Working with Genomic Intervals: Algorithms and Applications

Adam Taranto

COMP90014: Algorithms for Bioinformatics

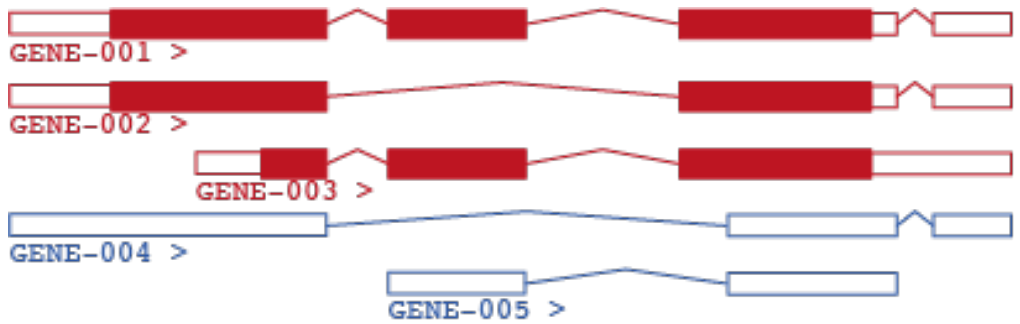
01/09/23

What is a genomic interval?

整个基因组内的特定范围

- A genomic interval is a specific range within a genome
- Belongs to a specific contig and has a start and end coordinate
- May be stranded
- Can contain metadata i.e gene names

每个基因组区间属于一个特定的连段，并有一个起始和结束的坐标。在这里，“连段”是指通过DNA测序拼接得到的一段没有间断的DNA序列。



可能是有向的（May be stranded）：指的是基因组区间可能具有方向性，

基因组区间可以包含例如基因名的元数据

- Common interval file types: BED, GFF, SAM

常见区间文件类型：提到了常用的几种文件格式，它们用于存储基因组区间的信息。BED（Browser Extensible Data）、GFF（General Feature Format）和SAM（Sequence Alignment/Map）都是在生物信息学中常用的文件格式，用于记录基因组区间和相关特征。

Working with intervals

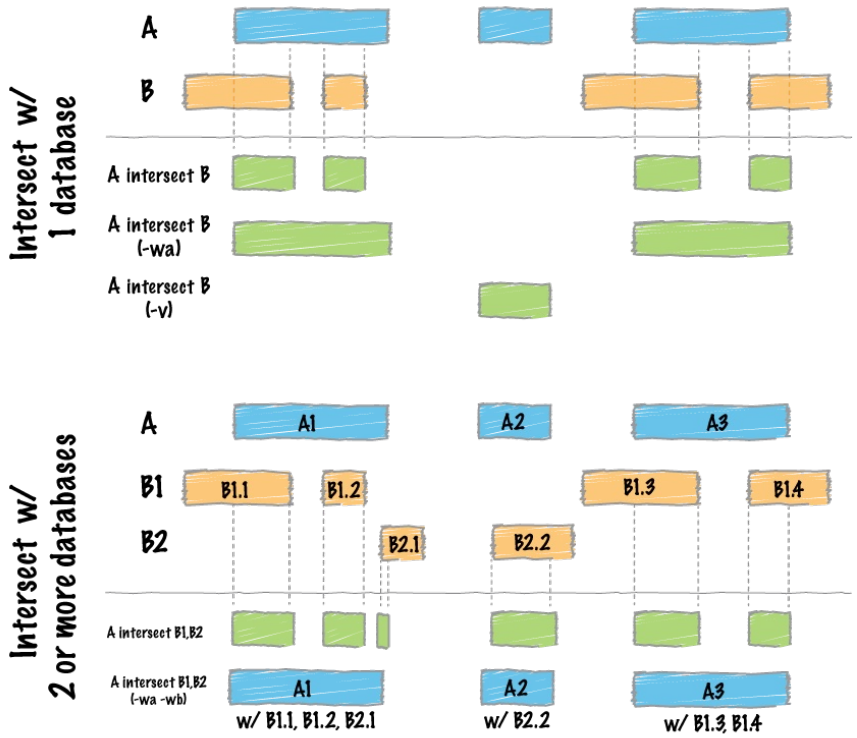
如何在生物信息学中处理基因组区间的数据

是否存在特定的区间？
在某个范围内存在哪些区间？
一个区间类型在何处与另一个重叠？
哪些基因在它们的启动子区域有特定的基序？
两个序列之间最长的不重叠对齐集是什么？
计算阅读深度（read depth，即某一区间被测序读段覆盖的次数）

- What kind of queries do we want to perform?
  - Does a specific interval exist?
  - What intervals exist in range?
  - Where does one interval type overlap another?
  - Which genes have a specific motif in their promoter region?
  - What is the longest non-overlapping set of alignments between two sequences?
  - Calculating read depth
- Why is this non-trivial with big datasets?

为何在大型数据集上进行此类操作并非微不足道？：当数据集规模变大时，进行这些查询和操作需要更复杂的数据结构和算法来保证效率和准确性。

- Note: Many common interval queries /manipulations are implemented in the bedtools package
- Tab delimited data types (sam, gff, bed, vcf) can be indexed with Tabix from htslib for fast lookup.



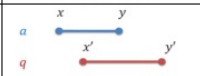
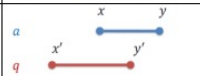
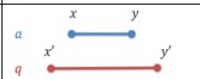
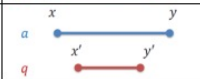
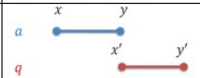
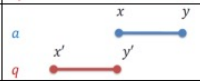
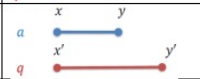
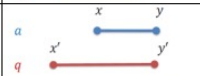
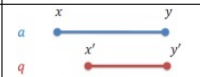
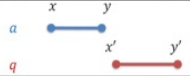
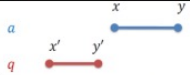
Finding intersections with bedtools

A 和 B 是两个不同的区间集合。
A intersect B 展示了A和B的交集区域。
A intersect B (-wa) 展示了A中与B重叠的部分。
A intersect B (-v) 展示了A中与B不重叠的部分。

Working with intervals

- There are many possible types of relationship between two intervals.
- When datasets are large we want to avoid comparing all pairs of intervals.

Table 1 Allen's interval relations and their transformation to the bound range for range query. Note: $a = [x, y]$ is an interval from the data interval set, $q = [x', y']$ is the query interval.

