Binary Search Trees

Key set \( K = \{3, 7, 12, 18, 25, 29, 37, 43, 51, 55, 61, 71\} \)

Size \( n = |K| \)
Key set $K = \{3, 7, 12, 18, 25, 29, 37, 43, 51, 55, 61, 71\}$

Size $n = |K|$

$T$ Binary Search Tree for $K$

$v$ node of $T$: $v$.LC, $v$.RC, $v$.PAR, $v$.key

$T_v$ subtree rooted at $v$

$K_v$ keys in $T_v$
Binary Search Trees

Key set $K = \{3, 7, 12, 18, 25, 29, 37, 43, 51, 55, 61, 71\}$

Size $n = |K|$

$T$ Binary Search Tree for $K$

$v$ node of $T$: $v$.LC, $v$.RC, $v$.PAR, $v$.key

$T_v$ subtree rooted at $v$

$K_v$ keys in $T_v$
Key set $K = \{3, 7, 12, 18, 25, 29, 37, 43, 51, 55, 61, 71\}$

Size $n = |K|$ 

$T$ Binary Search Tree for $K$

$v$ node of $T$: $v$.LC, $v$.RC, $v$.PAR, $v$.key

$T_v$ subtree rooted at $v$

$K_v$ keys in $T_v$

Additional leaf for each primitive interval

$\overline{T}$ Extended Binary Search Tree for $K$
Binary Search Trees

Key set $K = \{3, 7, 12, 18, 25, 29, 37, 43, 51, 55, 61, 71\}$

Size $n = |K|$

$T$ Binary Search Tree for $K$

$v$ node of $T$: $v$.LC, $v$.RC, $v$.PAR, $v$.key

$T_v$ subtree rooted at $v$

$K_v$ keys in $T_v$

Additional leaf for each primitive interval

$\overline{T}$ Extended Binary Search Tree for $K$

$I_v$ is the union of the primitive intervals associated with the leaves of $\overline{T_v}$ together with $K_v$
Binary Search Trees

Key set $K = \{3, 7, 12, 18, 25, 29, 37, 43, 51, 55, 61, 71\}$

Size $n = |K|$

$T$ Binary Search Tree for $K$

$v$ node of $T$: $v$.LC, $v$.RC, $v$.PAR, $v$.key

$T_v$ subtree rooted at $v$

$K_v$ keys in $T_v$

Additional leaf for each primitive interval

$\overline{T}$ Extended Binary Search Tree for $K$

$I_v$ is the union of the primitive intervals associated with the leaves of $\overline{T}$ together with $K_v$

key $x \in \mathbb{R}$:

$\text{path}(x) = \{v \in \overline{T} | x \in I_v\}$
Binary Search Trees

Key set $K = \{3, 7, 12, 18, 25, 29, 37, 43, 51, 55, 61, 71\}$

Size $n = |K|$

$T$ Binary Search Tree for $K$

$v$ node of $T$: $v$.LC, $v$.RC, $v$.PAR, $v$.key

$T_v$ subtree rooted at $v$

$K_v$ keys in $T_v$

Additional leaf for each primitive interval

$\overline{T}$ Extended Binary Search Tree for $K$

$I_v$ is the union of the primitive intervals associated with the leaves of $\overline{T_v}$ together with $K_v$

key $x \in \mathbb{R}$:

$\text{path}(x) = \{v \in \overline{T} | x \in I_v\}$

interval $[\alpha, \beta]$ with $\alpha, \beta \in K$:

$\text{span}[\alpha, \beta] = \{v \in \overline{T} | I_v \subseteq [\alpha, \beta] \text{ but } I_v.PAR \not\subseteq [\alpha, \beta]\}$
Binary Search Trees

Size \( n = |K| \)

\( v \) node of \( T \):
- \( T_v \) subtree rooted at \( v \)
- \( K_v \) keys in \( T_v \)
- \( I_v \) is the union of the primitive intervals associated with the leaves of \( \overline{T_v} \) together with \( K_v \)

Key \( x \in \mathbb{R} \):
- \( \text{path}(x) = \{ v \in T | x \in I_v \} \)

Interval \( [\alpha, \beta] \) with \( \alpha, \beta \in K \):
- \( \text{span}[\alpha, \beta] = \{ v \in \overline{T} | I_v \subseteq [\alpha, \beta] \text{ but } I_v.PAR \not\subseteq [\alpha, \beta] \} \)

