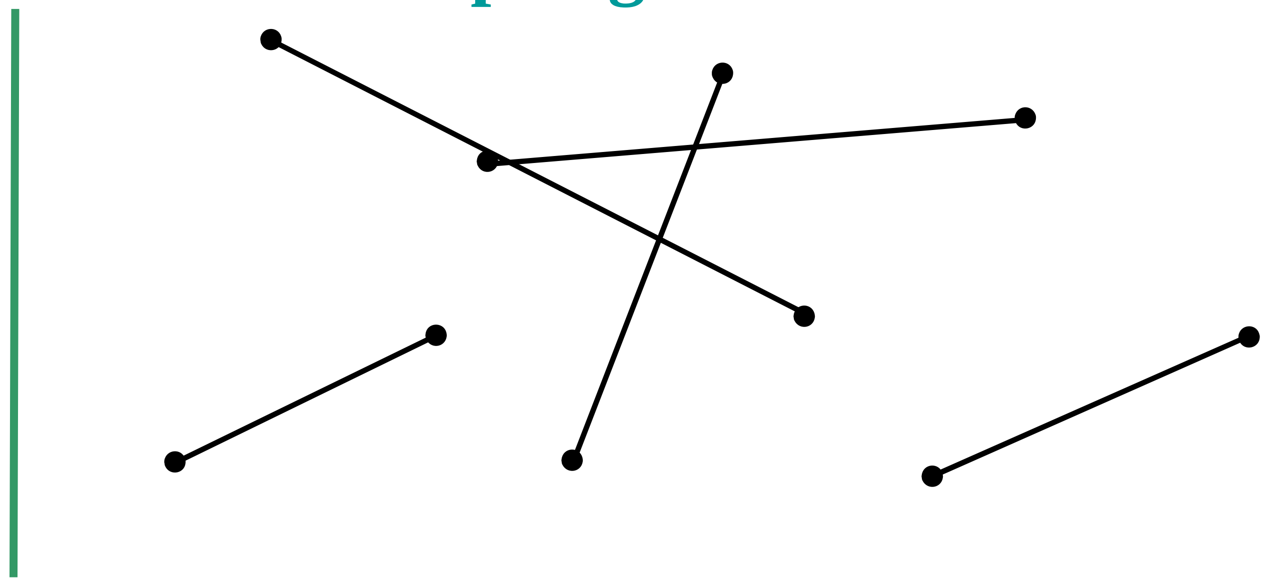
CMPS 3130/6130 Computational Geometry
Spring 2015

