Advanced Data Structures Jan-Apr 2012

Lecture 5-6 February 2-3, 2012

Lecturer: Venkatesh Raman *Scribe: Anil Shukla and Sankardeep Chakraborty*

1 Overview

For the past few lectures we have been studying self organizing binary search trees. In these two lectures we will study a novel connection between binary search trees(BSTs) and points satisfying a simple property from [1]. Using this correspondence, we will restate many results and conjectures relating to BSTs and dynamic optimality.

2 Introduction

For all our analysis and discussions, cost model will be The BST model.

The BST model: For concreteness, we choose the following model for BSTs, among many choices that are constant-factor equivalent. A search is conducted with a pointer starting at the root, which is free to move about the tree and perform rotations; however, the pointer must at some point in the operation visit the item being searched. The cost of a search is simply the total number of distinct nodes in the tree that have been visited by the pointer during the operation. The letters n and m always refer to the size of a BST, and the total number of search operations performed on it, respectively. For simplicity, we denote the ordered values in the BST by the integers $1, 2, \ldots, n$. The letters n and m always refer to the size of a BST, and the total number of search operations performed on it, respectively.

We measure the total cost of executing a sequence of searches S= $\{s_1, s_2, ...s_m\}$, where each search s_i is chosen from among the fixed set of n keys in the BST. Let OPT(S) denote the minimal cost for executing the access sequence S in the BST model, or equivalently, the cost of the best offline BST algorithm which knows S a priori. This value is well-defined and its decision version is in NP (by exhibiting a sequence of rotations and pointer moves).

OPEN PROBLEM 1. Can we compute or approximate OPT(S) in polynomial time?

For static BSTs (no rotations allowed), the question was addressed by a well-known dynamic-programming algorithm of Knuth[2].

OPEN PROBLEM 2. Is there an online BST algorithm whose total cost is O(OPT(S)) for all S?

The best known guarantee is the $O(\lg lgn)$ competitive ratio achieved recently by Tango trees[3]. This is also the best proven approximation factor achieved for the offline problem.

2.1 A geometric view

We present an exact correspondence between the BST model of computation and the following clean question about points in the plane.

Call a set P of points arborally satisfied if, for any two points $a, b \in P$ not on a common horizontal or vertical line, there is at least one point from $P \backslash \{a, b\}$ in the axis aligned rectangle defined by a and b; see Figure 1. We plot an access sequence S in the natural two-dimensional way: $P = \{(s_1, 1), (s_2, 2), ..., (s_m, m)\}$. We show that finding the best BST execution for S is equivalent to finding the minimum cardinality superset $P' \supseteq P$ that is arborally satisfied.

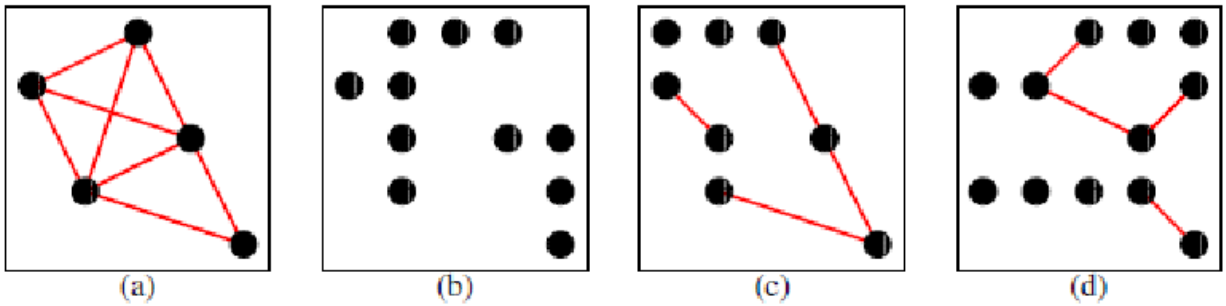

Figure 1: This is a representation of access sequence in BST (lines connect points which are creating unsatisfiable rectangles). The diagrams are:
a) An access sequence which is not arborally satisfied.
b) An arborally satisfied superset of (a)(output of GREEDYFUTURE (refer section 3.2))
c) A superset of (a) that is not arborally satisfied but ◪-satisfied (see section 4.2).
d) A superset of (a) which is neither arborally satisfied nor ◪-satisfied.