Lemma: \( T \) binary tree for \( n \) keys with height \( O(\log n) \).
- for any key \( x \) we have \( |\text{path}(x)| = O(\log n) \)
- for any interval \([\alpha, \beta]\) we have \( |\text{span}[\alpha, \beta]| = O(\log n) \)
- If \( \alpha, \beta \in K \) then \( [\alpha, \beta] = \bigcup \{ I_v | v \in \text{span}[\alpha, \beta] \} \).
- \( \text{path}(x) \) and \( \text{span}[\alpha, \beta] \) can be found in \( O(\log n) \) time.
Size $n = |K|$  

$v$ node of $T$:  
- $T_v$ subtree rooted at $v$  
- $K_v$ keys in $T_v$  
- $I_v$ is the union of the primitive intervals associated with the leaves of $T_v$ together with $K_v$  

A set of objects, each $a \in A$ has a value $a$, key associated with it.  

A range tree for $A$ is a balanced binary search tree $T$ whose key set $K$ contains $\{a.key | a \in A\}$ and that stores for each node $v$ of $T$ the set $A_v = \{a \in A | a.key \in K_v\}$.
Range Trees

Size \( n = |K| \)

\( v \) node of \( T \):
- \( T_v \) subtree rooted at \( v \)
- \( K_v \) keys in \( T_v \)
- \( I_v \) is the union of the primitive intervals associated with the leaves of \( \overline{T_v} \) together with \( K_v \)

A set of objects, each \( a \in A \) has a value \( a.key \) associated with it.

A range tree for \( A \) is a balanced binary search tree \( T \) whose key set \( K \) contains \( \{a.key | a \in A\} \) and that stores for each node \( v \) of \( T \) the set \( A_v = \{a \in A | a.key \in K_v\} \).

**Lemma**: Let \( A \) be a set of objects with keys in \( K \), and \( n = |K| \). Let \( T \) be a range tree for \( A \) with key set \( K \)

- \( \sum_{v \in T} |A_v| = O(|A| \log n) \)
- Given interval \([\alpha, \beta]\) the set \( \{a \in A | a.key \in [\alpha, \beta]\} \) can be found as a disjoint union of \( O(\log n) \) blocks in \( O(\log n) \) time.
- If \( |A| = O(n) \) and the \( A_v \)'s are stored in data structures that admit updates in time \( O(\log^k n) \) then the range tree can be updated in time \( O(\log^{k+1} n) \).
Size $n = |K|$

$v$ node of $T$:
- $T_v$ subtree rooted at $v$
- $K_v$ keys in $T_v$
- $I_v$ is the union of the primitive intervals associated with the leaves of $\overline{T_v}$ together with $K_v$

A set of objects, each $a \in A$ has a segment $a.seg$ associated with it.

A segment tree for $A$ is a balanced binary search tree $T$ whose key set $K$ contains all endpoints of segments $\{a.seg|a \in A\}$ and that stores for each node $v$ of $T$ the set $S_v = \{a \in A|v \in \text{span}(a.seg)\}$. 
Size $n = |K|$

$v$ node of $T$:

- $T_v$ subtree rooted at $v$
- $K_v$ keys in $T_v$
- $I_v$ is the union of the primitive intervals associated with the leaves of $T_v$ together with $K_v$

A set of objects, each $a \in A$ has a segment $a\cdot\text{seg}$ associated with it.

A segment tree for $A$ is a balanced binary search tree $T$ whose key set $K$ contains all endpoints of segments $\{a\cdot\text{seg}| a \in A\}$ and that stores for each node $v$ of $T$ the set $S_v = \{a \in A| v \in \text{span}(a\cdot\text{seg})\}$.

**Lemma:** Let $A$ be a set of objects each associated with a segment with endpoints in $K$. Let $n = |K|$ and let $T$ be a segment tree for $A$ with key set $K$

- $\sum_{v \in T} |S_v| = O(|A| \log n)$
- Given key $x \in \mathbb{R}$ the set $\{a \in A| x \in a\cdot\text{seg}\}$ can be found as a disjoint union of $O(\log n)$ blocks in $O(\log n)$ time.
- If $|A| = O(n)$ and the $S_v$’s are stored in data structures that admit updates in time $O(\log^k n)$ then the segment tree can be updated in time $O(\log^{k+1} n)$. 
Hierarchies of Range and Segment Trees

Example 1:

A set of $n$ objects each having an $x$-key and $y$-key. Build a data structure for $A$ so that for any axis-parallel rectangle $B = x_{seg} \times y_{seg}$ you can tell quickly for which objects in $A$ you have $(a.xkey, a.ykey) \in B$. 
Example 2:

A set of \( n \) objects each having an \( x_{\text{seg}} \) and \( y_{\text{seg}} \), defining the axis-parallel rectangle \( a.\text{Box} = x_{\text{seg}} \times y_{\text{seg}} \).

