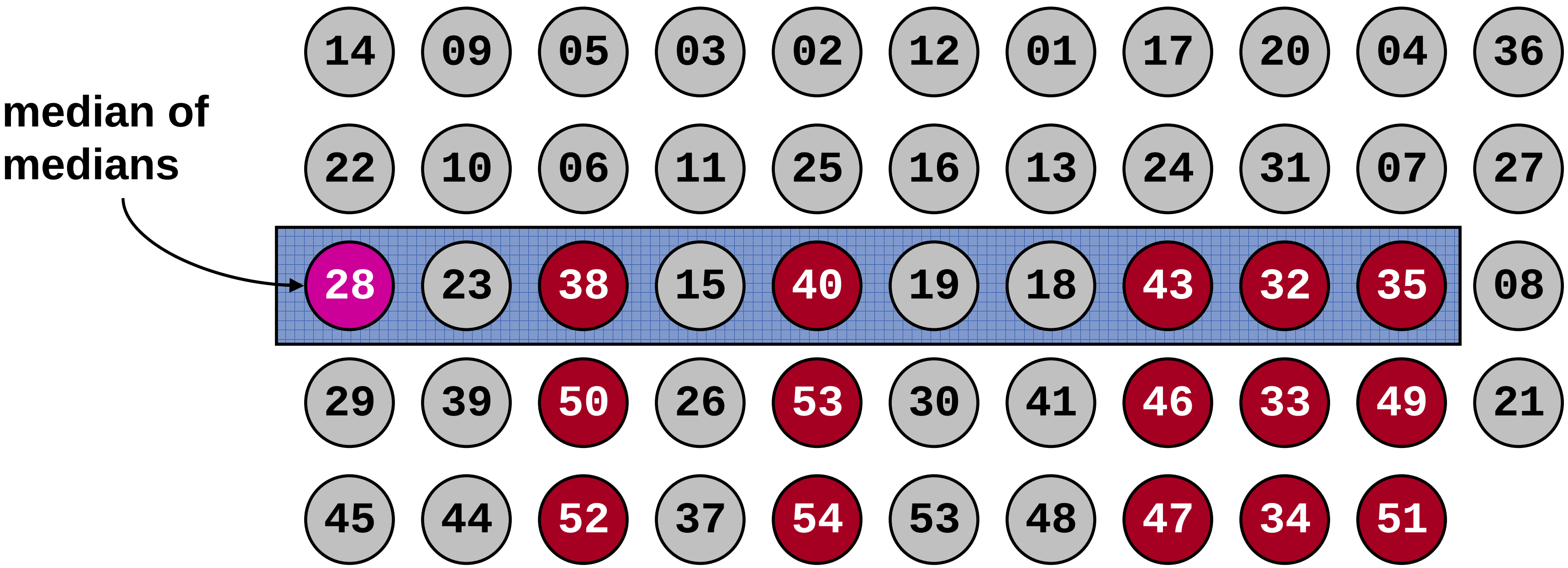
Selection Analysis

- **Crux of proof:**
  - At least half of medians found in step2 $> x$, except for…
    - » at least $\lceil \lceil n/5 \rceil /2 \rceil - 2 = \lceil n/10 \rceil -2$ medians $> x$