3 Trees and Arborally Satisfied Points Sets

We begin with a precise definition of the BST model of computation and its costs:

Definition 1. *Given a BST T_1, a subtree τ of T_1 containing the root, and a tree τ' on the same nodes as τ, we say T_1 can be reconfigured by an operation $\tau \to \tau'$ to another BST T_2 if T_2 is identical to T_1 except for τ being replaced by τ'. The cost of the reconfiguration is $|\tau| = |\tau'|$.*

Definition 2. *Given a search sequence $S = \{s_1, s_2, \ldots, s_m\}$, we say a BST algorithm executes S by an execution $E = \{T_0, \tau_1 \to \tau_1', \ldots, \tau_m \to \tau_m'\}$ if all reconfigurations are valid, and $s_i \in \tau_i$ for all i. For $i = 1, 2, \ldots, m$, define T_i to be T_{i-1} with the reconfiguration $\tau_i \to \tau_i'$. The cost of execution E is $\sum_{i=1}^{m} |\tau_i|$.*

This model is constant-factor equivalent to other BST models.

Observation 1. In an arborally satisfied point set P, for any $a, b \in P$ not orthogonally collinear, there is at least one point from $P \backslash \{a, b\}$ on the sides of $\square ab$ incident to a, and at least one point on the sides incident to b. (The two points need not be distinct.)

Proof. Consider any two points $a, b \in P$ that are not orthogonally collinear. Because $\square ab$ is satisfied, it contains some other point $c \in P$. If c is not on either of the sides of $\square ab$ incident to a, then we can recurse into $\square ac$ until we find such a point. Similarly, if c is not on either of the sides of $\square ab$ incident to b, then we can recurse into $\square cb$ until we find such a point. $\square$

We now plot an execution of the BST algorithm in an intuitive way: at time i (row i), we plot all nodes touched in τ_i. The BST model has been chosen to ignore just the right amount of detail (e.g., precise rotations and pointer movements) to make this geometric view easy.

Definition 3. *The geometric view of a BST execution E is the point set $P(E) = \{(x, y) | x \in \tau_y\}$.*

Lemma 1. The point set $P(E)$ for any BST execution E is arborally satisfied.

Proof. Assume for contradiction that we can find $a \in \tau_i$ and $b \in \tau_j$, with $i < j$ and $a \neq b$, and yet no other nodes in $[a, b]$ were touched in the closed time interval $[i, j]$. Let c be the lowest common ancestor of a and b in tree T_i. We distinguish two cases:

- $c \neq a$ Then c must be touched at time i to get to $a (c \in \tau_i)$ and c must have a key value between a exclusive and b inclusive. Contradiction.
- $c = a$ Then, at time i, a is an ancestor of b. By assumption that $\square ab$ is unsatisfied, b is not touched from time i inclusive to j exclusive. Thus a will remain on the access path of b, i.e., a must be an ancestor of b in T_j , and will be touched then ($a \in \tau_j$). Contradiction.

$\square$

3.1 Offline Equivalence

In this section we will show that we can reconstruct the execution sequence of any geometric view that satisfies the necessary condition of being arborally satisfied.

Lemma 2. For any arborally satisfied point set X, there exists a BST execution E with $P(E) = X$. We call E the arboral view of X, and write $P^{-1}(X) = E$.

Proof. We describe an algorithm for the reverse transformation of $P^{-1}(\cdot)$. Define the next access time $N(x, i)$ of x at time i to be the minimum y coordinate of any point in X on the ray from (x, y) to (x, ∞). If there is no such point, $N(x, i) = \infty$. Let T_i be the treap defined on all points $(x, N(x, i))$.

We know that a treap is a binary tree where each node has a pair of values (k, p), where k is the key value satisfying binary search tree properties and p represents the priorities which satisfies heap property.