Build a data structure for \( A \) so that for any query point \( q \in \mathbb{R}^2 \) you can determine quickly for which objects in \( A \) you have \( q \in a.\text{Box} \).
Hierarchies of Range and Segment Trees

Example 3:

A a set of \( n \) horizontal segments \( a \cdot xseg \).

Build a data structure for \( A \) so that for any vertical query segment \( s \) you can determine quickly the segments in \( A \) that intersect \( q \).
Hierarchies of Range and Segment Trees

Example 3:

A a set of \( n \) horizontal segments \( a \cdot xseg \).
Build a data structure for \( A \) so that for any vertical query segment \( s \) you can determine quickly the segments in \( A \) that intersect \( q \).
Example: Given a set of axis parallel boxes in $\mathbb{R}^2$ compute area of their union.
Example: Given a set of axis parallel boxes in $\mathbb{R}^2$ compute area of their union.

Sweep horizontal line $L_t : y = t$ from bottom to top across the plane.
Example: Given a set of axis parallel boxes in $\mathbb{R}^2$ compute area of their union.

Sweep horizontal line $L_t : y = t$ from bottom to top across the plane and maintain an **Invariant** so that in the end the veracity of the invariant implies correctness of the computation.
Sweep Algorithms

Example: Given a set of axis parallel boxes in $\mathbb{R}^2$ compute area of their union.

INV Invariant

SLS (Sweepline structure): Maintains interaction between $L_t$ and the geometry

EQ (Event queue): Priority Queue for predicting the next “event”, i.e. qualitative change during the sweep
Example: Given a set of axis parallel boxes in $\mathbb{R}^2$ compute area of their union.

INV Invariant Geometric-Semantic-Part: Maintain $A_t$ the area of the intersection of the boxes that is in $L_t^-$

SLS (Sweepline structure): Maintains interaction between $L_t$ and the geometry

EQ (Event queue): Priority Queue for predicting the next “event”, i.e. qualitative change during the sweep
Sweep Algorithms

Example: Given a set of axis parallel boxes in $\mathbb{R}^2$ compute area of their union.

SLS (Sweepline structure): Maintains interaction between $L_t$ and the geometry
Let $B_t$ the boxes in $B$ that intersect $L_t$. SLS stores the interval set $\{b \cap L_t | b \in B_t\}$ in a structure that allows updates and queries for the length of the union of all intervals in the structure.
Example: Given a set of axis parallel boxes in $\mathbb{R}^2$ compute area of their union.

EQ (Event Queue): Events happen when $L_t$ meets a lower or upper edge of a box in $B$. There are two types: lower and upper. EQ maintains all these events in a priority queue.
Example: Given a set of axis parallel boxes in $\mathbb{R}^2$ compute area of their union.

Invariant semantic-geometric:
**Example:** Given a set of axis parallel boxes in $\mathbb{R}^2$ compute area of their union.
Example: Given a set of axis parallel boxes in $\mathbb{R}^2$ compute area of their union.
Sweep Algorithms
Sweep Algorithms

Example: Given a set $S$ of $n$ non-horizontal segments in the plane, report all their pairwise intersections.
Example: Given a set $S$ of $n$ non-horizontal segments in the plane, report all their pairwise intersections.

Invariant semantic-geometric:
Sweep Algorithms

Example: Given a set $S$ of $n$ non-horizontal segments in the plane, report all their pairwise intersections.

Invariant SLS:
Sweep Algorithms

**Example:** Given a set $S$ of $n$ non-horizontal segments in the plane, report all their pairwise intersections.

$$L_t$$

**Invariant EQ:**
Sweep Algorithms

Example: Given a set $S$ of $n$ non-horizontal segments in the plane, report all their pairwise intersections.
Sweep Algorithms

Example: Given a set $B$ of $n$ non-horizontal, non-intersecting blue segments in the plane and given a set $R$ of $n$ non-horizontal, non-intersecting red segments, report the number of red-blue intersections.
Sweep Algorithms

Example: Given a set \( B \) of \( n \) non-horizontal, non-intersecting blue segments in the plane and given a set \( R \) of \( n \) non-horizontal, non-intersecting red segments, report the number of red-blue intersections.
Example: Given a set $B$ of $n$ non-horizontal, non-intersecting blue segments in the plane and given a set $R$ of $n$ non-horizontal, non-intersecting red segments, report the number of red-blue intersections.