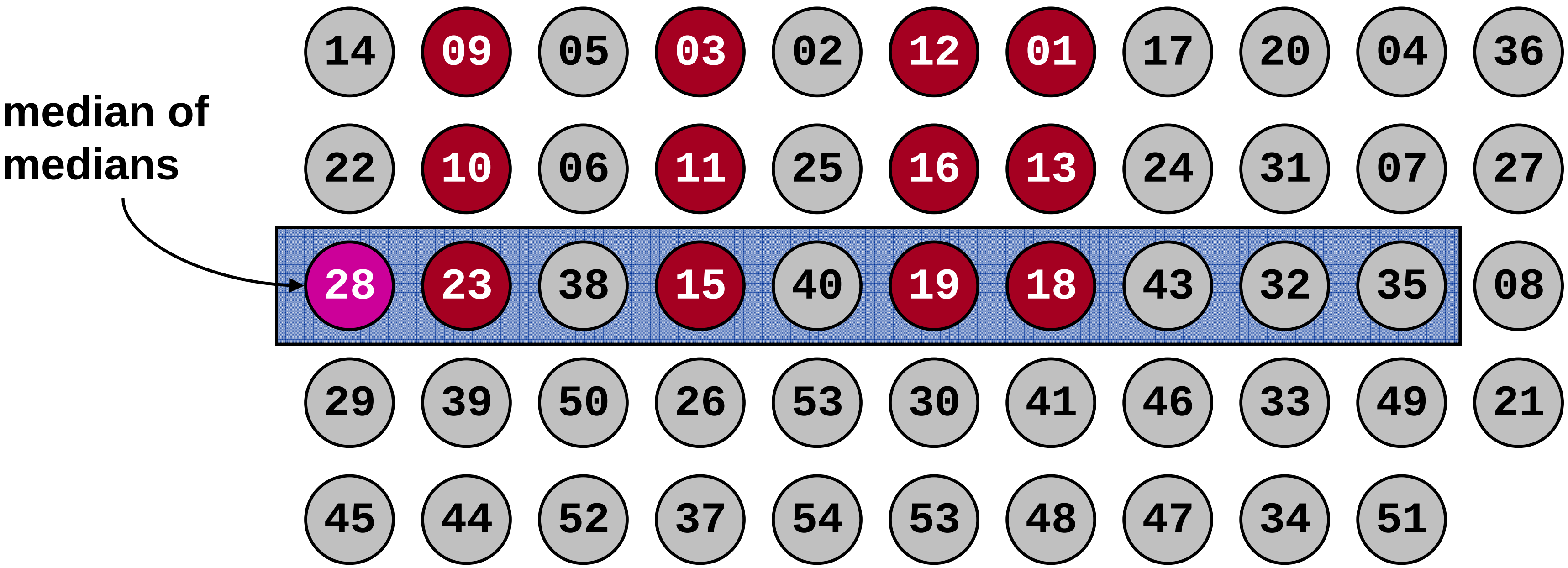
Selection Analysis

- Crux of proof:
  - At least half of medians found in step2 $> x$, except for…
    - at least $\lceil \lceil n/5 \rceil /2 \rceil - 2 = \lceil n/10 \rceil - 2$ medians $> x$
  - At least $3(\lceil n/10 \rceil - 2)$ elements $> x$.



Selection Analysis

- **Crux of proof:**
  - At least half of medians found in step2 $> x$, except for…
    - » at least $\lceil \lceil n / 5 \rceil / 2 \rceil - 2 = \lceil n/10 \rceil - 2$ medians $> x$
  - At least $3(\lceil n/10 \rceil - 2)$ elements $> x$.
  - At least $3(\lceil n/10 \rceil - 2)$ elements $< x$.



Selection Analysis

- **Crux of proof:**
  - At least half of medians found in step2 $> x$, except for…
    - » at least $\lceil \lceil n/5 \rceil /2 \rceil - 2 = \lceil n/10 \rceil - 2$ medians $> x$
  - At least $3(\lceil n/10 \rceil -2) \geq 3n/10 - 6$ elements $> x$.
  - At least $3n/10 - 6$ elements $< x$.

    $\Rightarrow$ SELECT( ) called recursively (step 5) with at most $n - (3n/10 - 6) = 7n/10 + 6$ elements in the worst case, regardless of which partition is used
- **Algorithm analysis**
  - Step 1,2, and 4 take $O(n)$ time,
  - Step 3 takes time $T(\lceil n/5 \rceil)$, and
  - Step 5 takes time at most $T(7n/10 + 6)$

Selection Analysis

- Claim: Thus after partitioning around x in step 4, step 5 will call SELECT() on at most 70% of the elements

$$T(n) \le \begin{cases} \Theta(1) & \text{if } n \le 140 \\ T(n/5) + T(7n/10 + 6) + O(n) & \text{if } n > 140 \end{cases}$$

- Claim: $T(n) \le cn$.
  - Base case: $n \le 140$.
  - Inductive hypothesis: assume true for $1, 2, \ldots, n\text{-}1$.
  - Induction step: for $n > 140$, we have:

$$\begin{aligned} T(n) &\le c\, n/5 + c\,(7n/10 + 6) + n \\ &\le cn/5 + c + 7cn/10 + 6c + n \\ &\le 9\,c\,n/10 + n + 7\,c \\ &\le c\,n \quad \Leftarrow \text{if we choose } c \text{ such that } 7c + n/10 \le cn/10 \\ &\qquad\qquad \text{e.g. true for } c = 20 \text{ if } n > 140 \end{aligned}$$