Let τ_i be the points in X with $y = i$. By the treap property of T_i, these must form a connected subtree of T_i that includes the root (because i is the minimum possible access time $N(\star, i)$). Now, form T_{i+1} by re-arranging the nodes in τ_i to form a treap based on the next access time $(x, N(x, i + 1))$. All we need to show is that T_{i+1} is a treap on $(x, N(x, i + 1))$. The BST property trivially

holds by construction, so we look at the heap property. It suffices to show that the heap property holds between every parent/child pair (q, r) in T_i. If both were in τ_i, the heap property follows by construction, and if both were outside τ_i, the heap property holds because neither their next access times nor their parent/child relationship changed from i to $i+1$. We are left with the case $q \in \tau_i$ and $r \notin \tau_i$. See Figure 2.

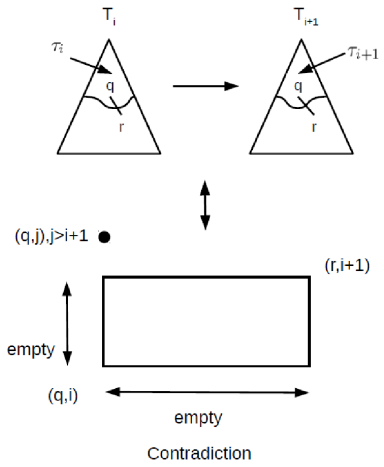

Figure 2. Proof of Lemma 2

For contradiction, suppose T_{i+1} was not a treap. Then $N(q, i+1) > N(r, i+1) = N(r, i)$. Hence vertical side incident on (q, i) in $\square qr$ will be empty (since q will be accessed at time strictly greater that $i+1$). Now we will show that horizontal side incident on (q, i) in $\square qr$ is also empty. Suppose not, then there exists a point (c, i) in this line, which implies that r is not a child of q in T_{i+1}, a contradiction (since the value of c is in between q and r, c can not be accessed in T_{i+1} with out accessing r). Contradiction. □

Let the geometric view of an access sequence S be the set of points $P(\mathrm{S}) = \{(s_1, 1), (s_2, 2), \ldots, (s_m, m)\}$. Lemmas 1 and 2 have shown that the arboral statement E executes S is equivalent to the geometric statement $P(\mathrm{S}) \subset P(E)$ Letting minASS(S) be the size of the smallest arborally satisfied superset of $P(\mathrm{S})$, we have $\mathrm{OPT}(\mathrm{S}) = \mathrm{minASS}(P(\mathrm{S}))$. Thus, Open Problem 1 is equivalent to designing algorithms for finding the minimum arborally satisfied superset.

Above, we gave a combinatorial equivalence of tree executions and arborally satisfied sets, effectively characterizing offline BST algorithms. In their paper they have also presented the correspondence between online BST algorithm and standard geometric representation. We will state the following lemma without proof.

Lemma 3. For any online ASS algorithm A, there exists an online BST algorithm A' such that, on any access sequence, the cost of $A'$is bounded by a constant times the cost of A.

3.2 Greedy Future, a candidate for dynamic optimality

Lucas[4] presented an offline BST algorithm, GREEDYFUTURE, which is as follows:

Algorithm 1. *The* GREEDYFUTURE *algorithm follows two principles: (1) only touch the nodes on the search path, and (2) to re-arrange the search path, move the next item to be accessed as high as possible, and recurse.*

More formally, let τ_i be the search path for s_i in T_i. If the next search in the access sequence is in τ_i, make that node the root and recurse on both sides. If the next search is in a subtree hanging from the path τ_i, re-arrange the predecessor and successor from τ_i as the root and the right child of the root; then recurse on the parts of the tree that have yet to be specified.

Lucas conjectured that her algorithm gives a constant-factor approximation for OPT(S). More than a decade later, Munro[6] proposed the same algorithm independently. He conjectured that the cost of GREEDYFUTURE is OPT(S) + O(m). (This conjecture about additive optimality is made in our precise BST model, because other models differ by constant factors). The simple example of Figure 3 demonstrates a cost of OPT(S)+m/2.

Conjecture 1: On any access sequence S, GREEDYFUTURE has cost O(OPT(S)). More strongly, it has cost OPT(S) + O(m).

