Summary of §2 Searching

- 2. Searching
  - Linear Search
  - Neighbor Search
  - Binary Search
  - Uniqueness
  - Hashing
  - Order Statistics/ Median in linear time!
  - 1D Range Counting/Reporting
  - 2D Range Counting/Reporting

- Specification
- Primitives: semantics and cost
- Design
- Analysis
- (Optimality)

2. Searching

linear search

Specification: Fix set X.

Input: $N \in \mathbb{N}$ and finite sequence $x_1, \ldots x_N \in X$ in array $x[1 \ldots N]$
as well as $y \in X$.

Output: $n \in \{1, \ldots N\}$ such that $x_n = y$ or $n=0$ if $x_j \neq y$

Primitives: Access $\mathbb{N} \ni n \to x[n] \in X$ cost 1
Comparison "$x=y$?" cost 1
Index integer arithmetic cost 1

Algorithm:

```
for n=1…N
  if x[n]=y then return n;
return 0.
```

runtime $O(N)$

correctness: √

2. Searching *neighbor* search

Specification: Fix set X with total order $\leq$.

Input: $N \in \mathbb{N}$ and finite sequence $x_1,\ldots x_N \in X$ in array $x[1\ldots N]$
as well as $y \in X$. with $x_1 < \ldots < x_N$

Output: $n \in \{0,\ldots N\}$ such that $x_n \leq y < x_{n+1}$ where $x_{N+1} := \infty$, $x_0 := -\infty$

Primitives: Access $\mathbb{N} \ni n \to x[n] \in X$ cost 1

ordered comparison "$x \leq y$?" cost 1

Index integer arithmetic cost 1

```
if x[1] > y then return 0;
for n=1…N
  if x[n] ⋗ y then return n-1;
return N.
```

runtime $O(N)$

correctness: √

2. Searching

binary search

Specification: Fix set X with total order $\leq$.

Input: $N \in \mathbb{N}$ and finite sequence $x_1, \ldots x_N \in X$ as well as $y \in X$. — in array $x[1 \ldots N]$ with $x_1 \leq \ldots \leq x_N$

Output: smallest $r \leq N+1$ such that $y < x_r$ — where $x_{N+1} := \infty$, $x_0 := -\infty$

Primitives: Access $\mathbb{N} \ni n \to x[n] \in X$ — cost 1

ordered comparison "$x \leq y$?" — cost 1

Index integer arithmetic — cost 1

```
l:=0; r:=N+1;
while l+1<r do
  n := ⌊(l+r)/2⌋;
  if y<x[n] then r:=n else l:=n;
```

runtime $O(\log N)$

correctness: $x[l] \leq y < x[r]$

time analysis: $r' - l' \leq \lceil (r-l)/2 \rceil$

2. Searching

uniqueness

Specification: Fix set X with total order $\leq$.

Input: $N \in \mathbb{N}$ and finite sequence $x_1, \ldots x_N \in X$ in array $x[1 \ldots N]$ with $x_1 \leq \ldots \leq x_N$

Output: **1** if all elements are distinct: $\forall i,j: x_i = x_j \Rightarrow i = j$

0 if some element is repeated: $\exists i \neq j: x_i = x_j$

Primitives: Access $\mathbb{N} \ni n \to x[n] \in X$ cost 1

ordered comparison "$x \leq y$?" cost 1

Uniq1 ($x[1..N]$) runtime $O(N^2)$

For i:=2 to N do

  For j:=1 to i-1 do

    If $x[i]=x[j]$ Return **0**;

Return **1**;

Uniq2 ($x[1..N]$) runtime $O(N)$

For m:=1 to $N-1$ do

  If $x[m]=x[m+1]$ Return **0**;

Return **1**;

2. Searching

uniqueness

Specification: Fix set $X=\{0,\ldots M-1\}$

Input: $N\in\mathbb{N}$ and finite sequence $x_1,\ldots x_N\in X$ in array $x[1\ldots N]$

Output: **1** if all elements are distinct: $\forall i,j:\ x_i=x_j \ \Rightarrow\ i=j$

0 if some element is repeated: $\exists i\neq j:\ x_i=x_j$

Primitives: Iterable access $\mathbb{N}\ni n \to m:=x[n]\in\mathbb{N}$ cost 1

Uniq3 ($x[1..N]$)