Linear-Time Median Selection: Applications

- Given a "black box" $O(n)$ algorithm that finds median, what can we do?
- i-th order statistic:
  - Find median x
  - Partition input around x
  - if ($i \le (n+1)/2$) recursively find i-th element of first half
  - else find $(i - (n+1)/2)$-th element in second half
  - $T(n) = T(n/2) + O(n) = O(n)$
- Worst-case $O(n \lg n)$ quicksort
  - Find median x and partition around it
  - Recursively quicksort two halves
  - $T(n) = 2\,T(n/2) + O(n) = O(n \lg n)$

Searching

Two fundamental problems in CS:

- Sorting
- Searching, defined as follows

(1)SELECTION PROBLEM

Input: Set *A* and index *i*

Output: *i*th smallest (or largest) element in *A*

(2)DICTIONARY PROBLEM

Input: Set *A* and key *k*

Output: element in *A* with the key if found,
and "not found" otherwise.

Min/Max problem

- **Find the minimum element in a set of *n* elements**

MINMUM(A)

```
MINIMUM(A)
1  min ← A[1]
2  for i ← 2 to length[A]
3       do if min > A[i]
4               then min ← A[i]
5  return min
```

- **Algorithm finds the maximum/ minimum in $n-1$ comparisons, which is optimal w.r.t. the number of comparisons**

Min-Max problem

- **Find the maximum and minimum elements in a set of *n* elements**

MAXIMUM-MINIMUM(A)

```
MAXIMUM-MINIMUM(A)
max ← min ← A[1]
for i ← 2 to length[A]
    do if A[i] > max
          then max ← A[i]
          else if A[i] < min
                  then min ← A[i]
return min & max
```

- **Algorithm finds the maximum and minimum in $n-1$ comparisons, if the array is already sorted in increasing order and in $2n$-2 comparisons in decreasing order.**

Min-Max problem

Search	Min	Max	Min & Max
#comparisons	$n-1$	$n-1$	$2n-2$

Algorithm1: Find maximum and then find minimum $2n-2$ **comparisons**

Algorithm2: a) Pair elements up $\lceil n/2 \rceil$ pairs

$$(a_1, a_2), (a_3, a_4), \ldots, (a_{n-1}, a_n)$$

b) Reorder: For $k = 1, \ldots, \lceil n/2 \rceil$ $\lceil n/2 \rceil$ comparisons

$$(a'_{2k-1}, a'_{2k}) = \begin{cases} (a_{2k-1}, a_{2k}) & \text{if } a_{2k-1} < a_{2k} \\ (a_{2k}, a_{2k-1}) & \text{otherwise} \end{cases}$$

c) For $k = 1, \ldots, \lceil n/2 \rceil$ $\lceil n/2 \rceil + \lceil n/2 \rceil$ comparisons

Compare a'_{2k-1} **with current min, and** a'_{2k} **with current max**

Algorithm2 finds the maximum and minimum in $3\lceil n/2 \rceil$

Min and Max Example

- Method 3: Divide and Conquer .
- Find the min and max of {3, 5, 6, 2, 4, 9, 3, 1}

- Example
  - $A = \{3,5,6,2\}$ and $B = \{4,9,3,1\}$
  - $\min(A) = 2$, $\min(B) = 1$
  - $\max(A) = 6$, $\max(B) = 9$

- $\min\{\min(A),\min(B)\} = 1$
- $\max\{\max(A), \max(B)\} = 9$

Min-Max problem

MAX & MIN

```
MAXMIN(i,j,fmax,fmin)
if (i=j)
    then fmax ← fmin ← a[i]
if (i=(j-1)) then
    if a[i]<a[j]
        then fmax ← a[j]
             fmin ← a[i]
        else fmax ← a[i]
             fmin ← a[j]
else
        mid ← ⌊(i+j)/2⌋
        MAXMIN(i,mid,gmax,gmin)
        MAXMIN(mid+1,j,hmax,hmin)
        fmax ← max{gmax,hmax}
        fmin ← min{gmin,hmin}
```

Time Complexity

- **Let $T(n)$ be the number of comparisons made when finding the min and max of n elements**

$$T(n) = \begin{cases} 0 & n = 1 \\ 1 & n = 2 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 2 & n > 2 \end{cases}$$

- **To solve the recurrence, assume n is a power of 2 and use repeated substitution.**
- $T(n) = \lceil 3n/2 \rceil - 2$