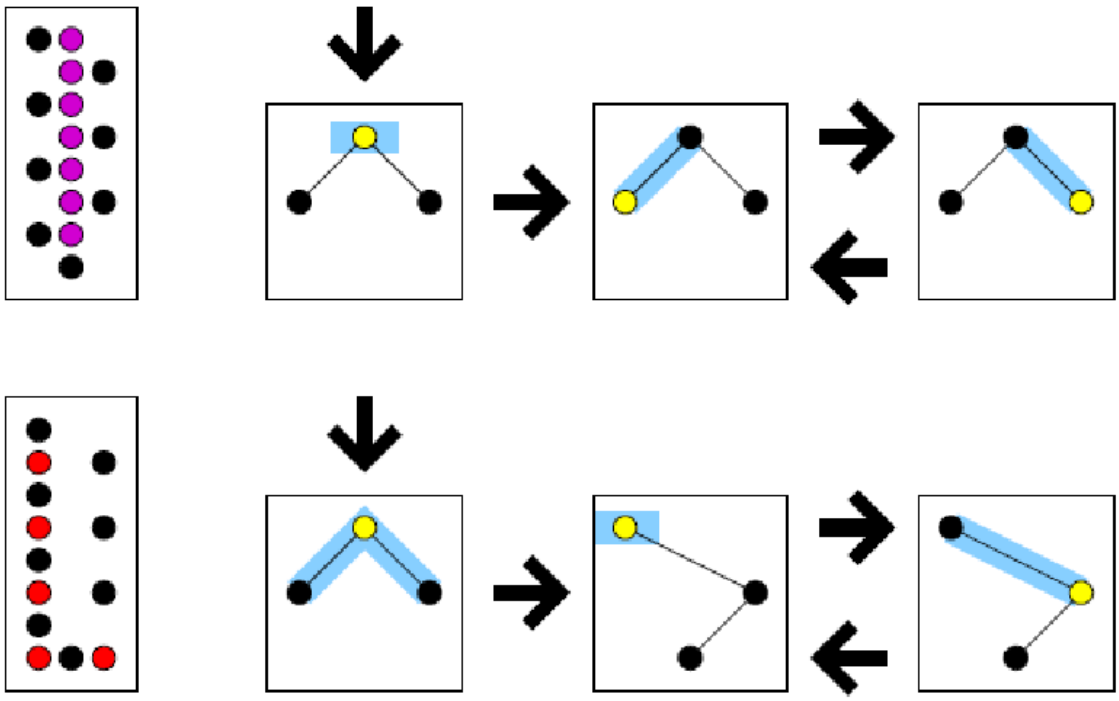

Figure 3: A simple three node example where the greedy algorithm performs suboptimally. In the figure greedy algorithm is on top and optimal algorithm is on bottom. Yellow node is to be searched. Access sequence are the alternate leaf nodes. Greedy algorithm will always cost 2 per access (as the future access is not in the search path). Whereas optimal algorithm will cost 3 for first access and modifies the BST such that it costs 3 per two access. So on average greedy algorithm costs 2, while optimal costs 1.5.

Looking at GREEDYFUTURE in our geometric view hides the ugly details of re-arranging the path, and transforms the algorithm into the following natural greedy algorithm for the ASS problem:

Algorithm 2. *Sweep the point set with a horizontal line by increasing y coordinate. At time i,* GREEDYASS *places the minimal number of points at $y = i$ to make the point set up to $y \leq i$ arborally satisfied. This minimal set of points is uniquely defined: for any unsatisfied rectangle formed with (s_i, i) in one corner, add the other corner at $y = i$.*

The key observation is that GREEDYASS is an online ASS algorithm, because its decisions depend only on the past (points at lower y coordinates). This means that the only important part of the algorithm is the greedy decision to only touch the search path (equivalently, to add the minimum number of points on the sweep line). Then, by Lemma 3, the online GREEDYASS can be turned back into an online BST algorithm with equal cost (up to a constant factor) as the original GREEDYFUTURE!

4 Lower Bounds

This section discusses lower bounds for BST algorithms in our geometric view. Equivalently, these are lower bounds on the size of the minimum satisfied superset, which an approximation algorithm can try to match. The main results are (1) a class of lower bounds that includes all known lower bounds ideas, in particular Wilber???s bounds(we will not prove this); and (2) a greedy algorithm, SIGNEDGREEDY, that gives the best lower bound in this class, up to a factor of two.