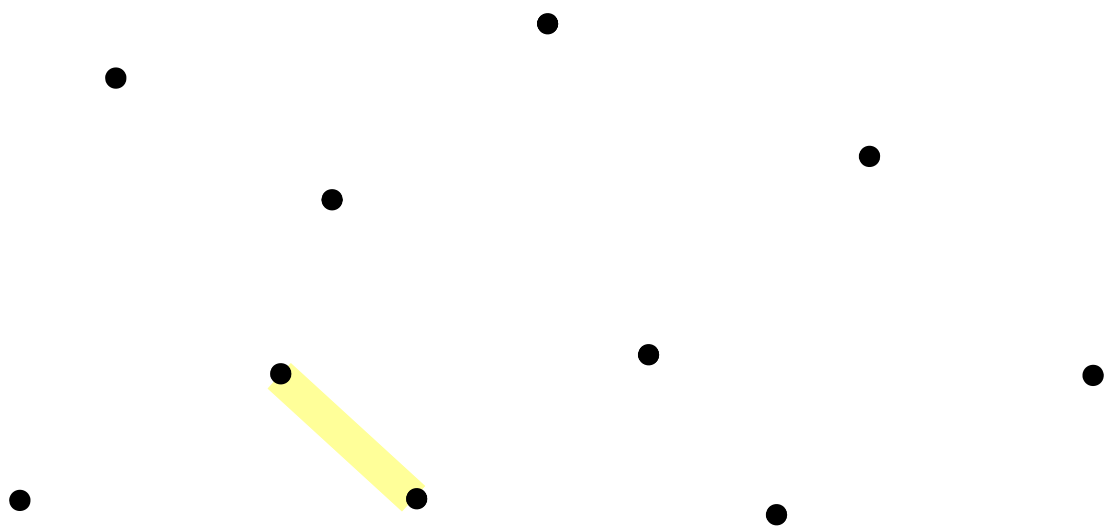


Plane Sweep Algorithms I

Carola Wenk

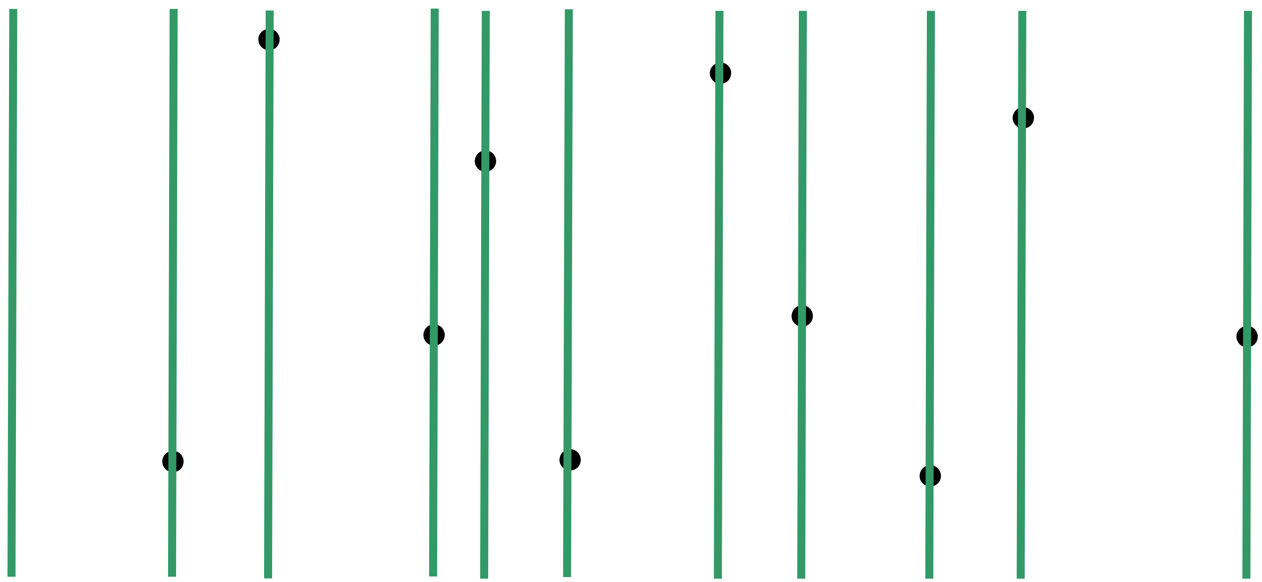
Closest Pair

- **Problem:** Given $P \subseteq \mathbf{R}^2$, $|P|=n$, find the distance between the closest pair in P



Plane Sweep: An Algorithm Design Technique

- Simulate sweeping a vertical line from left to right across the plane.
- Maintain **cleanliness property**: At any point in time, to the left of sweep line everything is clean, i.e., properly processed.
- **Sweep line status**: Store information along sweep line
- **Events**: Discrete points in time when sweep line status needs to be updated



Plane Sweep: An Algorithm Design Technique

- Simulate sweeping a vertical line from left to right across the plane.
- Maintain **cleanliness property**: At any point in time, to the left of sweep line everything is clean, i.e., properly processed.
- **Sweep line status**: Store information along sweep line
- **Events**: Discrete points in time when sweep line status needs to be updated

Algorithm Generic_Plane_Sweep:

Initialize **sweep line status** S at time $x=-\infty$

Store initial events in **event queue** Q, a priority queue ordered by x-coordinate

while $Q \neq \emptyset$

    // extract next event e:
    e = Q.extractMin();
    // handle event:
    Update sweep line status
    Discover new upcoming events and insert them into Q

Plane sweep for Closest Pair

Algorithm Generic_Plane_Sweep:

Initialize **sweep line status** S at time $x=-\infty$
Store initial events in **event queue** Q, a priority queue ordered by x-coordinate
while $Q \neq \varnothing$
  // extract next event e:
  e = Q.extractMin();
  // handle event:
  Update sweep line status
  Discover new upcoming events and insert them into Q

- **Problem:** Given $P \subseteq \mathbf{R}^2$, $|P|=n$, find the distance of the closest pair in P
- **Sweep line status:**
  - Store current distance Δ of closest pair of points to the left of sweep line
  - Store points in Δ-strip left of sweep line
  - Store pointer to leftmost point in strip
- **Events:** All points in P. No new events will be added during the sweep.
  → Presort P by x-coordinate.

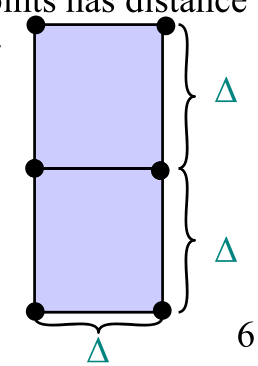
Cleanliness property

Plane sweep for Closest Pair, II

Algorithm Generic_Plane_Sweep:

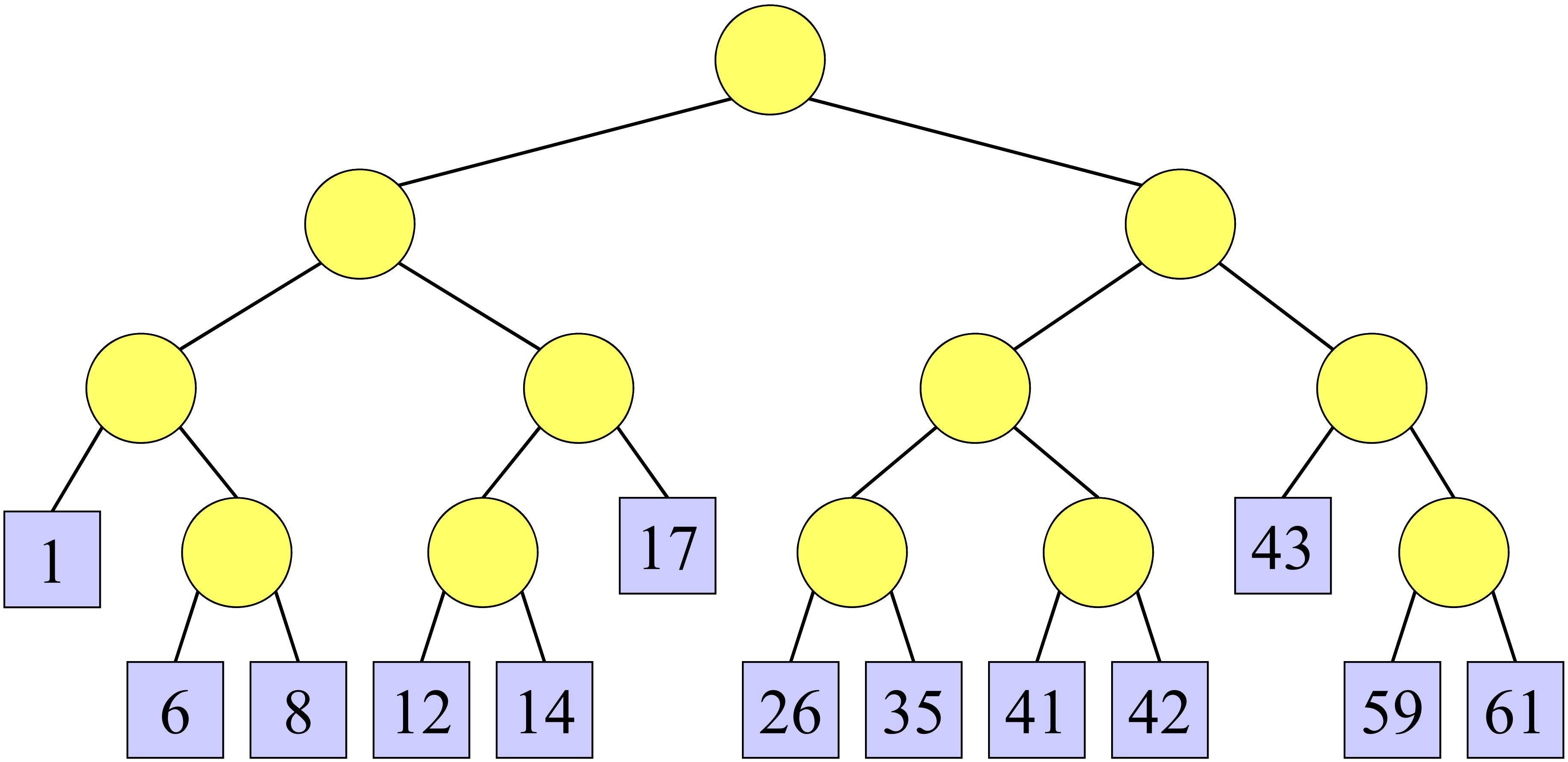
Initialize **sweep line status** S at time $x=-\infty$
Store initial events in **event queue** Q, a priority queue ordered by x-coordinate
while $Q \neq \emptyset$
    // extract next event e:
    e = Q.extractMin();
    // handle event:
    Update sweep line status
    Discover new upcoming events and insert them into Q