Symbols	Relation	Illustration	Definition	Rewriting as range query
o	$a\ o\ q$		$x < x' < y < y'$	$0 < x < x'$ $x' < y < y'$
oi	$a\ oi\ q$		$x' < x < y' < y$	$x' < x < y'$ $y' < y < \infty$
d	$a\ d\ q$		$x' < x < y < y'$	$x' < x < y'$ $x' < y < y'$
di	$a\ di\ q$		$x < x' < y' < y$	$0 < x < x'$ $y' < y < \infty$
m	$a\ m\ q$		$x < y = x' < y'$	$0 < x < x'$ $y = x'$
mi	$a\ mi\ q$		$x' < y' = x < y$	$x = y'$ $y' < y < \infty$
s	$a\ s\ q$		$x' = x < y < y'$	$x = x'$ $x' < y < y'$
si	$a\ si\ q$		$x = x' < y' < y$	$x = x'$ $y' < y < \infty$
f	$a\ f\ q$		$x' < x < y = y'$	$x' < x < y'$ $y = y'$
fi	$a\ fi\ q$		$x < x' < y' = y$	$0 < x < x'$ $y = y'$
$<$	$a < q$		$x < y < x' < y'$	$0 < x < x'$ $0 < y < x'$
$>$	$a > q$		$x' < y' < x < y$	$y' < x < \infty$ $y' < y < \infty$
$=$	$a = q$		$x = x' < y' = y$	$x = x'$ $y = y'$

Today’s Algorithms

- Binary search trees
  - Does a point exist in a dataset?
- Interval trees
  - Extended from BST
  - Supports interval queries
    - i.e. Does any interval annotations fall within a given range?

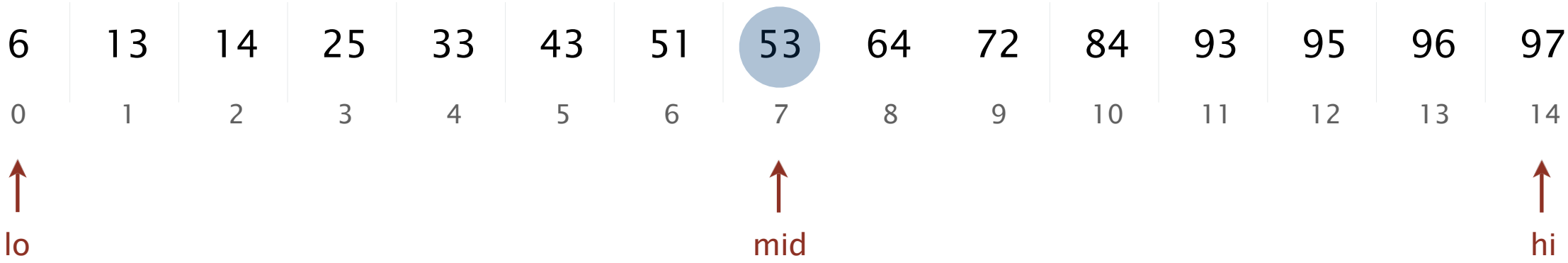
区间树（Interval trees）：

是从二叉搜索树扩展来的。区间树是一种特殊类型的二叉搜索树，用于存储区间并允许快速查找所有与给定区间重叠的区间。
支持区间查询。这意味着可以有效地查询数据集中的任何区间注释是否落在给定的范围内。区间树对于需要快速查找和管理许多有重叠的范围数据（如生物信息学中的基因组区间）的应用程序特别有用

Binary search

- Goal: Find the position of a key in a sorted array
- Binary Search
  - Split array at middle index
  - If midpoint value == key, done!
  - Too small, go left
  - Too big, go right

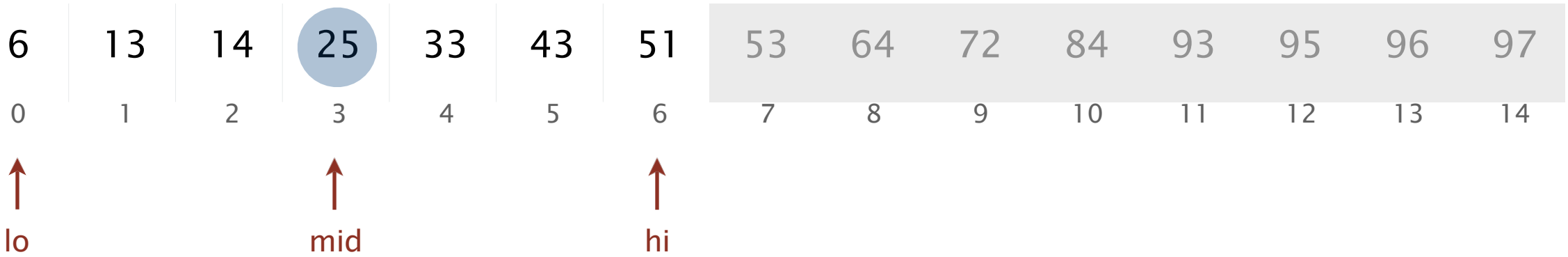
Search for 33



Example from: Algorithms 4th Ed. Sedgewick and Wayne

Binary search

- Goal: Find the position of a key in a sorted array
- Binary Search
  - Split array at middle index
  - If midpoint value == key, done!
  - Too small, go left
  - Too big, go right



Example from: Algorithms 4th Ed. Sedgewick and Wayne

Binary search

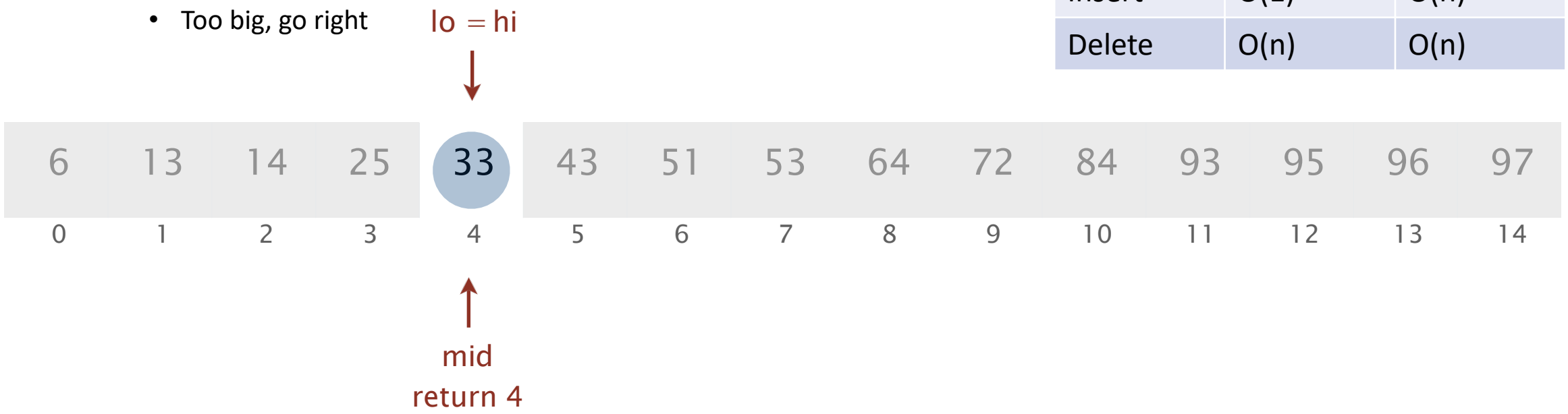
- Goal: Find the position of a key in a sorted array
- Binary Search
  - Split array at middle index
  - If midpoint value == key, done!
  - Too small, go left
  - Too big, go right