Boolean array *present*$[0\ldots M-1]$; // initialized with **0/false**

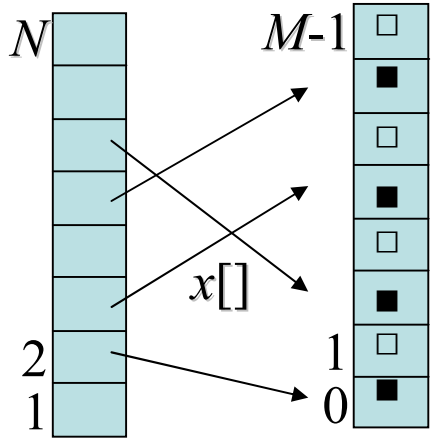
For n:=1 to N do

  If *present*$[x[n]]$ then return **0**;

  present$[x[n]]$:= **1**;

Endfor; return **1**

runtime $O(N)$
memory $O(M)$



2. Searching

hash functions
$\phi_k(x) = x \cdot k \bmod P$

(division) hashing

Specification: Fix set $X=\{1,\ldots M-1\}$

Input: $N \in \mathbb{N}$ and finite sequence $x_1,\ldots x_N \in X$ in array $x[1\ldots N]$

Output: **1** if all elements are distinct: $\forall i,j:\ x_i=x_j \ \Rightarrow\ i=j$

0 if some element is repeated: $\exists i\neq j:\ x_i=x_j$

Primitives: Arithmetic (on values!)

Let $P \geq N$ be prime.

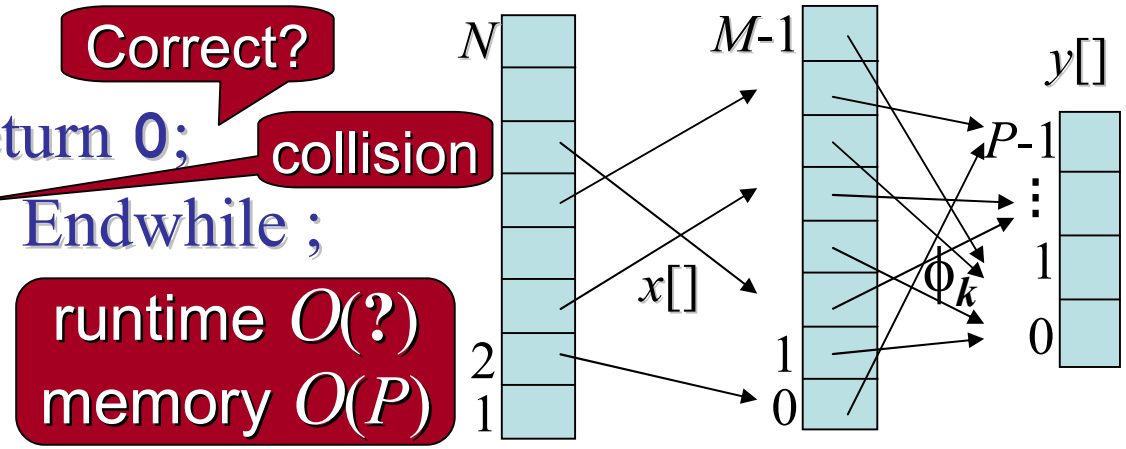
Guess *random* $k \in \{1,\ldots P-1\}$

```
Integer array y[0…P–1]; // initialized with 0s
For n:=1 to N do ;   p := x[n]·k mod P;
  While y[p]≠0 do
     If y[p]=x[n] then return 0;
     p := (p + 1) mod P ;  Endwhile ;
   y[p] := x[n] ;
Endfor; return 1
```

Correct?

collision

runtime $O(?)$
memory $O(P)$



Hashing with Collision Illustrated

0	1	2	3	4	5	6					
17	34	−17	0	0	0	6	−11	23	40	0	0

$P = 17,\ k = 1$:

$\Phi_k(x) = x \bmod 17$

```
Integer array y[0…P–1]; // initialized with 0s
For n:=1 to N do ;   p := x[n]·k  mod  P;
  While  y[p]≠0  do
     If  y[p]=x[n]  then return 0;
     p := (p + 1) mod P ;  Endwhile ;
  y[p] := x[n] ;
Endfor;  return 1
```

$P > N$

Correct??

Worst-case!

Average case?

Choice of P?

runtime $O(N \cdot P)$

memory $O(P)$

2. Searching

order statistics/ median

Specification: Fix set X with total order $\leq$.