Average-Case Analysis

- Define indicator random variable X_k

$$X_k = \mathrm{I}\{\text{the subarray } A[p..q] \text{ has exactly } k \text{ elements}\}$$

$$\mathrm{E}[X_k] = 1/n$$

- $T(n) \le 1/n(\ T(\max(1, n-1)) + \sum_{k=1 \text{ to } n-1} T(\max(k, n-k))\ ) + O(n)$

$$\le 1/n(\ T(n-1) + 2 \sum_{k=\lceil n/2 \rceil \text{ to } n-1} T(k)\ ) + O(n)$$

$$= 2/n \sum_{k=\lceil n/2 \rceil \text{ to } n-1} T(k) + O(n)$$

- Substitution Method: Guess $T(n) \le c\ n$

$$T(n) \le 2/n \sum_{k=\lceil n/2 \rceil \text{ to } n-1} ck + O(n)$$

$$\le 2c/n\ (\ \sum_{k=1 \text{ to } n-1} k - \sum_{k=1 \text{ to } \lceil n/2 \rceil -1} k\ ) + O(n)$$

$$= 2c/n\ ((n-1)n/2 - 1/2(\lceil n/2 \rceil -1)\lceil n/2 \rceil) + O(n)$$

$$\le c(n-1) - (c/n)(n/2 - 1)(n/2) + O(n)$$

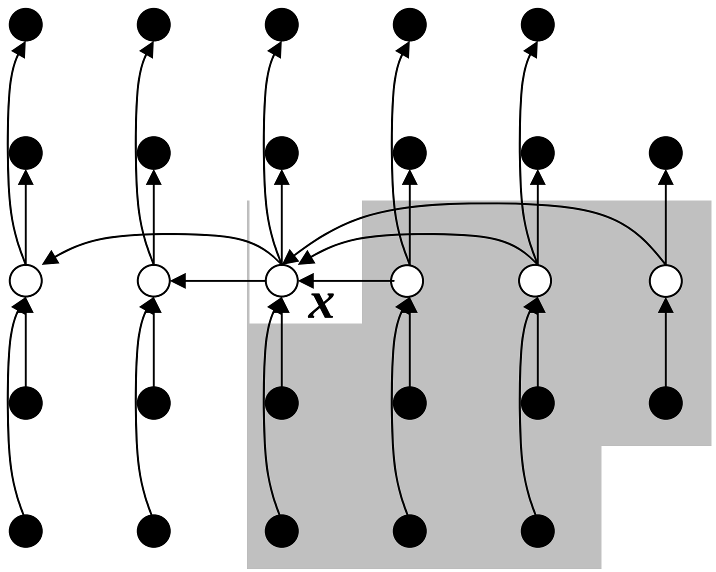
$$\le c(3n/4 - 1/2) + O(n)$$

$\le cn$ $\Leftarrow$ if we pick c large enough so that $c(n/4 + 1/2)$ dominates $O(n)$