$O(n \log n)$
- Presort P by x-coordinate
- How to store points in Δ-strip?
  - Store points in Δ-strip left of sweep line in a balanced binary search tree, ordered by y-coordinate
    $\rightarrow$ Add point, delete point, and search in $O(\log n)$ time
- **Event handling:**
  - New event: Sweep line advances to point $p \in P$
  - Update sweep line status:
    - $O(n \log n)$ total — (1) Delete points outside Δ-strip from search tree by using previous leftmost point in strip and x-order on P
    - $O(n \log n + 6n)$ total — (2) Compute candidate points that may have distance $\leq \Delta$ from p:
      - Perform a search in the search tree to find points in Δ–strip whose y-coordinates are at most Δ away from $p.y$.
        $\rightarrow$ Δ x 2Δ rectangle
      - Because of the cleanliness property each pair of these points has distance $\geq \Delta$.
        $\rightarrow$ A Δ x 2Δ rectangle can contain at most 6 such points.
    - $O(6n)$ total — (3) Check distance of these points to p, and possibly update Δ
  - No new events necessary to discover



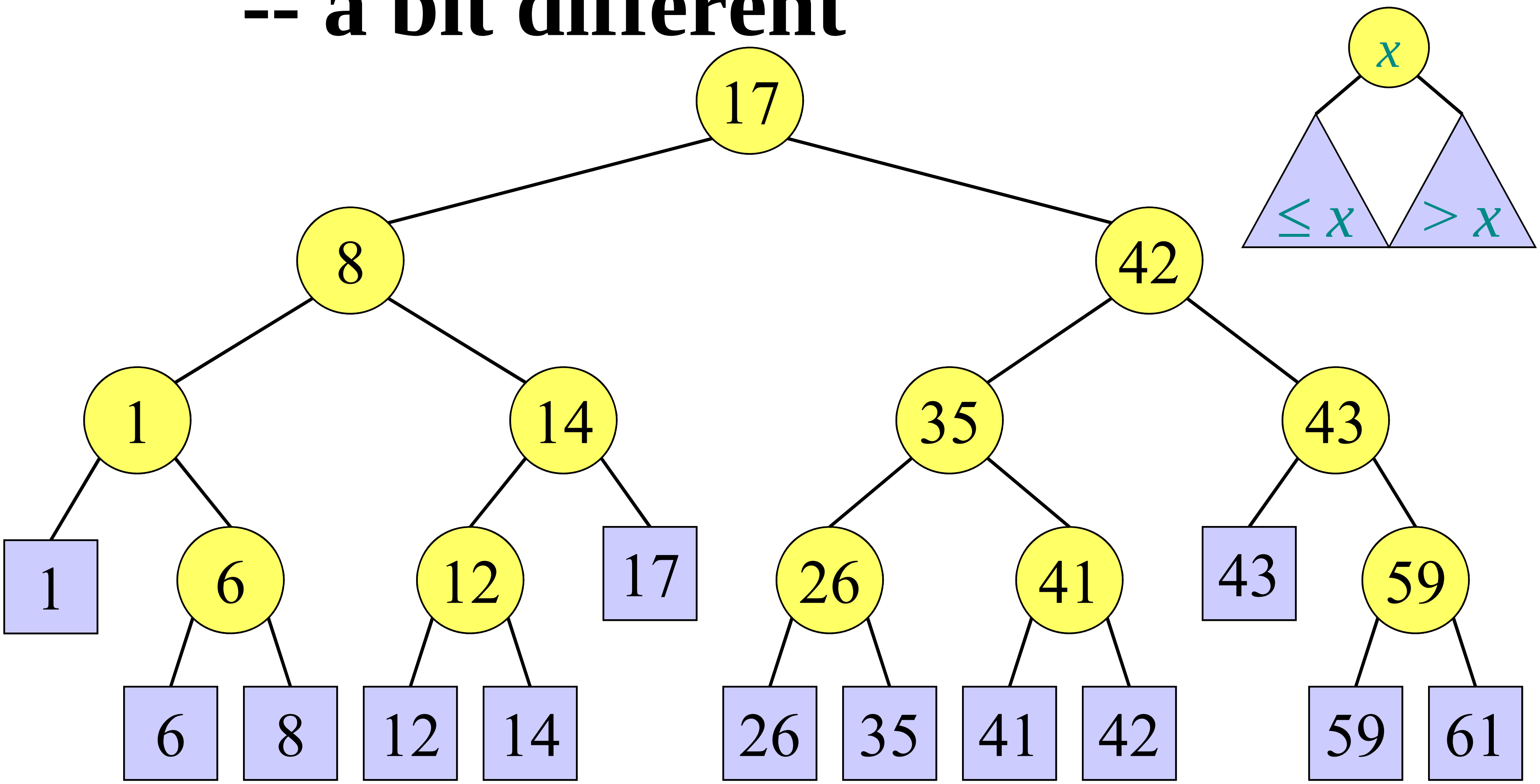
Total runtime: $O(n \log n)$

Balanced Binary Search Tree -- a bit different



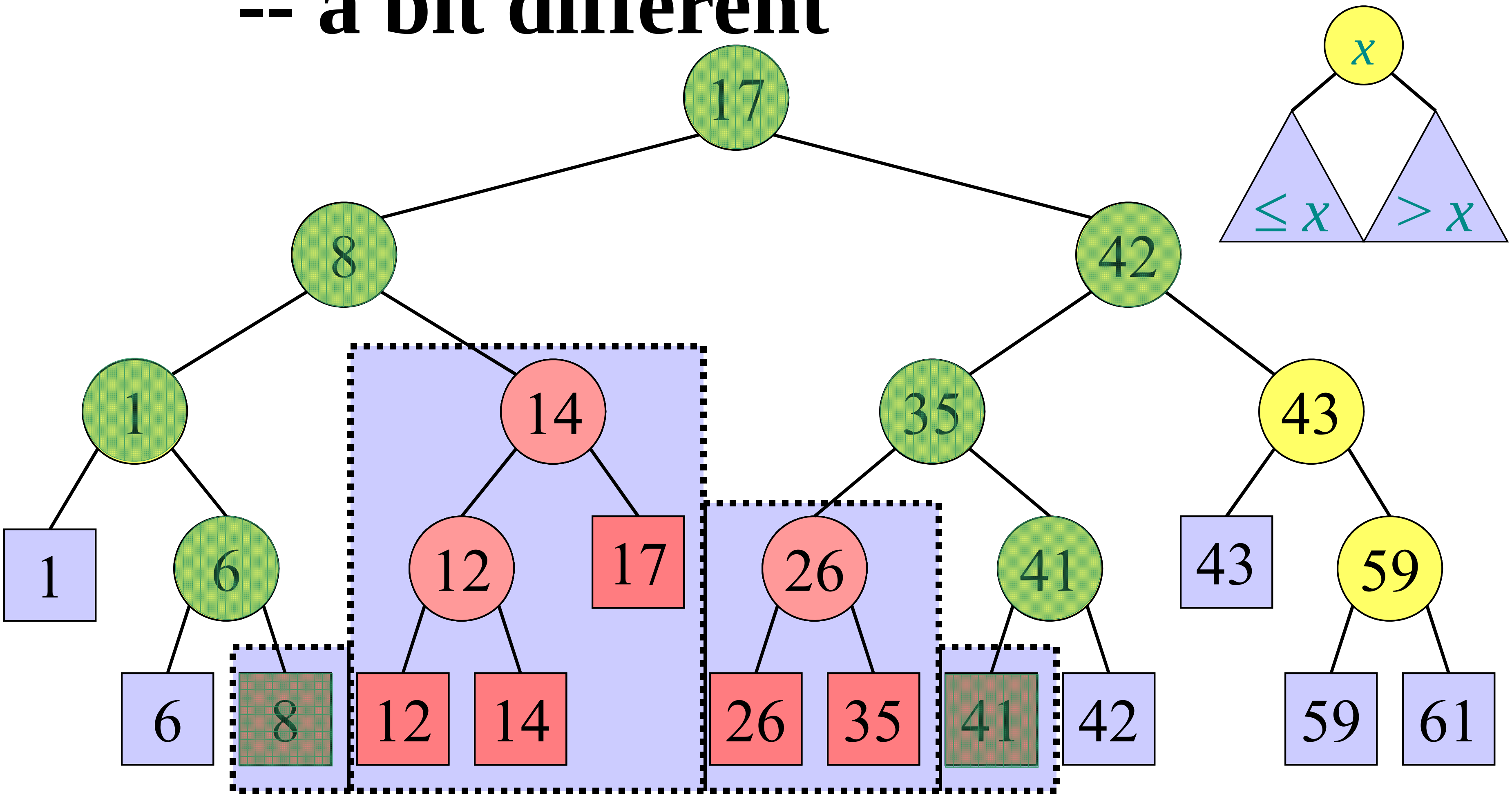
key[*x*] is the maximum key of any leaf in the left subtree of *x*.

Balanced Binary Search Tree -- a bit different



key[*x*] is the maximum key of any leaf in the left subtree of *x*.

Balanced Binary Search Tree -- a bit different



RANGE-QUERY([7, 41])

Plane Sweep: An Algorithm Design Technique

- Plane sweep algorithms (also called sweep line algorithms) are a special kind of incremental algorithms
- Their correctness follows inductively by maintaining the cleanliness property
- *Common* runtimes in the plane are $O(n \log n)$:
  - n events are processed
  - Update of sweep line status takes $O(\log n)$
  - Update of event queue: $O(\log n)$ per event