To simplify proofs, this section assumes that the access sequence S is a permutation, i.e., $m = n$ and each item is searched exactly once. Geometrically, this restriction says that the point set $X = P(\mathrm{S})$ is in ???general position??? from an orthogonal perspective (no two points are orthogonally collinear).

4.1 Independent Rectangle Bounds

We call two rectangles $\square ab$ and $\square cd$ independent (in X) if the rectangles are not arborally satisfied and no corner of either rectangle is strictly inside the other rectangle; see Figure 4.

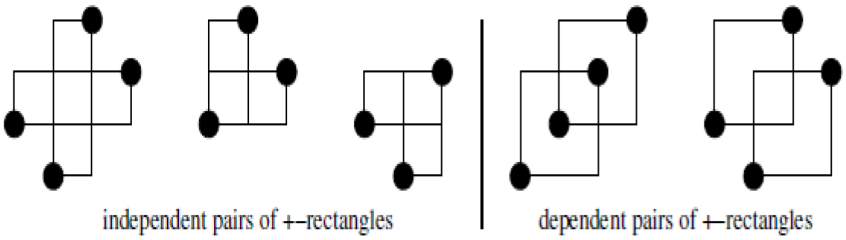

Figure 4: All combinatorial distict cases when overlapping ⊿- rectangles are indepedent/dependent

Now we will state the most important claim of this paper which we will prove later.

Claim 1. If a point set X contains an independent set I of rectangles, then $\mathrm{minASS}(X) \geq |I|/2 + |X|$. In particular, if $X = P(\mathrm{S})$ for an access sequence S, then $OPT(\mathrm{S}) \geq |I|/2 + |S|$.

Let maxIRB denote the maximum independent rectangle bound that can be formed on the point set X.

4.2 Approximating maxIRB

Observe from figure 4 that any rectangle can be defined by points at their 'southeast' and 'northwest' corners or 'southwest' and 'northeast' corners. Based on that we say that a rectangle $\square ab$ defined by two points a, b is a $\boxslash$-rectangle ($\boxbslash$-rectangle) if the slope of the line $\overline{ab}$ is positive (negative). Below, all statements and definitions using $\boxslash$-rectangles have symmetric statements for $\boxbslash$-rectangles, which we do not explicitly state.

A point set is $\boxslash$-satisfied if every pair of points (a, b) that form a $\boxslash$-rectangle $\square ab$ is arborally satisfied; see Figure 1. In other words, $\boxbslash$-rectangles need not be satisfied for $\boxslash$-satisfaction. Let $minASS_{\boxslash}(X)$ be the size of the minimum $\boxslash$-satisfied superset of X. We propose the following greedy strategy for computing $minASS_{\boxslash}(X)$, which is nothing more than GREEDYASS that ignores $\boxbslash$-rectangles:

Algorithm 3 *Sweep the point set X with a horizontal line by increasing y coordinate. When considering point p on the sweep line, for each unsatisfied $\boxslash$-rectangle formed by x and a point below the sweep line, add the rectangle???s northwest corner on the sweep line to make it satisfied. Let $add_{\boxslash}(X)$ be the final set of added points (excluding X).*

We do not focus in computing $minASS_{\boxslash}(\cdot)$ instead we focus on the following connection to independent rectangle bounds:

Lemma 4. *For any X, there exists an independent set of $\boxslash$-rectangles $IRB_{\boxslash}(X)$ with $|IRB_{\boxslash}(X)| = |add_{\boxslash}(X)|$.*

Proof. Follows from the Algorithm 3 (Whenever Algorithm adds a point it satisfies only one unique unsatisfied $\boxslash$-rectangle).

Now we will state a Lemma which we will prove later.

Lemma 5. Given an independent set I of $\boxslash$-rectangles in a point set X, any $\boxslash$-satisfied superset Y of X must have cardinality at least $|I| + |X|$.