Worst-Case Linear-Time Selection

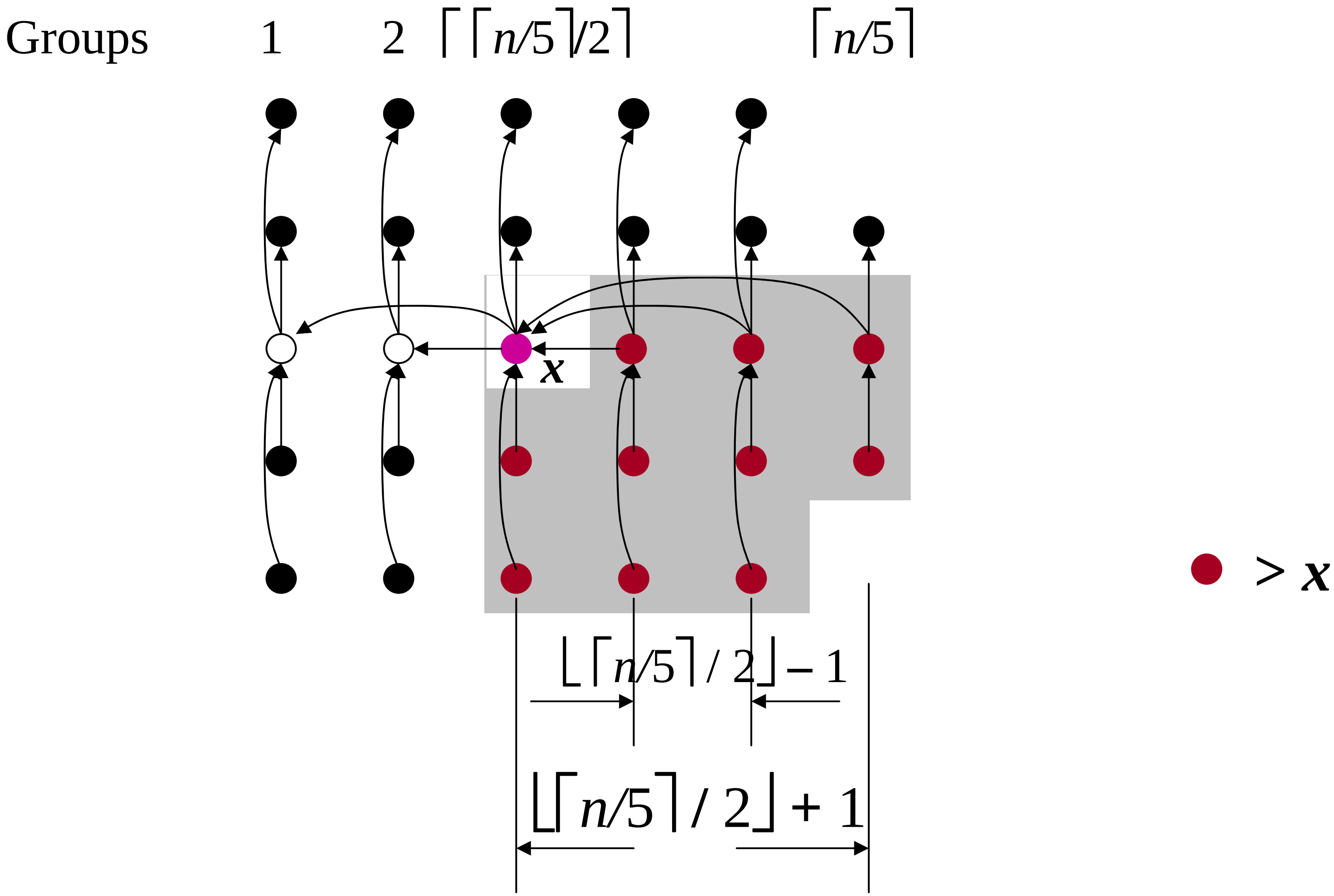
- Randomized algorithm works well in practice
- What follows is a worst-case linear time algorithm, really of theoretical interest only
- Basic idea:
  - Generate a good partitioning element
  - Call this element x

Pictorial Analysis of Select



- **Fig. 9.1 Analysis of the algorithm SELECT. The *n* elements are represented by small circles, and each group occupies a column. The medians of the group are whitened, and the median-of-medians *x* is labeled.**

Pictorial Analysis of Select



Worst-Case Linear-Time Selection

SELECT(i, n)

1. Divide the n elements into groups of 5. Find the median of each 5-element group by hand.
2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.
3. Partition around pivot x. Let k = rank(x)
4. if $i = k$ then return x
   else if $i < k$
   then recursively SELECT the ith smallest element in lower part
   else recursively SELECT the $(i-k)$th smallest element in right part

Same as RAND-SELECT

Choosing the pivot

– Divide n elements into $\lfloor n/5 \rfloor$ groups of 5 elements each, plus extra.

29	10	38	37	02	55	18	24	34	35	36
22	44	52	11	53	12	13	43	20	04	27
28	23	06	26	40	19	01	46	31	49	08
14	09	05	03	54	30	48	47	32	51	21
45	39	50	15	25	16	41	17	33	07	