Example from: Algorithms 4th Ed. Sedgewick and Wayne

Binary search

- Goal: Find the position of a key in a sorted array
- Binary Search
  - Split array at middle index
  - If midpoint value == key, done!
  - Too small, go left
  - Too big, go right

	Linear search	Binary Search
Search	O(n)	O(log n)
Insert	O(1)	O(n)
Delete	O(n)	O(n)

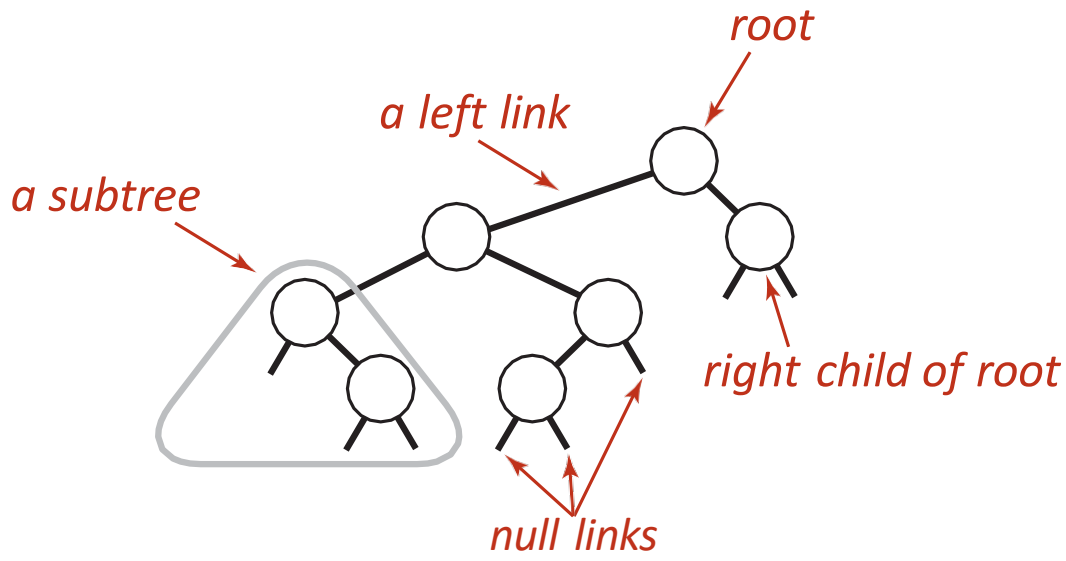


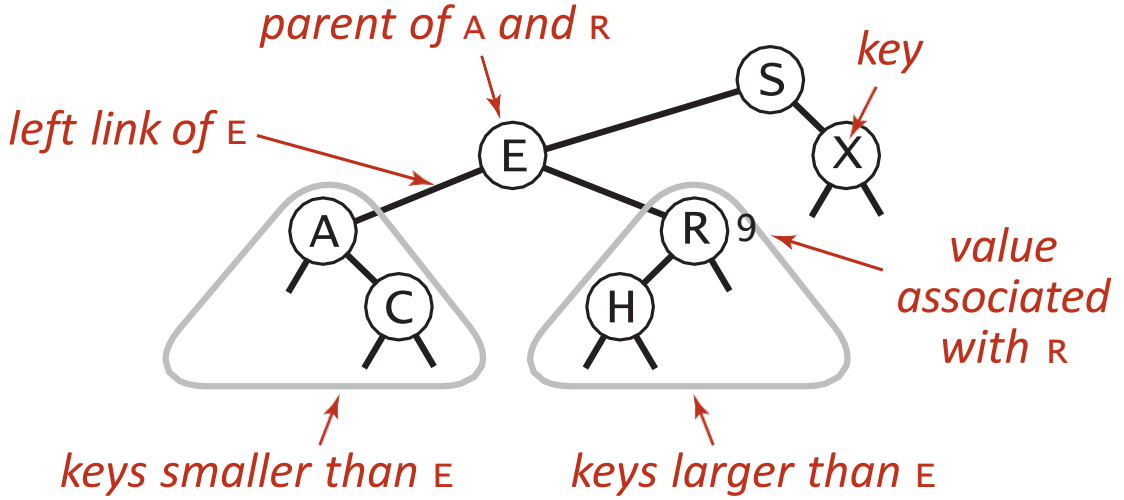
Example from: Algorithms 4th Ed. Sedgewick and Wayne

Binary Search Tree

- A BST is a binary tree in symmetric order

- Each node has a key which is:
  - Greater that all keys in its left subtree
  - Lesser than all keys in its right subtree