Since $minASS(X)$ is also $\boxslash$-satisfied superset of X. Therefore from Lemma 5, $IRB_{\boxslash}(X) + |X| \leq minASS(X)$, and from Lemma 4 we have, $add_{\boxslash}(X)+|X| \leq minASS(X)$. Similarly $add_{\boxbslash}(X)+|X|$ is also the lower bound for $minASS(X)$. Now since $minASS(X)$ is equivalent to OPT(S) where S= $P(X)$. It follows that these are the lower bounds of OPT(S) as well. We define SIGNEDGREEDY by the obvious strategy: run both versions of Algorithm 3 and output $max\{|add_{\boxslash}(X)|, |add_{\boxbslash}(X)|\}+ |X|$. Thus we obtain the following fact:

$$\text{SIGNEDGREEDY} \leq OPT(\text{S}) \leq \text{GREEDYASS}$$

We are now going to show that the lower bound output by SIGNEDGREEDY is at least $\frac{1}{4}maxIRB(X)+ \frac{1}{2}|X|$, making it within a constant factor of the best independent rectangle bound. Now we will define the following notion of $\boxtimes$-satisfied set:

Definition 4. A superset Z of X is $\boxtimes$-satisfied with respect to X if there exist subsets $Z_{\boxslash}$ and $Z_{\boxbslash}$ of Z such that $Z_{\boxslash} \cup X$ is $\boxslash$-satisfied and $Z_{\boxbslash} \cup X$ is $\boxbslash$-satisfied. Let $minASS_{\boxtimes}(X)$ be the size of

the smallest $\boxtimes$-satisfied set Z with respect to X.

Note that $Z_{\boxslash}$ and $Z_{\boxbslash}$ need not be disjoint; in fact, to minimize $|Z|$ the two parts might overlap significantly. On the other hand, any $\boxslash$-satisfied superset can be combined with any $\boxbslash$-satisfied superset to give a $\boxtimes$-satisfied superset, so,

$$minASS_{\boxtimes}(X) \leq minASS_{\boxslash}(X) + minASS_{\boxbslash}(X) \tag{1}$$

Furthermore, any arborally satisfied set Y is also $\boxtimes$-satisfied (with $Z_{\boxslash} = Z_{\boxbslash} = Y$), so,

$$minASS_{\boxtimes}(X) \leq minASS(X) \tag{2}$$

Thus to prove **Claim 1** it is sufficient to prove the following:

Theorem 1. If X contains an independent set I of rectangles, then $minASS_{\boxtimes}(X) \geq |I|/2 + |X|$.

We will prove this theorem later.

Now we will show that output of SIGNEDGREEDY is actually a constant factor approximation of maxIRB.

$$\begin{aligned}
&\frac{1}{2}max\{WilberI(X), WilberII(X)\} + |X| \\
&\leq \frac{1}{2}maxIRB(X) + |X| && \text{[IRB bound is atleast Wilber's bounds (proof skipped)]} \\
&\leq minASS_{\boxtimes}(X) && \text{[by Theorem 1]} \\
&\leq minASS_{\boxslash}(X) + minASS_{\boxbslash}(X) && \text{[by equation 1]} \\
&\leq |add_{\boxslash}(X)| + |add_{\boxbslash}(X) + 2|x| && \text{[by defintion of } minASS_{\boxslash}\text{]} \\
&= |IRB_{\boxslash}(X)| + |IRB_{\boxbslash}(X)| + 2|X| && \text{[by Lemma 4]} \\
&\leq 2maxIRB(X) + 2|X| && \text{[by definition of maxIRB]} \\
&\leq 2maxIRB(X) + 4|X| && [2 \leq 4] \\
&\leq 4minASS_{\boxtimes}(X) && \text{[by Theorem 1]} \\
&\leq 4minASS(X) && \text{[by equation 2]}
\end{aligned}$$

Now observe that from the above deduction we have,

$$\frac{1}{2}maxIRB(X)+|X| \leq \text{SIGNEDGREEDY}+|X| \leq |add_{\boxslash}(X)|+|add_{\boxbslash}(X)|+2|x| \leq 2maxIRB(X)+4|X|$$

or,

$$\frac{1}{2}maxIRB(X) \leq \text{SIGNEDGREEDY} \leq 2maxIRB(X) + 4|X|$$

which is what we want to prove.