Input: $N \in \mathbb{N}$ and finite sequence $x_1, \ldots x_N \in X$ in array $x[1 \ldots N]$

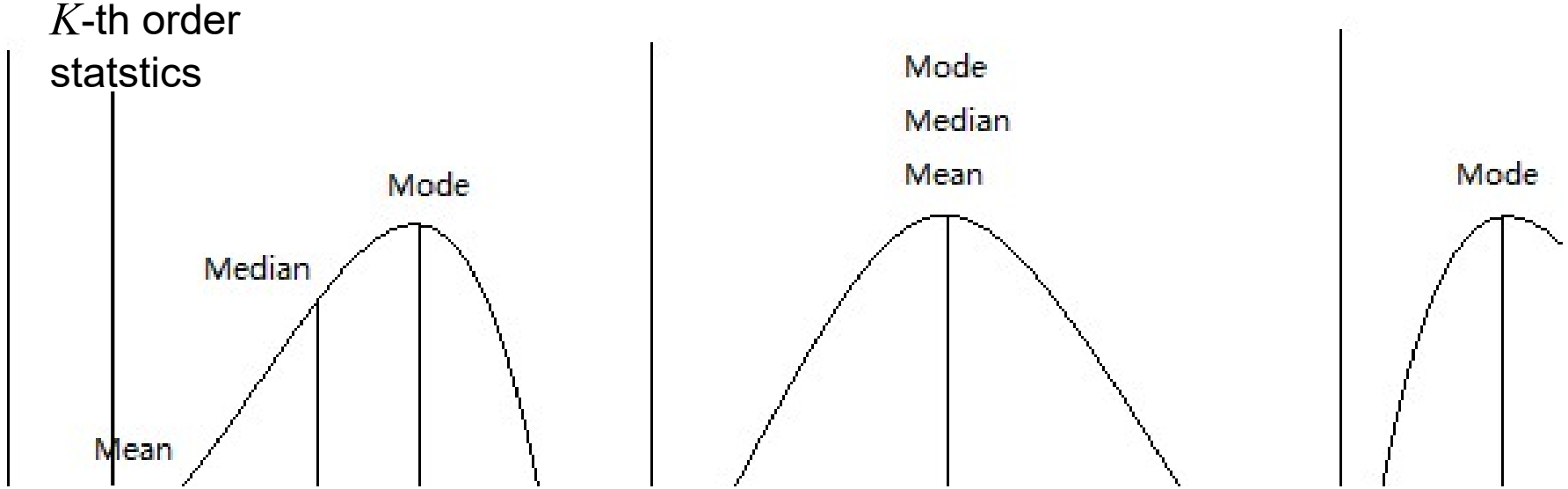
as well as $K \in \{1, \ldots, N\}$.

always exists and is unique!

Output: $m \in \{1, \ldots N\}$ s.t. $\#\{n : x_n < x_m\} < K \leq \#\{n : x_n \leq x_m\}$

Primitives: Access $\mathbb{N} \ni n \to x[n] \in X$ cost 1

ordered comparison "$x \leq y$?" cost 1



2. Searching

order statistics/ median

Specification: Fix set X with total order $\leq$.

Input: $N \in \mathbb{N}$ and finite sequence $x_1, \ldots x_N \in X$ in array $x[1 \ldots N]$
as well as $K \in \{1, \ldots, N\}$. with $x_1 \leq \ldots \leq x_N$

Output: $m \in \{1, \ldots N\}$ s.t. $\#\{ n : x_n < x_m \} < K \leq \#\{ n : x_n \leq x_m \}$

Primitives: Access $\mathbb{N} \ni n \to x[n] \in X$ cost 1

ordered comparison "$x \leq y$?" cost 1

OrderStat1 ($x[1..N]$; K)

runtime $O(N^2)$

For m:=1 to N do
  a:=0; b:=0; For n:=1 to N do
    If $x[n] < x[m]$ then a:=a+1;
    If $x[n] \leq x[m]$ then b:=b+1;
  If $a < K \leq b$ then Return m;

OrderStat2 ($x[1..N]$; K)
Return K;

