1 Overview

In the last lecture, we introduced the definition of a dynamically optimal binary search tree (BST); and conjectured that splay trees are dynamically optimal. We then looked at how we can visualize a BST using the geometric view and defined a key property of the resulting point set, an arborally satisfied set [DHI+01].

**Theorem 1.** A point set $P \subseteq X \times [m]$ is arborally satisfied if and only if it corresponds to a valid BST for the query sequence $X = \{x_1, x_2, \cdots, x_{m-1}, x_m\}$.

In this lecture, we will prove this theorem by first proving the if direction, then proving the only if direction.

2 Proof of Theorem

2.1 Proof of the “if” direction

**Proposition 2.** Given a BST on a sequence of search queries $\{x_1, x_2, \cdots, x_{m-1}, x_m\}$, the corresponding geometric view with points $P = \{(x, i)\}$ is an arborally satisfied set.

*Proof.* Consider two points $(x, i), (y, j) \in P$. We will show that the rectangle $((x, i), (y, j))$ is arborally satisfied. Assume that $x < y$ and $i < j$ (proof of other cases is symmetric). Let $LCA_i(a, b)$ be the lowest common ancestor of two nodes $a$ and $b$ of the tree right before the $i^{th}$ query.

![Figure 1: Example of LCA](image_url)
We need to consider three cases:

- **Case 1:**
  If \( \text{LCA}_i(x, y) \neq x \), then \( \text{LCA}_i(x, y) \) was touched during the \( i^{th} \) search query. Hence, due to the structure of BSTs, \( x < \text{LCA}_i(x, y) \leq y \).

  ![Figure 2: Case 1 - Geometric View](image)

- **Case 2:**
  If \( \text{LCA}_j(x, y) \neq y \), then \( \text{LCA}_j(x, y) \) was touched during the \( j^{th} \) search query. By a similar argument as case 1, \( x \leq \text{LCA}_j(x, y) < y \).

  ![Figure 3: Case 2 - Geometric View](image)
Case 3:
If neither of the above cases are true, then \(LCA_i(x, y) = x\) and \(LCA_j(x, y) = y\). This implies that during the \(i^{th}\) search query, \(x\) is an ancestor of \(y\); and during the \(j^{th}\) search query, \(y\) is an ancestor of \(x\). Therefore, there must exist a \(\text{Rotate}_k(z)\) for some node \(z\) on the path, where \(i \leq k < j\) and \(x < z < y\).

In each of the above 3 cases, we have shown that any two arbitrary points in \(P\) are arborally satisfied. Therefore, \(P = \{(x, i)\}\) is an arborally satisfied set.

2.2 Proof of the “only if” direction

Definition 3. A Priority Search Tree (PST) is a tree on keys \{(a, b)\} such that it is a BST on the key \(a\) and a min-heap on the key \(b\).

Proposition 4. Given an arborally satisfied set, \(P = \{(x, i)\}\), there exists a binary search tree execution on a sequence of \(m\) queries, such that the \(i\)-th query touches exactly the nodes with keys \(x\), such that \((x, i) \in P\).

Proof. The BST we will be constructing is allowed to change from one query to another using rotations. On query \(i\), our BST will be \(T_i\) – a PST on \((x_i, N(x_i, i))\), where \(N(x, i)\) is the lowest point on the ray from \((x, i)\) to \((x, \infty)\) in \(P\) (see Figure 5 for an example).

By construction, because PST is a BST on the first coordinate, \(T_i\) is a BST. It only remains to show that we can obtain \(T_{i+1}\) from \(T_i\) using just rotations in time proportional to the number of points in \(P\) that lie on line \(y = i\). Then the time it takes to perform all transformations from \(T_1\) to \(T_m\) will be proportional to the number of points in the geometric representation.
Let $\tau_i$ be the set of points with the second coordinate equal to $i$ (i.e. the ones that lie on line $y = i$).

**Example** Given the point set shown below, we have that: $T_1 = \{(1,1), (2,1), (3,1), (4,3)\}$ and $\tau_1 = \{(1,1), (2,1), (3,1)\}$; and $T_2 = \{(1,\infty), (2,2), (3,2), (4,3)\}$ and $\tau_2 = \{(2,2), (3,2)\}$.