5 Proof of Theorem 1

Now we will prove some lemma which will help us to prove Theorem 1.

Lemma 6. *Suppose we are given a ◿-satisfied point set Y with integer x coordinates, a ◿-rectangle $\square ab$ with $a, b \in Y$, and a vertical line l at a non-integer x coordinate strictly between $a.x$ and $b.x$. Then we can find two points $p, q \in Y$ in the rectangle $\square ab$ such that $p.y = q.y$, p is left of l, q is right of l, and there are no points in Y on the horizontal segment connecting p to q.*

Proof. Let $p \in Y$ be the topmost rightmost point in $\square ab$ to the left of l. (Such a point exists because a is a candidate.) Let $q \in Y$ be the bottommost leftmost point in $\square ab$ to the right of l and at or above p. (Such a point exists because b is a candidate.) By construction, p is left of l, q is right of l, and there are no points in $\square ab$ strictly between p and q in x coordinate. It remains to show that $p.y = q.y$. If $p.y \neq q.y$, then by construction $p.y < q.y$, making $\square pq$ an unsatisfied ◿-rectangle, contradicting that Y is ◿-satisfied.

By symmetry, Lemma 6 also holds for ◹-rectangles in ◹-satisfied sets.

Lemma 7. Given an independent set I of rectangles in a set X such that each point has a distinct integer x coordinate, there exists a rectangle $\square ab \in I$ and a vertical line l at a non-integer x coordinate strictly between $a.x$ and $b.x$ such that, inside $\square ab$, l does not intersect the interior of any rectangle from $I \backslash \{\square ab\}$.

Proof. We claim that $\square ab$ can be chosen to be any rectangle in I that does not intersect any wider rectangles in I, for example, the widest rectangle in I. Assume by symmetry that $\square ab$ is a ◿-rectangle, and that $a.x < b.x$. Any rectangle that intersects $\square ab$ in their interiors cannot have a corner interior to $\square ab$, by independence, so such a rectangle must either intersect both the left and right edges of $\square ab$ or intersect both the top and bottom edges of $\square ab$ (or both). But, because $\square ab$ is (locally) maximally wide, only the latter case is possible. Thus all rectangles interior-overlapping $\square ab$ do so in its entire y extent from bottom edge to top edge.

Now we decompose rectangles interior-overlapping $\square ab$ into three types: (1) those that have a as a corner, (2) those that have b as a corner, and (3) those that have neither a nor b as a corner. All type-1 rectangles must be strictly left of all type-2 rectangles (in horizontal projection): if two were to overlap in their interiors, the corner of the type-1 rectangle on the bottom edge of $\square ab$ other than a itself would be interior to the type-2 rectangle, violating independence, and by distinctness of x coordinates, they cannot overlap on their boundaries either. If a type-3 rectangle intersects a type-1 or type-2 rectangle, then the former rectangle must be narrower than the latter and cross its top and bottom sides: otherwise, by independence, the type-3 rectangle would pierce the left and right sides and then have to be wider than $\square ab$, a contradiction. See Figure 5.

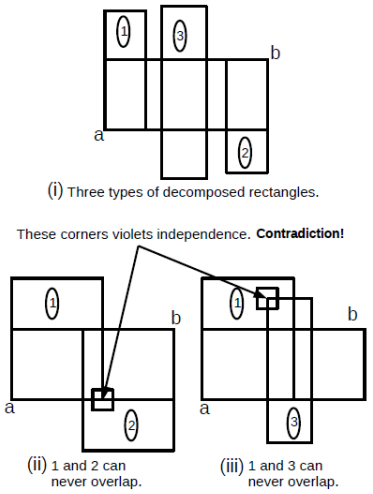

Figure 5.Proof of Lemma 7.

Thus we can decompose the horizontal span of $\square ab$ into a range starting at a containing type-1 rectangles and any overlapping type-3 rectangles, then a range of type-3 rectangles that intersect neither type-1 nor type-2 rectangles, followed by a range ending at b containing type-2 rectangles. (Any of these ranges may actually contain no rectangles.) By distinctness of x coordinates, there must be a positive-size gap between any adjacent pair of these ranges, and we can choose the line l to be in one of these gaps.