$n = 54$

Choosing the pivot

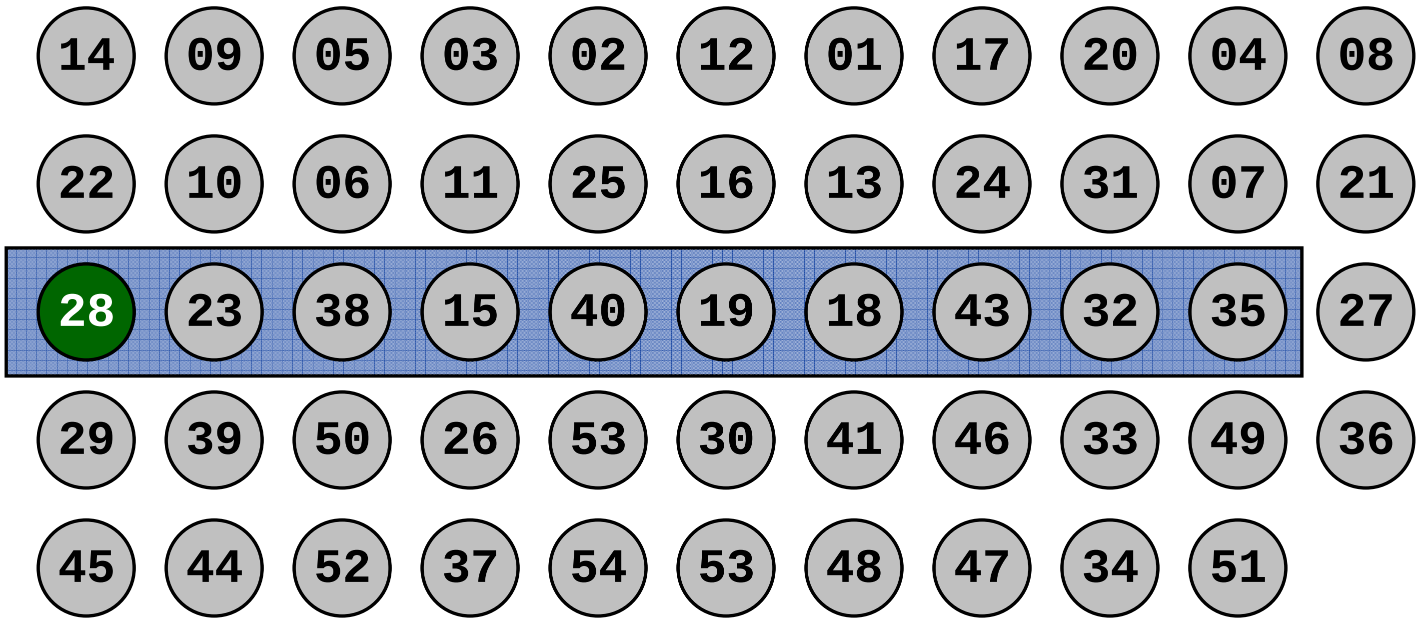
– Divide n elements into $\lfloor n/5 \rfloor$ groups of 5 elements each, plus extra.

– Find medians of each group by insertion sort.

14	09	05	03	02	12	01	17	20	04	08
22	10	06	11	25	16	13	24	31	07	21
28	23	38	15	40	19	18	43	32	35	27
29	39	50	26	53	30	41	46	33	49	36
45	44	52	37	54	53	48	47	34	51	