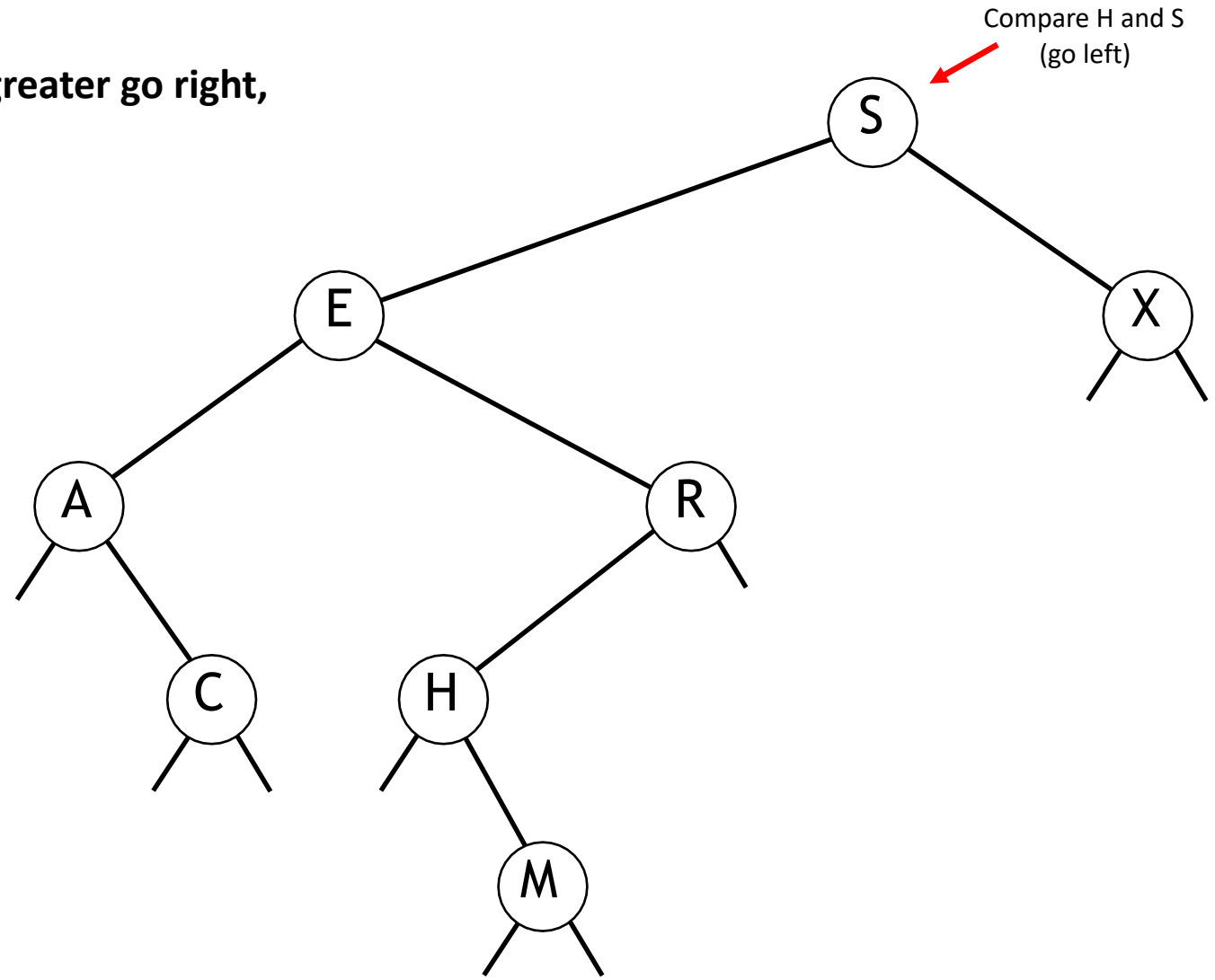




Binary Search Tree

Search: If query < node, go left; if greater go right, if equal, done!

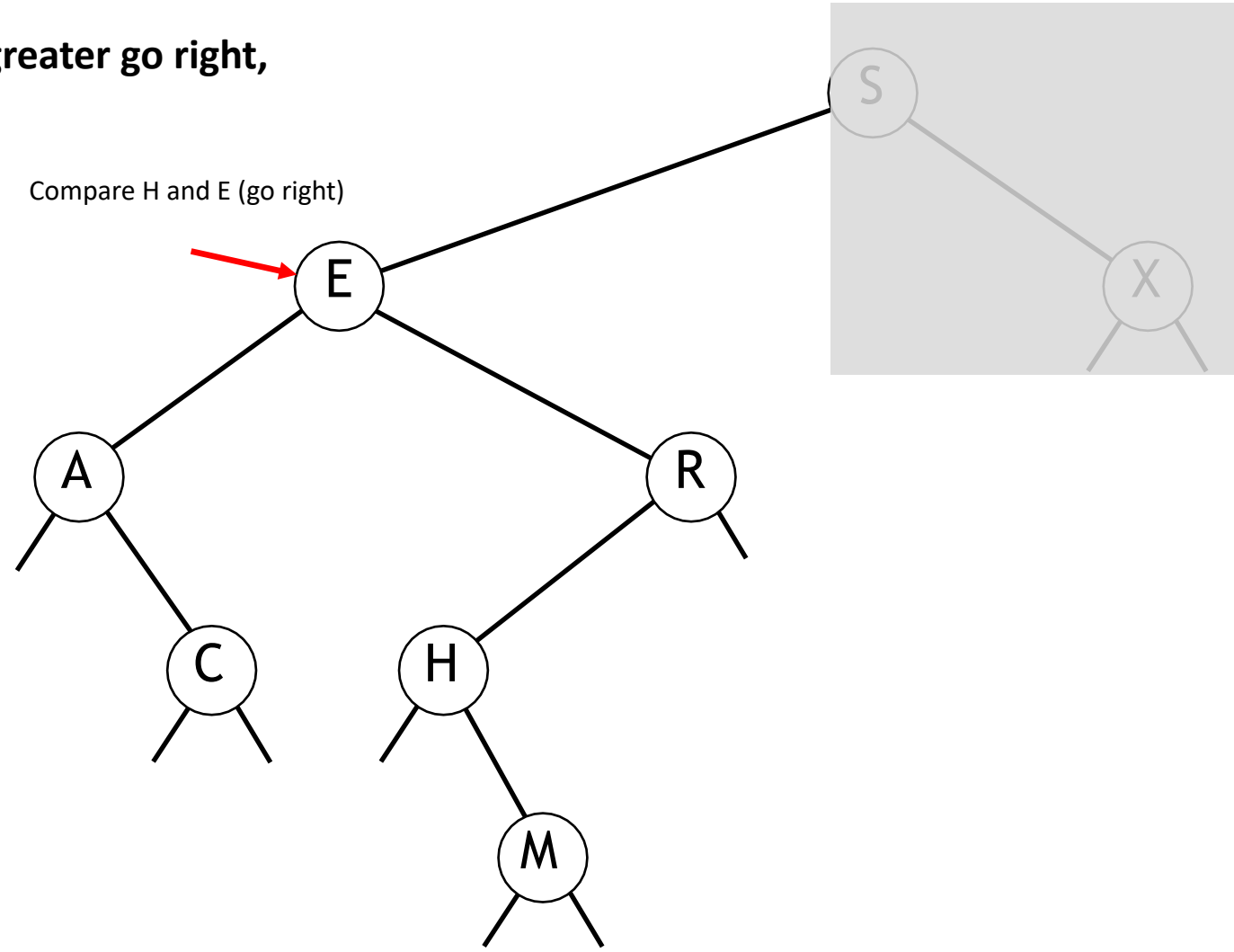
Search for H



Binary Search Tree

Search: If query < node, go left; if greater go right, if equal, done!