Proof of Lemma 5. We apply Lemma 7 to find a rectangle $\square ab$ in I and a vertical line l piercing $\square ab$ with the property that no other rectangle in I intersects l interior to $\square ab$. Then we apply Lemma 6 to find two points p, q horizontally adjacent in Y and on opposite sides of l in $\square ab$. We mark this pair (p, q) with rectangle $\square ab$. Then we remove $\square ab$ from I and repeat the process, until we have marked a pair of horizontally adjacent points in Y for every rectangle in I.

Whenever we remove a rectangle $\square ab$ from I, if p and q are not on the top or bottom sides of $\square ab$, then p and q do not simultaneously belong to any other rectangle in I, so they will never be marked again. On the other hand, if p and q are on the top (bottom) side of $\square ab$, then p and q are neither interior nor on the top (bottom) side of any other rectangle in I. Furthermore, because all rectangles in I are $\boxslash$-rectangles and coordinates in X are distinct, the top side of no rectangle in I coincides even partially with the bottom side of a rectangle in I. Thus, (1) each pair of horizontally adjacent points in Y can be marked at most once and (2) by distinctness of y coordinates in X, at most one point in a pair of horizontally adjacent points in Y can belong to X.

Suppose pair (p, q) was marked for $I_1 \in I$ and pair (q, r) for $I_2 \in I$. Clearly p, q, r are collinear in y-axis. Also $p, q \in I_1$ and $q, r \in I_2$, therefore q must belong to $Y \backslash X$ (otherwise I_1 and I_2 are not simultaneously independent). Also both p and r simultaneously does not belong to X (distinctness of x-coordinate). Thus one of them is from $Y \backslash X$, Without loss of generality, let $p \in Y \backslash X$. Then pick p as representative for I_1 and q for I_2. By the similar argument we can pick a unique point (taken from $Y \backslash X$) from each independent rectangle.

Therefore the number of points in $Y \backslash X$ is at least the number of rectangles in I, proving the lemma.

Proof of Theorem 1. Let Z be any $\boxtimes$-satisfied set with respect to X, where $Z_{\boxslash} \cup X$ is $\boxslash$-satisfied and $Z_{\boxbslash} \cup X$ is $\boxbslash$-satisfied. Let $I_{\boxslash}$ denote the $\boxslash$-rectangles in the independent set I, and similarly let $I_{\boxbslash}$ denote the $\boxbslash$-rectangles in I. By two applications of Lemma 5, we know that,

$|Z_{\boxslash} \cup X| \geq |I_{\boxslash}| \cup |X|$ and $|Z_{\boxbslash} \cup X| \geq |I_{\boxbslash}| \cup |X|$. Since, $|I| \leq |I_{\boxslash}| + |I_{\boxbslash}|$, and without loss of generality $|I_{\boxslash}| \geq |I_{\boxbslash}|$. So, $|I_{\boxslash}| \leq \frac{|I|}{2}$. Thus, $|Z| \geq |Z_{\boxslash} \cup X| \geq |I_{\boxslash}| + |X| \geq \frac{|I|}{2} + |X|$.

References

[1] Erik D. Demaine, Dion Harmon, John Iacono, Daniel Kane and Mihai Pătraşcu, The geometry of binary search trees, *In Proceedings of the 20th ACM-SIAM Symposium on Discrete Al gorithms (SODA) 2009* **28**:496–505, 2009.

[2] Donald E. Knuth, Optimum binary search trees. *Acta Informatica* **1**: 14-25, 1971.

[3] E. D. Demaine, D. Harmon, J. Iacono and M. Pătraşcu, Dynamic optimality−almost. *SIAM J. Comput.* **37(1)**: 240-251 (2007).

[4] J. M. Lucas, Canonical forms for competitive binary search tree algorithms. Tech. Rep. DCS-TR-250, Rutgers University, (1988).

[5] R. E. Wilber, Lower bounds for accessing binary search trees with rotations. *SIAM J. Comput.*, **18(1)**:56–67, (1989).

[6] J. I. Munro, On the competitiveness of linear search. *ESA*, :338–345, (2000).