runtime $O(1)$

2. Searching

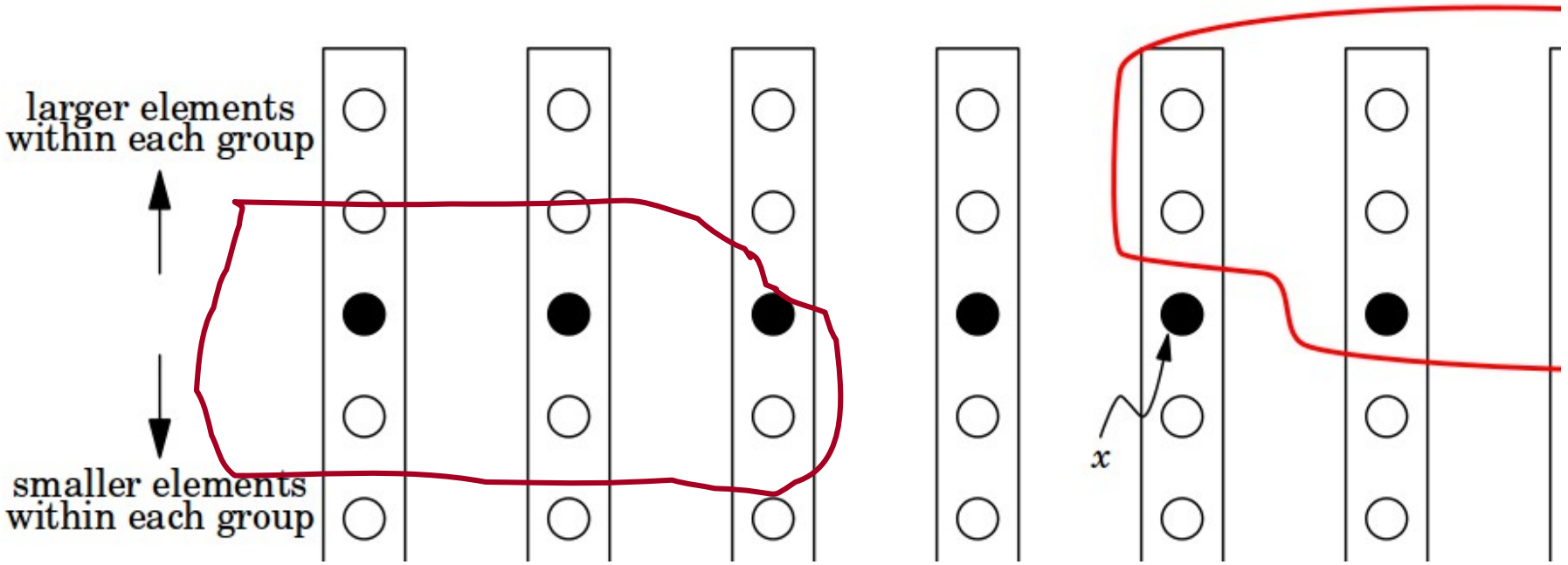
Approximate Median

Specification: Fix set X with total order $\leq$.

unsorted!

Input: $N \in \mathbb{N}$ and finite sequence $x_1, \ldots x_N \in X$ in array $x[1 \ldots N]$

Output: m such that $0.3 \cdot N \leq \#\{n : x_n \leq x_m\} \leq 0.7 \cdot N + 1$



Lemma: *Median of 5-medians* is

"$\geq$" at least $\frac{1}{2} \cdot \frac{3}{5} = 30\%$ of entries, thus "$<$" at most 70%.

2. Searching

Linear-time Median

Specification: Fix set X with total order $\leq$.

unsorted!

Input: $N \in \mathbb{N}$ and finite sequence $x_1, \ldots x_N \in X$ in array $x[1 \ldots N]$

Output: m such that $0.3 \cdot N \leq \#\{n : x_n \leq x_m\} \leq 0.7 \cdot N + 1$

Function **ApproxMedian** ($x[], l, r$) | $T_O(n) = O(n)$ | $T_A(n) = O(n)$

Process $x[l...r]$ in groups of 5, sorting each one to find its median.

Then call **OrderStat** to determine the median of these 5-medians.

Function **OrderStat** ($x[], \boldsymbol{l}, \boldsymbol{r}, K$)

While $\boldsymbol{l} < \boldsymbol{r}$ do $O(n)$, $n := r - l + 1$

$T_A(n) = O(n) + T_O(0.2 \cdot n)$,

$T_O(n) = T_A(n) + O(n) + T_O(0.7 \cdot n)$

Call **ApproxMedian:** determine *approx.* median $\boldsymbol{m}$ of $x[l \ldots r]$.

$m' :=$ **Split**$(x, l, r, x[m])$ s.t. all elements $< x[m]$ are left of those $> x[m]$

Proceed to the left (=decrease $\boldsymbol{r}$) /right (=increase $\boldsymbol{l}$) accordingly.

2. Searching

Splitting

Specification: Fix set X with total order $\leq$.