Search for H

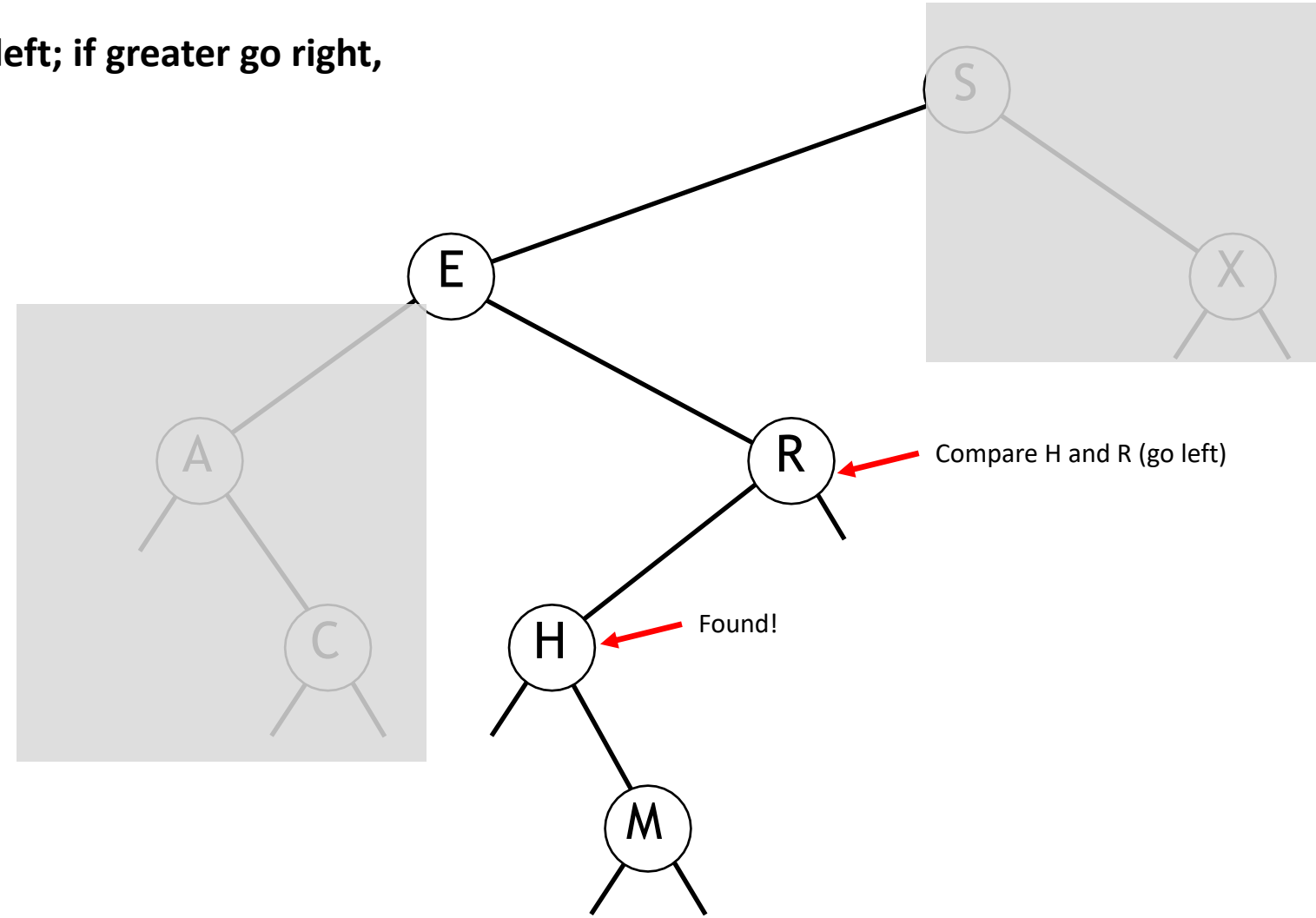


Binary Search Tree

Search: If query < node, go left; if greater go right, if equal, done!

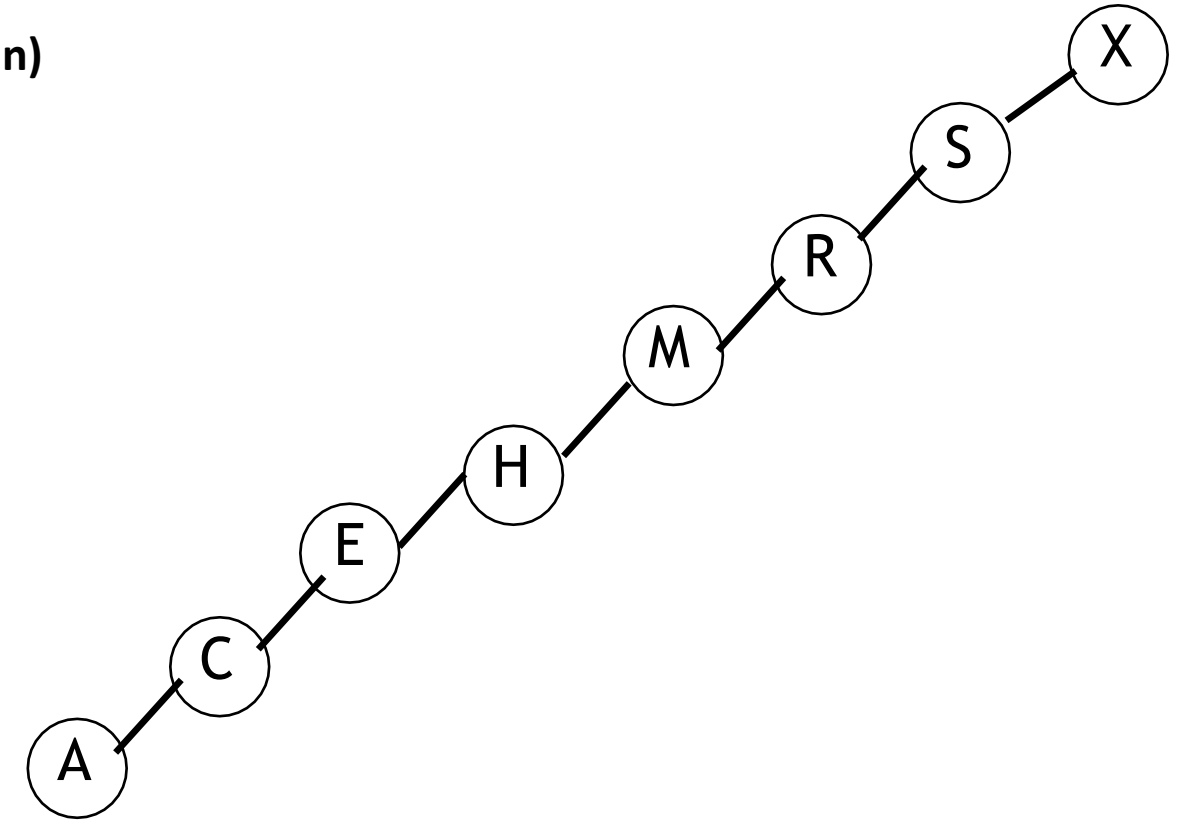
Search for H

Search space is halved at each step (if tree is balanced).



Binary Search Tree

If tree is unbalanced search can become O(n)



Binary Search Tree: Balancing

- Binary trees can be rebalanced after insertion/deletion using tree rotation.
- Tree rotation does not interfere with the element order
- In tree rotation one node is moved up and one down.
- This method reduces the height of a tree by moving smaller subtrees down the tree and larger subtrees up.

Binary Search Tree

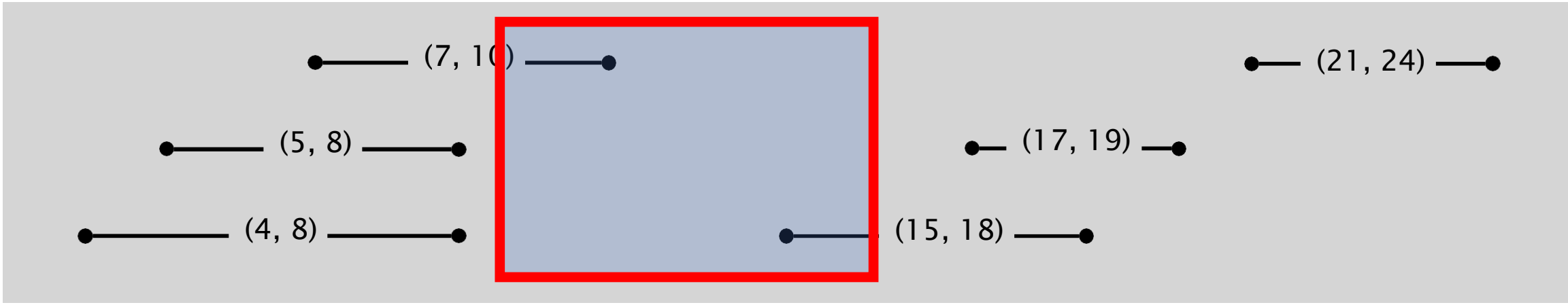
- Insertion
  - Similar to search
  - If less than current node, go left; if greater, go right; if null, insert.
- Deletion
  - If node has no children: delete
  - If node has one child, update parent node to link to child
  - If node has two children, find the maximum value in the **left** subtree and move that value to the location of the node being deleted.