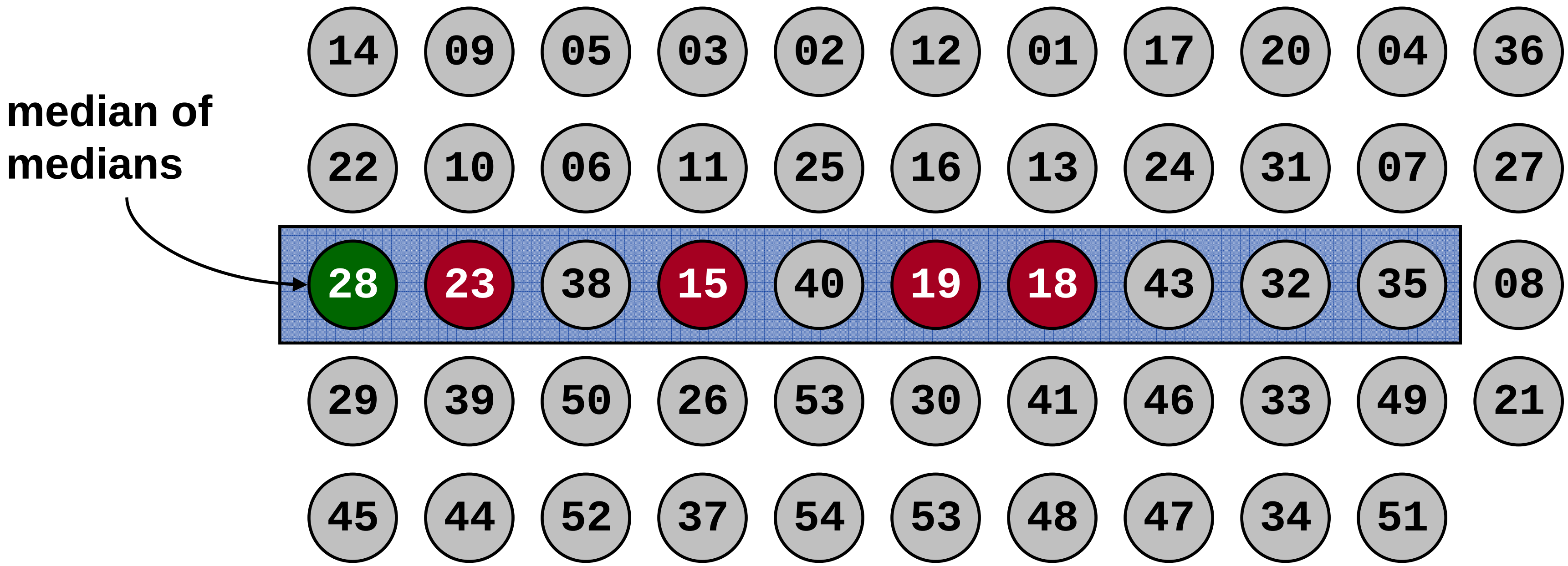
Choosing the pivot

- Divide n elements into $\lfloor n/5 \rfloor$ groups of 5 elements each, plus extra.
- Find medians of each group by insertion sort.
- Find x = "median of medians" by `SELECT()` on $\lfloor n/5 \rfloor$ medians.



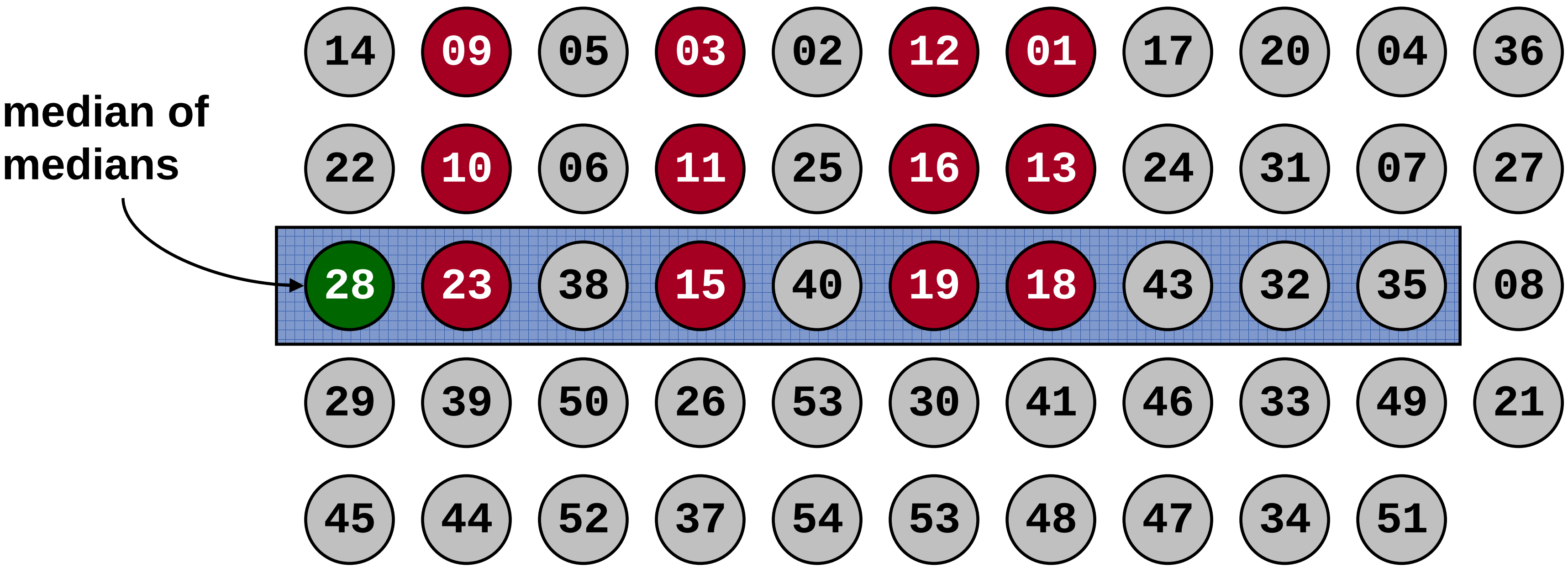
Selection Analysis

- **Crux of proof:**
  - At least **half** the group medians are $\le x$, which is at least $\lfloor \lfloor n/5 \rfloor / 2 \rfloor = \lfloor n/10 \rfloor$ group medians