Input: $N \in \mathbb{N}$ and finite sequence $x_1, \ldots x_N \in X$

unsorted!

in array $x[1 \ldots N]$

```
Function Split ( x[] ; l , r ; y)                  // splits x[l..r] into
If  l ≥ r  return.   // nothing to do               // x[l..m] entries <y  and
a := l ;  b := r ;  While  a < b  do                // x[m+1..r] those ≥y.
                                                    // Returns m.
   While  a<b  and  x[a]  < y  do  a := a+1   Endwhile;
   While  a<b  and  x[b] ≥ y  do  b := b−1   Endwhile;
   Swap(x[a],x[b]);   Endwhile      //   l ≤ a = b ≤ r
```

Call **ApproxMedian:** determine *approx.* median ***m*** of $x[l \ldots r]$.

Split $x[l \ldots r]$ s.t. all elements $<x[m]$ are left of all those $>x[m]$.

$O(n)$, $n := r-l+1$

2. Searching

1D Range Counting

Specification: Fix set X with total order $\leq$.

Input: $N \in \mathbb{N}$ and finite sequence $x_1, \ldots x_N \in X$
as well as input: $x' < x'' \in X$.

in array $x[1 \ldots N]$
with $x_1 \leq \ldots \leq x_N$

Output: $\#\{ m : x' < x_m \leq x'' \} \in \mathbb{N}$

Recall
r=**BinSearch**($x[], y$):
smallest $r \leq N+1$
such that $y < x_r$

Primitives: Access $\mathbb{N} \ni n \to x[n] \in X$
ordered comparison

Range1 ($x[]$; x' , x'')
t:=0; For m:=1 to N do
If $x' < x[m] \leq x''$
then $t := t+1$;

runtime $O(N)$

Range2 ($x[]$; x' , x'')
l := BinSearch($x[]$; x')
r := BinSearch($x[]$; x'')
If r-$l \leq 0$ then Return 0
else Return r-l;

runtime $O(\log N)$

2. Searching

1D Range Counting, *Range Tree*

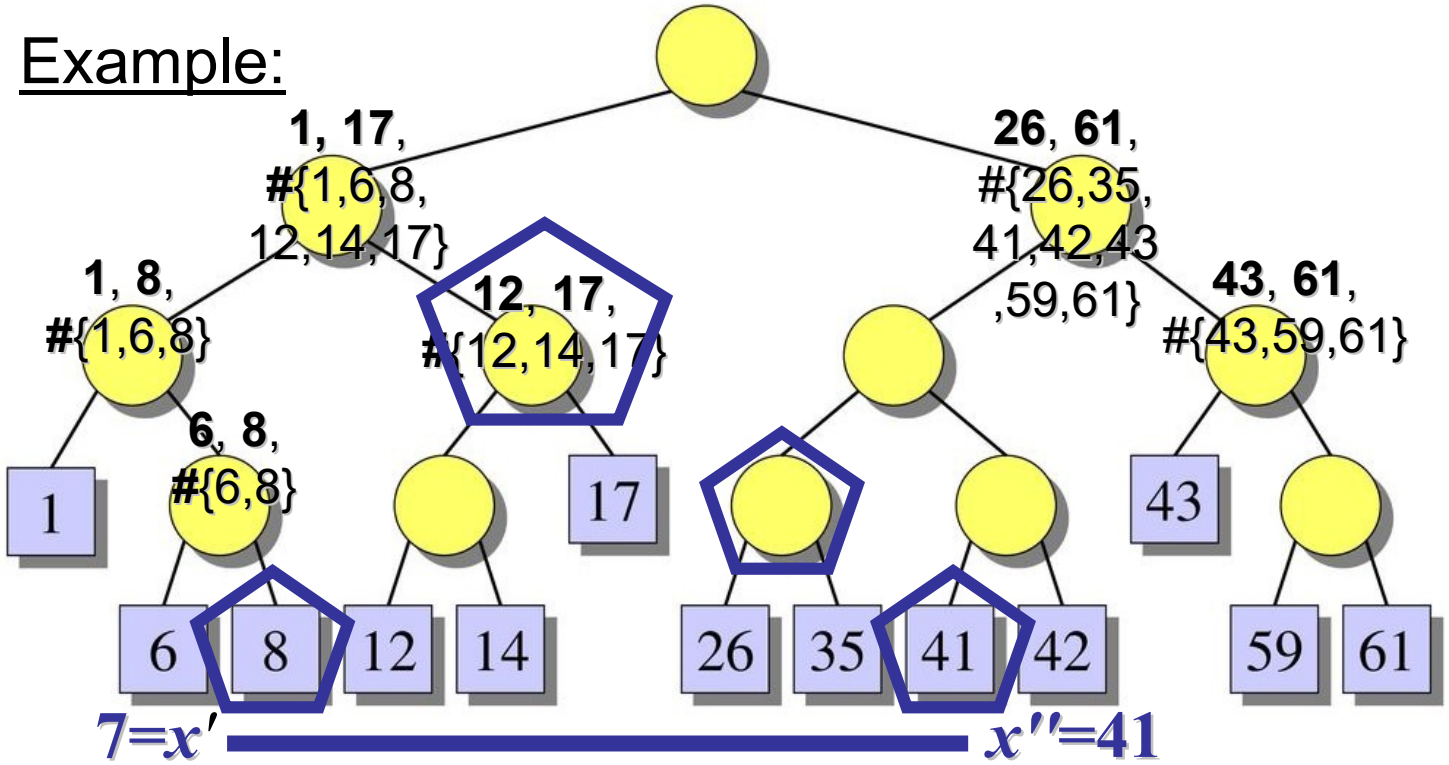
Specification: Fix set X with total order $\leq$.