	Binary Search	BST
Search	O(log n)	O(log n)
Insert	O(n)	O(log n)
Delete	O(n)	O(log n)

Interval Search Tree

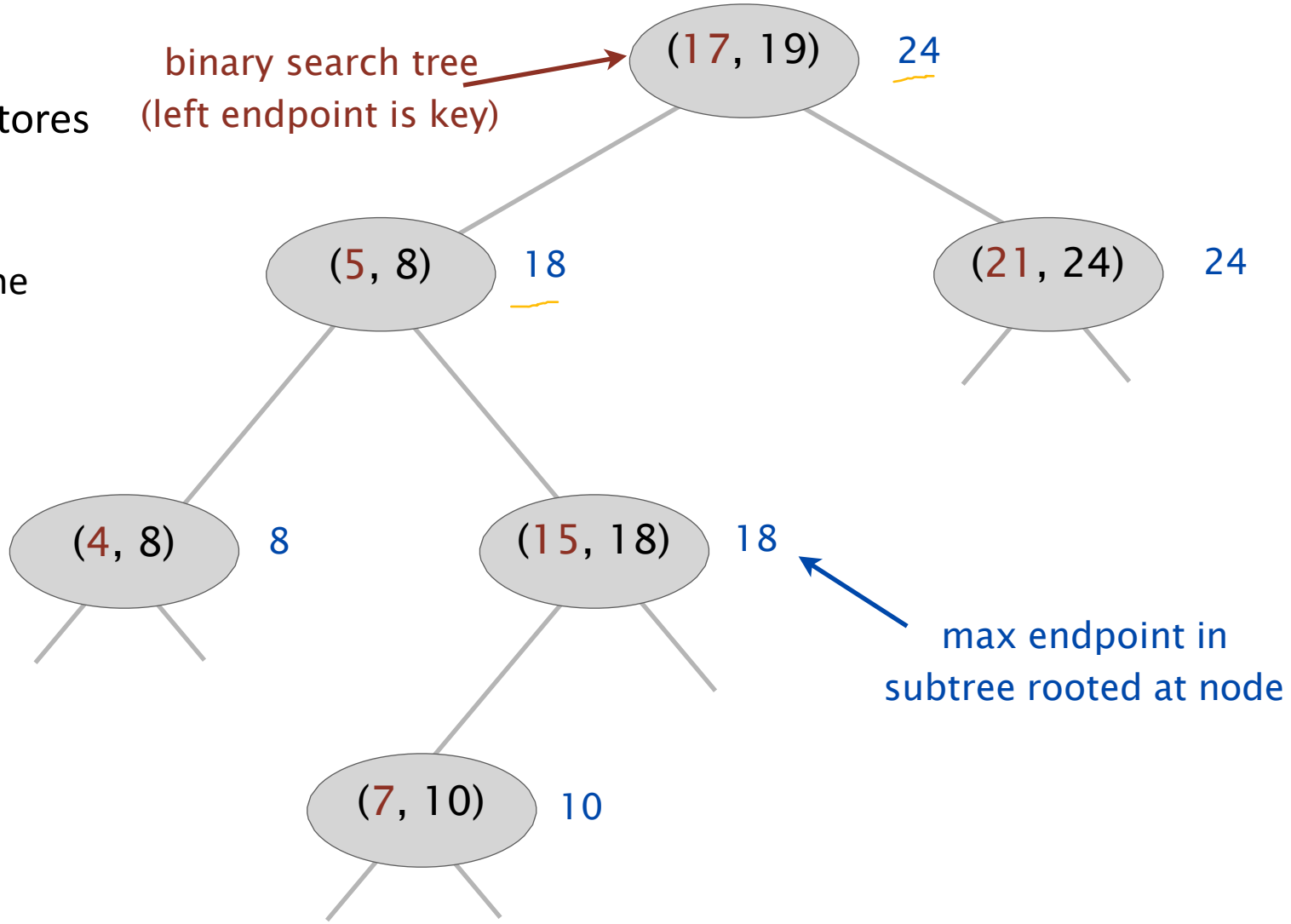
- Let's apply the BST concept to interval data
- For 1D interval data we want to perform an interval intersection query
  - Find all intervals that intersect a query interval

Find intervals that intersect the interval (9,16)