**Observation 5.** $\tau_i$ is a connected subtree of $T_i$ containing the root.

**Proof.** The second coordinate of every node in $\tau_i$ is equal to $i$, i.e. is the same, and the minimum value of $N(x,i)$ is $i$. Thus, by the heap property of PSTs, all nodes of $\tau_i$ will be at the top of the tree, will include the root and will be connected to each other (if they weren’t, the heap order would be violated).

The difference between the nodes of $T_{i+1}$ and $T_i$ is only in the nodes of $\tau_i$, which change the keys from $(x,N(x,i))$ to $(x,N(x,i+1))$. Thus, in order to construct $T_{i+1}$ from $T_i$, we will only rearrange
the nodes of \( \tau_i \) by building the PST on the new keys of these nodes (and leaving the rest of the tree the unchanged). We need to show that this simple change (1) will keep \( T_{i+1} \) as a priority search tree and (2) can be performed in \( O(|\tau_i|) \) time using only rotations.

To show (1), note that \( T_{i+1} \) is a BST by construction and we only need to show that min-heap order is preserved. Consider three cases:

- **Case 1**: If \( a \not\in \tau_i \) and \( b \not\in \tau_i \), then the min-heap order is preserved since the structure of between \( a \) and \( b \) did not change.

- **Case 2**: If \( a \in \tau_i \) and \( b \in \tau_i \), then by our definition of how we construct \( T_{i+1} \), the min-heap order is preserved.

- **Case 3**: If \( a \in \tau_i \) and \( b \not\in \tau_i \), consider the path from \( a \) to \( b \) (if there isn’t one, we don’t need to worry about the preservation of the heap property). Let \((a', b')\) be the edge in \( T_i \), such that \( a' \in \tau_i \) and \( b' \not\in \tau_i \), i.e. it is the edge that straddles the boundary between \( \tau_i \) and the rest of the tree. By the above two cases, the heap property of all the pairs of nodes on the path from \( a \) to \( a' \) and on the path from \( b \) to \( b' \) are preserved. Thus, we can focus only on the nodes \( a' \in \tau_i \) and \( b' \not\in \tau_i \) and \( a' \) is the parent of \( b' \).

Let us assume that the min-heap order between \( a' \) and \( b' \) is violated. We have that:

\[
\begin{align*}
    b' &\not\in \tau_i \\
    \implies N(b', i) &> i \\
    \implies N(b', i) &\geq i + 1 \\
    \implies N(b', i) &= N(b', i + 1).
\end{align*}
\]

Hence, from our assumption that min-heap order is violated, \( N(a', i + 1) > N(b', i + 1) = N(b', i) \). Since \( N(a', i + 1) > N(a', i) = i \), there exists no point on the vertical line segment defined by the two points \((a', i)\) and \((a', N(a', i + 1))\). I.e. there is no point in \( P \) that lies on the vertical edge of the rectangle defined by points \((a', i)\) and \((b', N(b', i))\), incident on point \((a', i)\).

Let us show that neither there is a point on the horizontal edge of the same rectangle, incident on \((a', i)\). Assume there was such point \((c', i)\) \(\in P\). That would mean that that \( a' < c' < b' \).

Since \( c' \) is on line \( y = i \), it follows that \( c' \in \tau_i \). However, then either the BST property will be violated or \( \tau_i \) will not be connected \((a' \text{ is a parent of } b')\). So node \( c' \) cannot exist.

Thus, \((c', i)\) cannot exist in \( P \). But then there are no points on either of the two rectangle edges incident on \((a', i)\) \(\in P\), which by Lemma 11 from previous lecture would imply that \( P \) is not an arborally satisfied set, which is a contradiction. Therefore, min-heap order between \( a' \) and \( b' \) must be preserved.

The final point that remains to be proven is that we can obtain \( T_{i+1} \) from \( T_i \) in \( O(|\tau_i|) \) time using only rotations. Sleator et al. [STT86] showed that any tree on \( n \) nodes can be converted into any other tree on \( n \) nodes in \( O(n) \) time using rotations. Since we modify only nodes of \( \tau_i \) to obtain \( T_{i+1} \), \( T_{i+1} \) can be obtained using rotations in \( O(|\tau_i|) \) time.

\[\Box\]
Thus, we can construct a valid BST algorithm on a sequence of search query in time linear with the number of points in $P$ and the $i$-th search query of the the BST algorithm touches only the nodes of the BST that correspond to points in $P$ that lie on horizontal line $y = i$.

References