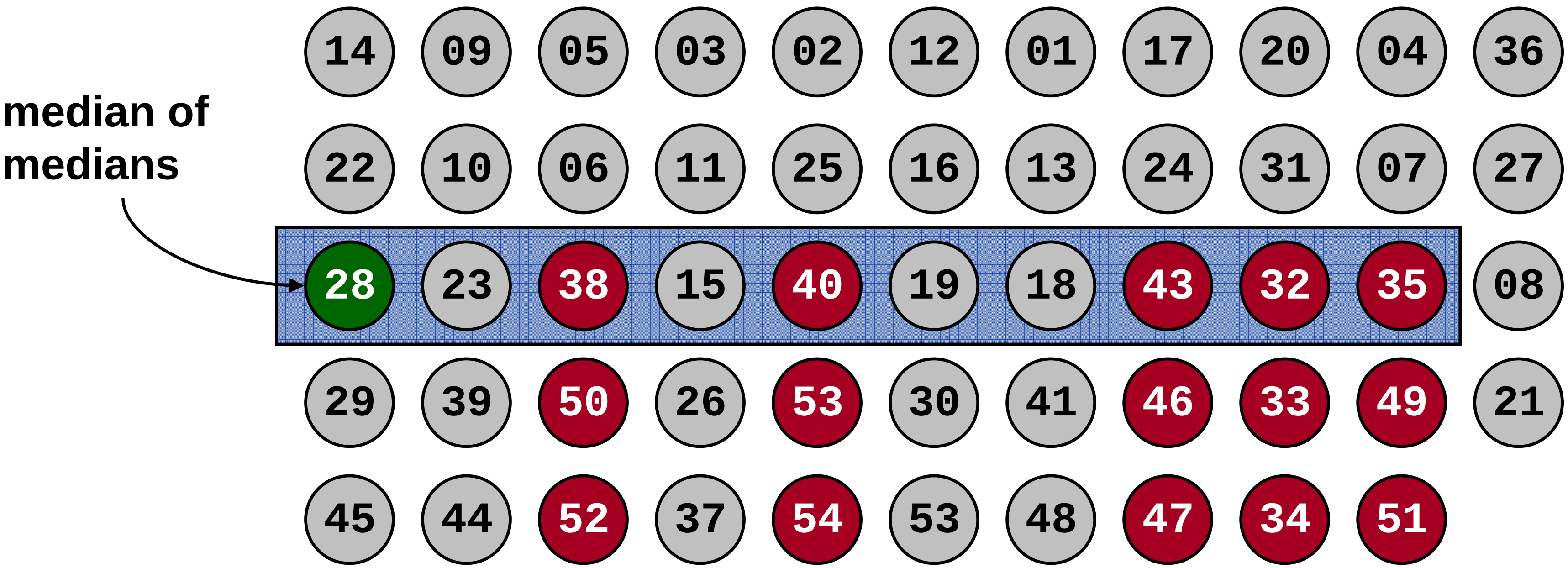
Selection Analysis

- Crux of proof:
  - At least **half** the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor / 2 \rfloor = \lfloor n/10 \rfloor$ group medians
  - Therefore, at least $3\lfloor n/10 \rfloor$ elements $\leq x$.



Selection Analysis

- **Crux of proof:**
  - At least **half** the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor / 2 \rfloor = \lfloor n/10 \rfloor$ group medians
  - Therefore, at least $3\lfloor n/10 \rfloor$ elements $\leq x$.
  - Similarly, at least $3\lfloor n/10 \rfloor$ elements $\geq x$.



Worst-Case Linear-Time Selection

$T(n)$ SELECT(i, n)

$\Theta(n)$ — 1. Divide the n elements into groups of 5. Find the median of each 5-element group by hand.

$T(n/5)$ — 2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.

$\Theta(n)$ — 3. Partition around pivot x. Let k = rank(x)

$T(7n/10)$ — 4. **if** $i = k$ **then** return x

else if $i < k$

then recursively SELECT the ith smallest element in lower part

else recursively SELECT the $(i-k)$th smallest element in right part

Solving the recurrence

$$T(n) = T(1/5\, n) + T(7/10\, n) + \Theta(n)$$

Substitution:

$T(n) \le cn$

$$T(n) \le 1/5\, cn + 7/10\, cn + \Theta(n)$$
$$= 18/20\, cn + \Theta(n)$$
$$= cn - (2/20\, cn - \Theta(n))$$
$$\le cn$$

If c is chosen large enough to handle the $\Theta(n)$.