Interval Search Tree

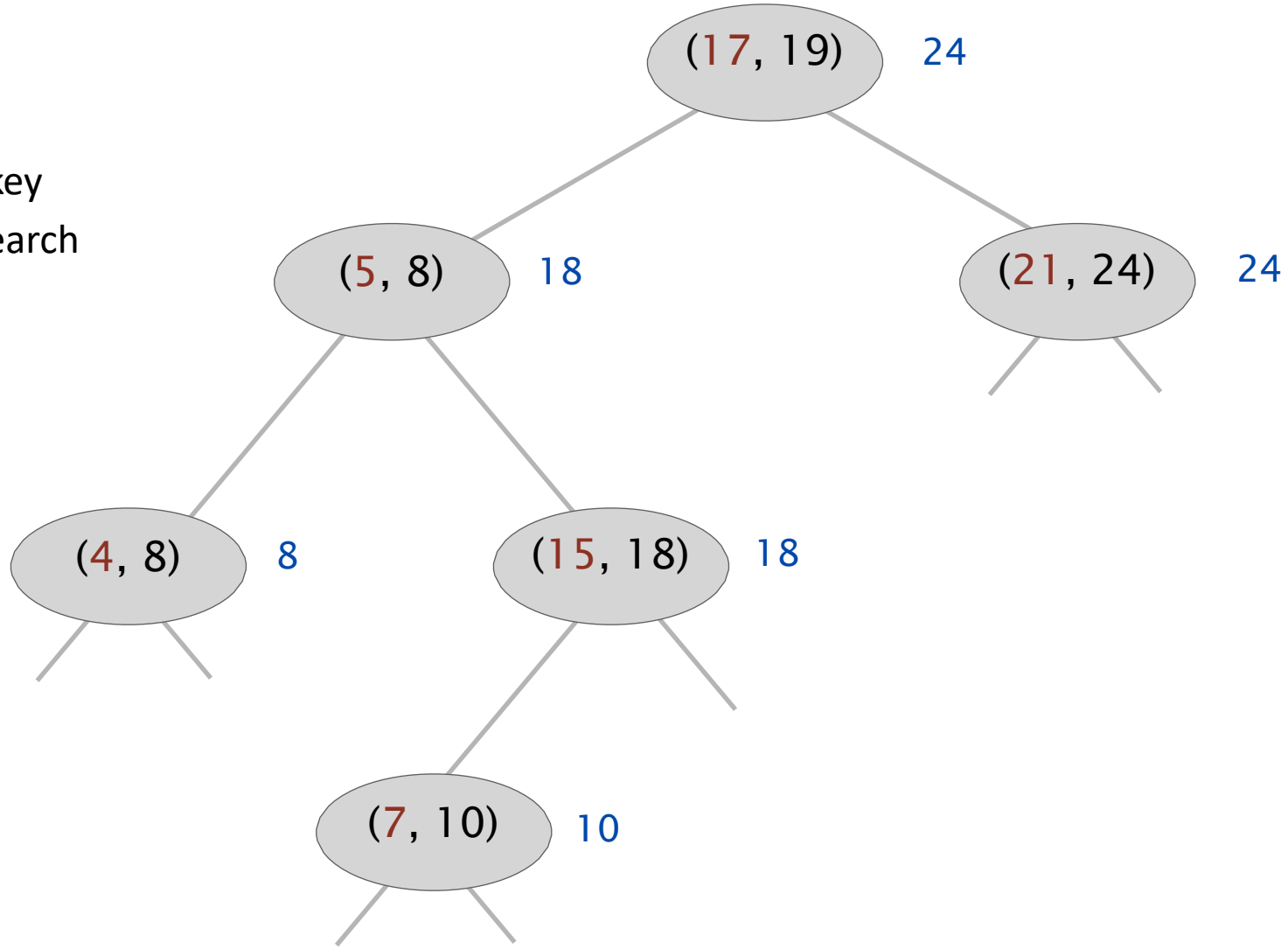
- Create a BST, where each node stores an interval *(lo,hi)*
  - Use left endpoint as BST key
  - Store the max endpoint from the subtree rotted at current node



Interval Search Tree: Insert

- Inserting an interval *(lo,hi):*
  - Insert into BST, using *lo* as the key
  - Update max in each node on search path

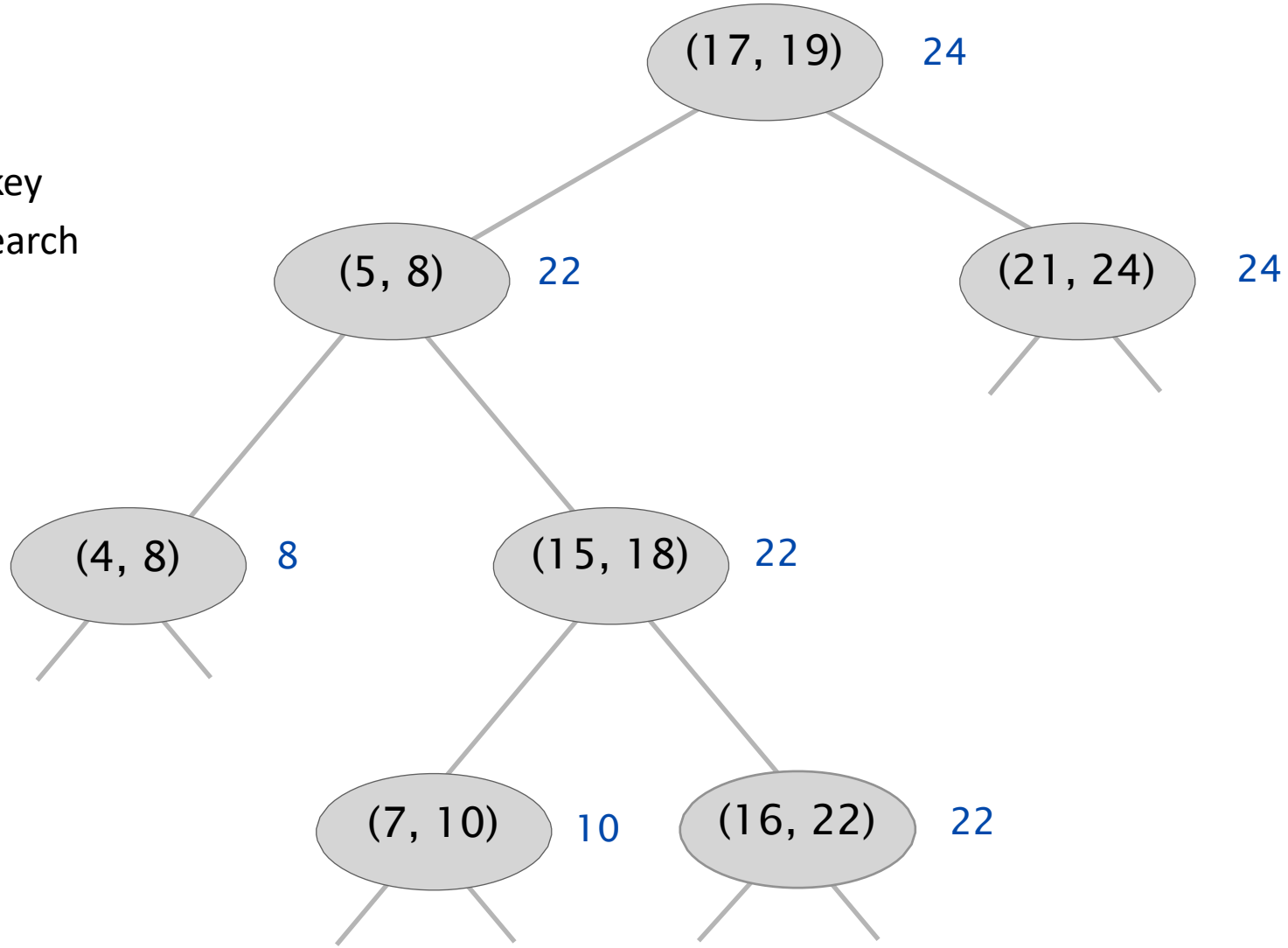
Insert interval (16, 22)



Interval Search Tree: Insert

- Inserting an interval *(lo,hi):*
  - Insert into BST, using *lo* as the key
  - Update max in each node on search path