Input: $N \in \mathbb{N}$ and finite sequence $x_1, \ldots x_N \in X$
as well as input: $x' < x'' \in X$.

Output: $\#\{ m : x' < x_m \leq x'' \} =: K$

Defer issues of preprocessing!

Example:



Memory: $O(N)$

Depth/ runtime $O(\log N)$

2. Searching

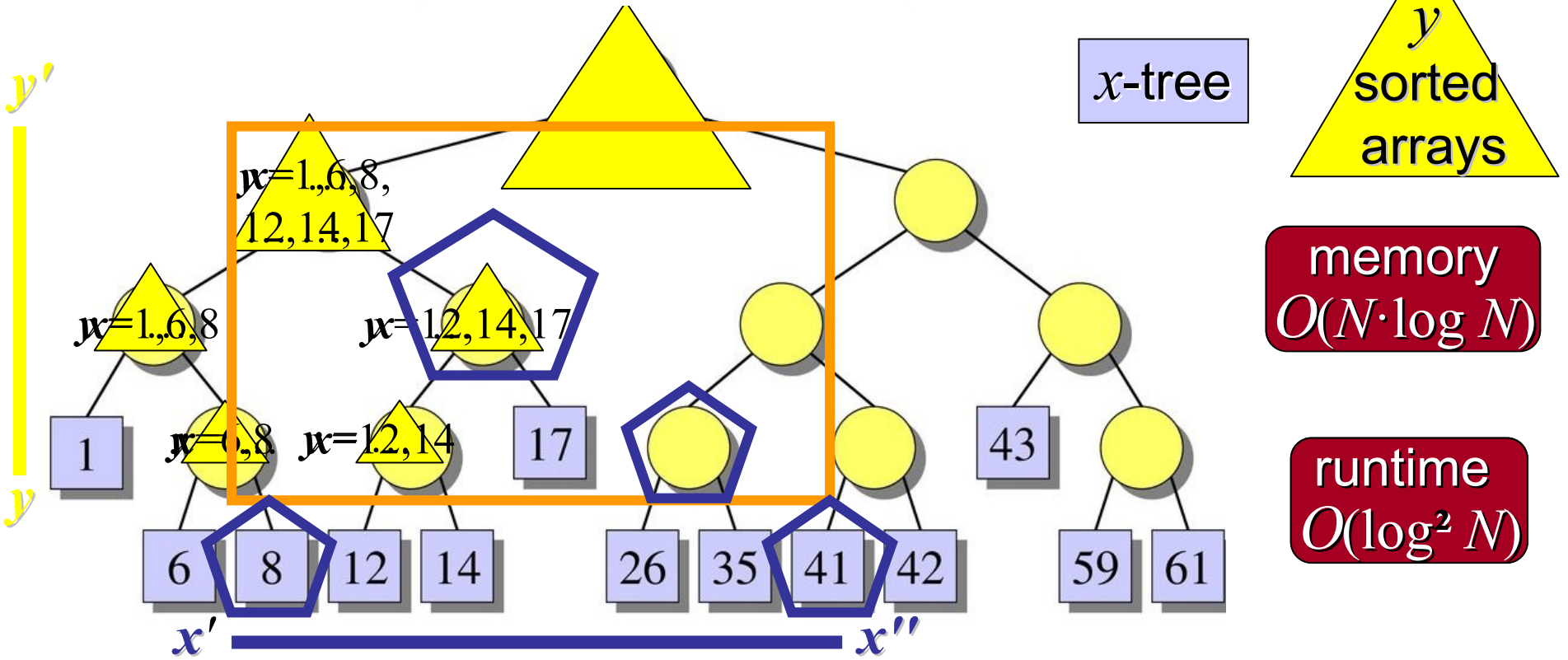
2D Range Counting

Specification: Fix sets X, Y with total orders $\leq$.

