Binary Search Trees: Introduction

Daniel Kane

Department of Computer Science and Engineering University of California, San Diego

Data Structures Fundamentals Algorithms and Data Structures

Learning Objectives

- Provide examples of the sorts of problems we hope to solve with Binary Search Trees.
- Show why data structures that we have already covered are insufficient.

Outline

1 Local Search

2 Attempts

Dictionary Search

Find all words that start with some given string.

```
mke a dictator 2 overbearing of orially adv. [Latin: related adv. [Latin: related diction / diks(ə)n/n. manner ciation in speaking or singing dictio from dico dict-say]

dictionary / diksənəri/n. (p. book listing (usu. alphabetic explaining the words of a lar giving corresponding words it language. 2 reference book explaining the source.
```

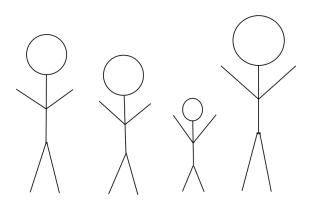
Date Ranges

Find all emails received in a given period.

box				
FROM	KNOW	то	SUBJECT	SENT TIME ▼
"lawiki.i2p admin" <j5uf></j5uf>		Bote User <uhod></uhod>	hi	Unknown
anonymous		Bote User <uh0d></uh0d>	Sanders 2016	Aug 30, 2015 3:27 PM
anonymous		Bote User <uh0d></uh0d>	I2PCon 2016	Aug 30, 2015 3:25 PM
Anon Developer <gvbm></gvbm>		Bote User <uhod></uhod>	Re: Bote changess	Aug 30, 2015 2:54 PM
I2P User <uuux></uuux>		Bote User <uh0d></uh0d>	Hello World!	Aug 30, 2015 2:51 PM

Closest Height

Find the person in your class whose height is closest to yours.



Local Search

Definition

A Local Search Datastructure stores a number of elements each with a key coming from an ordered set. It supports operations:

- RangeSearch(x, y): Returns all elements with keys between x and y.
- NearestNeighbors(z): Returns the element with keys on either side of z.

1	4	6	7	10	13	15
---	---	---	---	----	----	----

1 4 6 7 10 13 15

RangeSearch(5, 12)

1	4	6	7	10	13	15
---	---	---	---	----	----	----

1 4 6 7 10 13 15

RangeSearch(5, 12)

1 4 6 7 10 13 15

NearestNeighbors(3)

 1
 4
 6
 7
 10
 13
 15

Dynamic Data Structure

We would also like to be able to modify the data structure as we go.

- Insert(x): Adds a element with key x.
- Delete(x): Removes the element with key x.

1	4	6	7	10	13	15
---	---	---	---	----	----	----

1 4 6 7 10 13 15

Insert(3)

1 3 4 6	7	10	13	15
---------	---	----	----	----

1 4 6 7 10 13 15

Insert(3)

1 3 4 6 7 10 13 15

Delete(10)

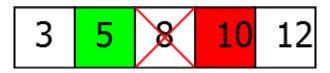
1 3 4 6 7 13 15

Problem

If an empty data structure is given these commands what does it output at the end?

- Insert(3)
- Insert(8)
- Insert(5)
- Insert(10)
- Delete(8)
- Insert(12)
- NearestNeighbors(7)

Answer



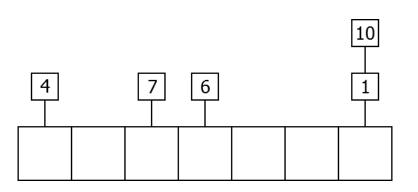
Outline

1 Local Search

2 Attempts

RangeSearch:

Impossible ×

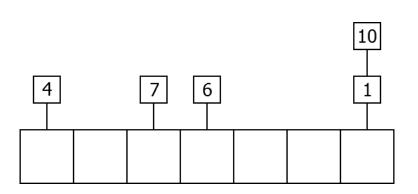


RangeSearch:

Impossible ×

NearestNeighbors:

Impossible ×

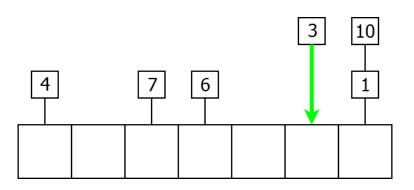


- RangeSearch:
- NearestNeighbors:
- Insert:

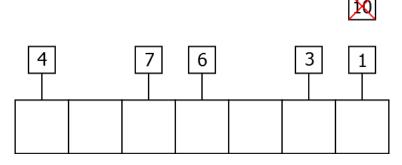
Impossible ×

Impossible ×

O(1) √



- RangeSearch: Impossible ×NearestNeighbors: Impossible ×
- Insert: O(1)
- Delete: O(1)



RangeSearch:

 $O(n) \times$

 7
 10
 4
 13
 1
 6
 15

RangeSearch:

 $O(n) \times$

■ NearestNeighbors:

 $O(n) \times$

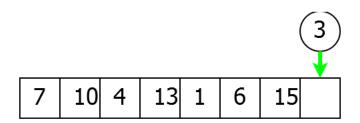
7 | 10 4 | 13 1 | 6 | 15

- RangeSearch:
- NearestNeighbors:
- Insert:







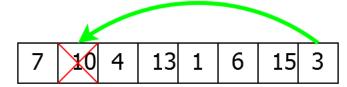


■ RangeSearch: $O(n) \times$

■ NearestNeighbors: $O(n) \times$

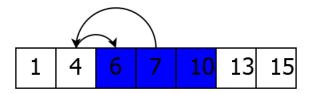
Insert: O(1)

■ Delete: O(1) ✓



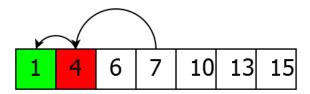
RangeSearch:

 $O(\log(n))$



 $O(\log(n)) \checkmark O(\log(n)) \checkmark$ RangeSearch:

NearestNeighbors:



RangeSearch:

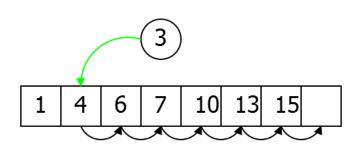
 $O(\log(n))$ \checkmark

NearestNeighbors:

 $O(\log(n))$

■ Insert:

 $O(n) \times$



RangeSearch:

 $O(\log(n))$

NearestNeighbors:

 $O(\log(n))$ \checkmark

Insert:

 $O(n) \times$

■ Delete:

 $O(n) \times$

1 3 4 6 7 10 13 15

RangeSearch:

$$O(n) \times$$

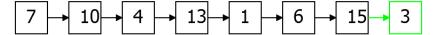


 $O(n) \times O(n) \times O(n) \times O(n)$ RangeSearch:

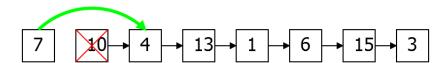
NearestNeighbors:



- RangeSearch:
- $O(n) \times$ NearestNeighbors:
- Insert:



- RangeSearch: $O(n) \times$
- NearestNeighbors: $O(n) \times$
- Insert: O(1) ✓
- Delete: O(1) ✓



Need Something New

Problem

Previous data structures won't work. We need something new.