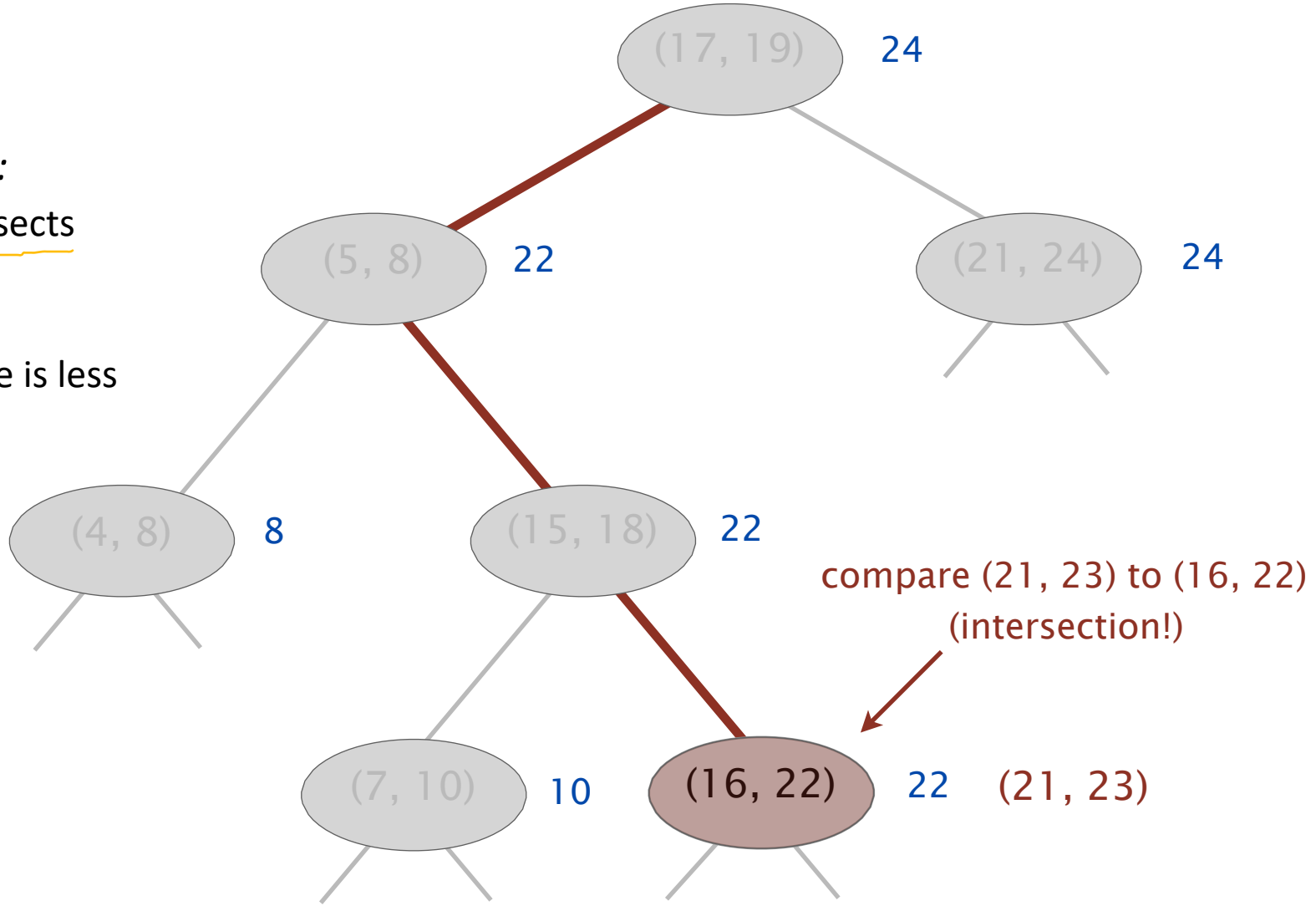
Insert interval (16, 22)



Interval Search Tree: Search

- Search for any one interval that intersects a query interval *(lo,hi):*
  - If interval in current node intersects query interval, return it.
  - Elif left subtree is null, go right.
  - Elif max endpoint in left subtree is less than *lo*, go right.
  - Else go left.

Find interval intersection for (21, 23)



Interval Search Tree: Search

To search for any one interval that intersects query interval $(lo, \quad hi)$:

- If interval in node intersects query interval, return it.
- Else if left subtree is null, go right.
- Else if max endpoint in left subtree is less than lo, go right.
- Else go left.

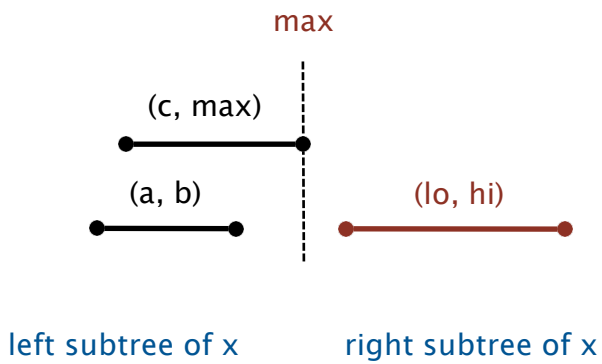
Case 1. If search goes right, then no intersection in left (and left in non-empty).

- Since went right, we have $max < lo$.
- For any interval (a, b) in left subtree of x, we have $b \leq max < lo$.

definition of max

reason for going right

- Thus, (a, b) will not intersect $(lo, \ hi)$.



Interval Search Tree: Search

To search for any one interval that intersects query interval (lo, hi):

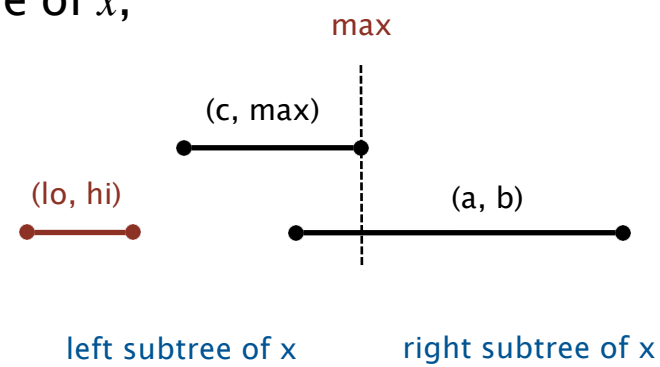
- If interval in node intersects query interval, return it.
- Else if left subtree is null, go right.
- Else if max endpoint in left subtree is less than lo, go right.
- Else go left.

Case 2. If search goes left, then there is either an intersection in left subtree or no intersections in either. Suppose no intersection in left.

- Since went left, we have $lo \leq max$.
- Then for any interval (a, b) in right subtree of x, $hi < c \leq a \Rightarrow$ no intersection in right.

no intersections in left subtree

intervals sorted by left endpoint



Interval Search Tree

*Assuming a balanced search tree

operation	brute	interval search tree	best in theory
insert interval	1	$\log N$	$\log N$
find interval	N	$\log N$	$\log N$
delete interval	N	$\log N$	$\log N$
find any one interval that intersects (lo, hi)	N	$\log N$	$\log N$
find all intervals that intersects (lo, hi)	N	$R \log N$	$R + \log N$

Summary

- Genomic annotations (intervals) may comprise massive datasets.
- Interval operations / queries are frequently used in bioinformatics.
- Avoid linear search for interval queries, instead:
  - Index intervals (tabix), or
  - Reduce search space with interval trees
- Search efficiency is dependent on binary trees being balanced