Input: $N \in \mathbb{N}$ and finite sequence $(x_1, y_1), \ldots (x_N, y_N) \in X \times Y$
as well as input: $(x', y') < (x'', y'')$.

Output: $\#\{\, m : (x', y') < (x_m, y_m) \leq (x'', y'') \,\} \in \mathbb{N}$

componentwise



2. Searching

3D Range Counting

Nested Trees

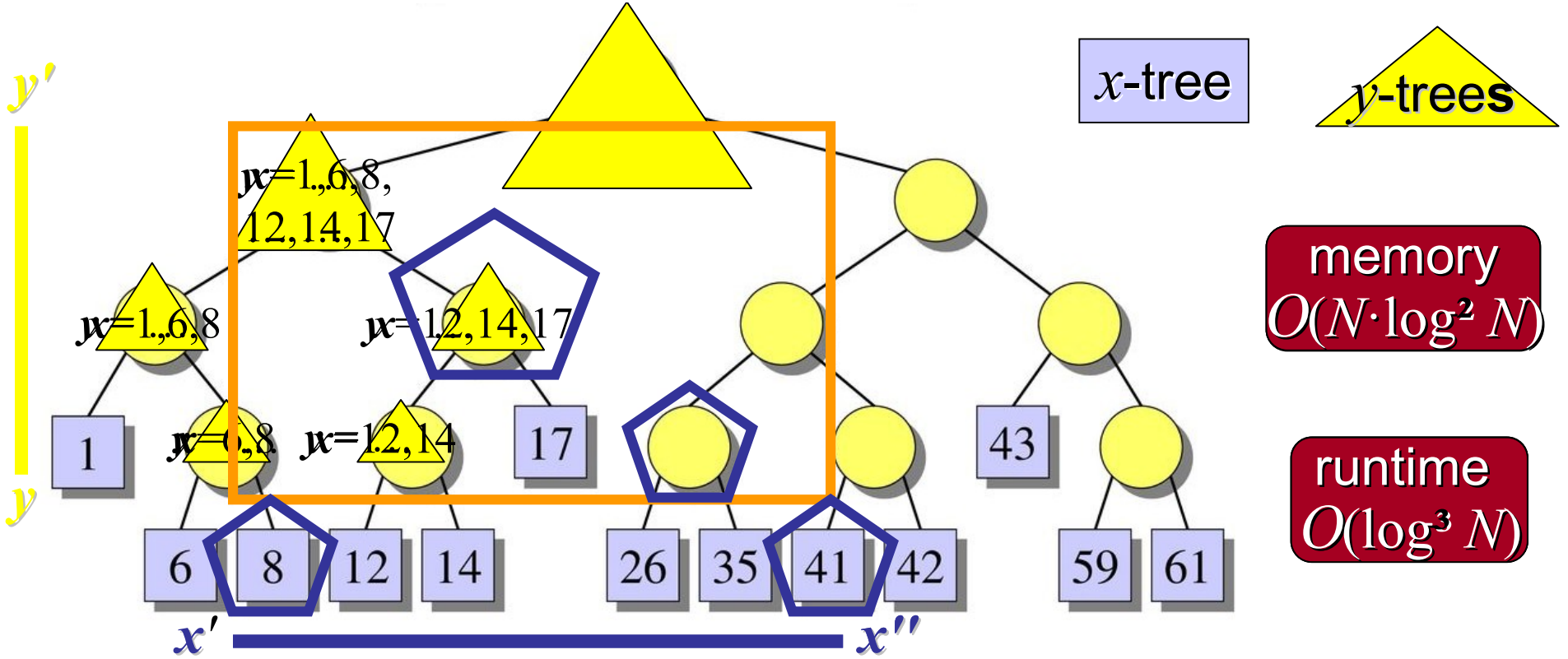
Specification: Fix sets X, Y, Z with total orders $\leq$.

Input: $N \in \mathbb{N}$ and finite sequence $(x_1,y_1,z_1),\ldots(x_N,y_N,z_N) \in X \times Y \times Z$

as well as input: $(x,y,z) < (x',y',z')$.

componentwise

Output: $\#\{\, m : (x,y,z) < (x_m,y_m,z_m) \leq (x',y',z') \,\} \in \mathbb{N}$



2. Searching

1D Range Reporting

Specification: Fix set X with total order $\leq$.

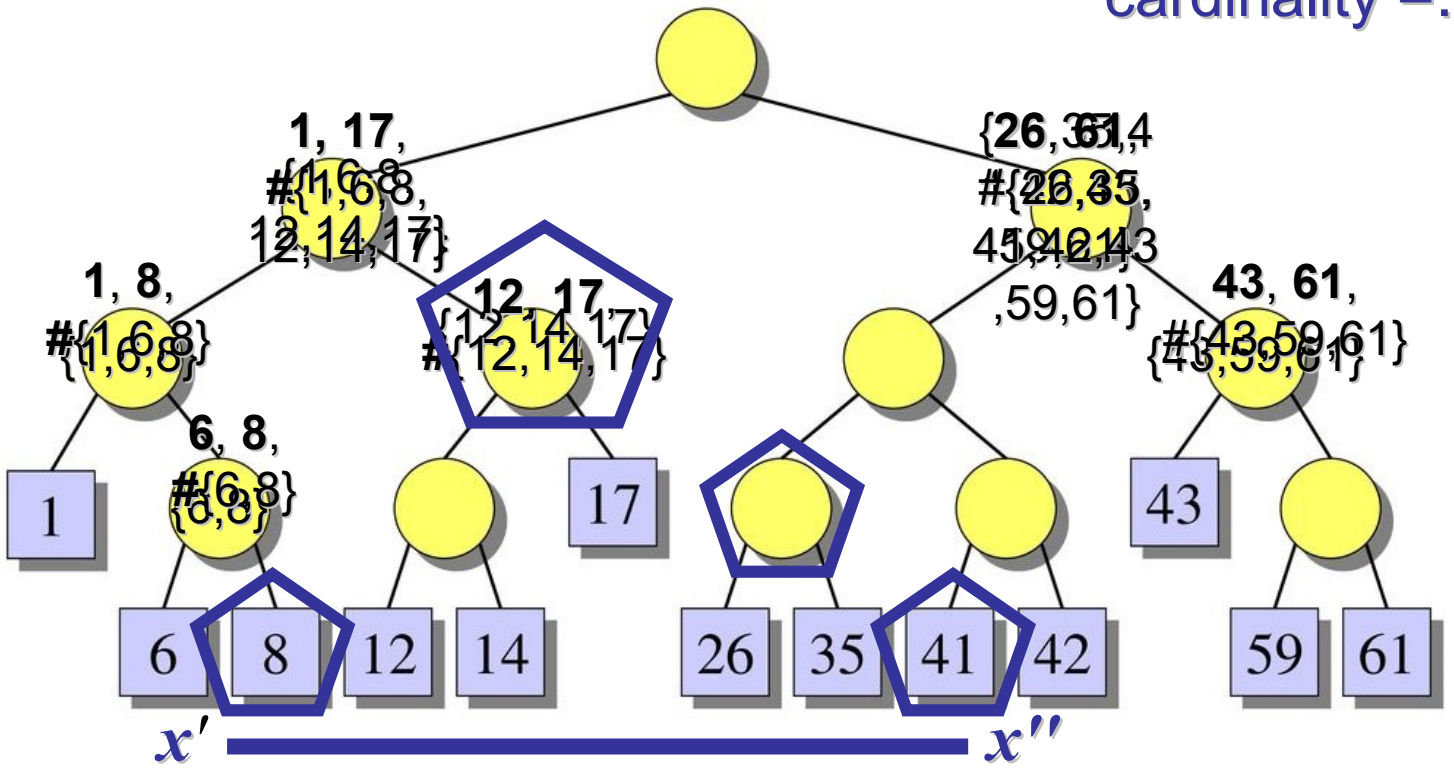
Input: $N \in \mathbb{N}$ and finite sequence $x_1, \ldots x_N \in X$

as well as input: $x' < x'' \in X$.

Output: $\#\{ m : x' < x_m \leq x'' \}$

not a number, but a set

cardinality =: $K \in \{0 \ldots N\}$



memory $O(N \cdot \log N)$

runtime $O(K + \log N)$

Recap

2. Searching

- Linear Search
- Neighbor Search
- Binary Search
- Uniqueness
- Hashing
- Order Statistics/ Median in linear time!
- 1D Range Counting/Reporting
- 2D Range Counting/Reporting

- Specification
- Primitives: semantics and cost
- Design
- Analysis
- (Optimality)