Binary Search Trees and Vector-set

CS 151: Introduction to Computer Science I

Bits

Previous lecture:
One byte: a character or a small number

Bit: 0 or 1
One byte: 8 bits. An integer between 0 and 255

A representative picture of your computer's memory:

This is why binary occurs so frequently in computer science

How Vector Achieves Its Runtime

Each vector element is a fixed size (say 4 bits for this picture, but in reality it's 32 or 64 bits)


```
(vector-ref vec 5) ⟹ 1000
```

Why bother stepping through the first 5 elements? Just jump directly to index 5

But adding an element requires copying the whole vector...

vectors are called *arrays* in other programming languages, lists are called *linked lists* in other languages

vector-set!

You can also change the contents of a vector:

```
(: vec : (Vector Integer))
(define vec (vector 12 9 6))
(vector-set! vec 1 15)
(vector-set! vec 0 100)
(vector-set! vec 0 150)
```

Q: what type does `vector-set!` have?

```
(: vector-set! : (All (A) (Vectorof A) Integer A
-> Void))
```

It truly returns nothing

vector-set!

Inside a function, we can use `begin` to perform multiple updates in a row.

```
(begin (vector-set!  vec 0 150)
       (vector-set!  vec 1 100)
       (vector-ref!  vec 1))
```

The result of a `begin` expression is the last line
Make a function that swaps two elements in a vector

```
(:  swap!  :  (Vectorof Integer) Integer Integer ->
Void)
```

Swap

```
(: swap! : (Vectorof Integer) Integer Integer ->
Void)
(define (swap! vec i j)
  (local
    {(: tmp : Integer)
     (define tmp (vector-ref vec i))}
    (begin
           (vector-set! vec i (vector-ref vec j))
           (vector-set! vec j tmp))))
```

Runtime is $O(1)$

Reverse

Functions that just modify the input and don't return anything are said to be *in-place*.

```
(: reverse : (Vectorof Integer) -> Void)
```

See Piazza for code

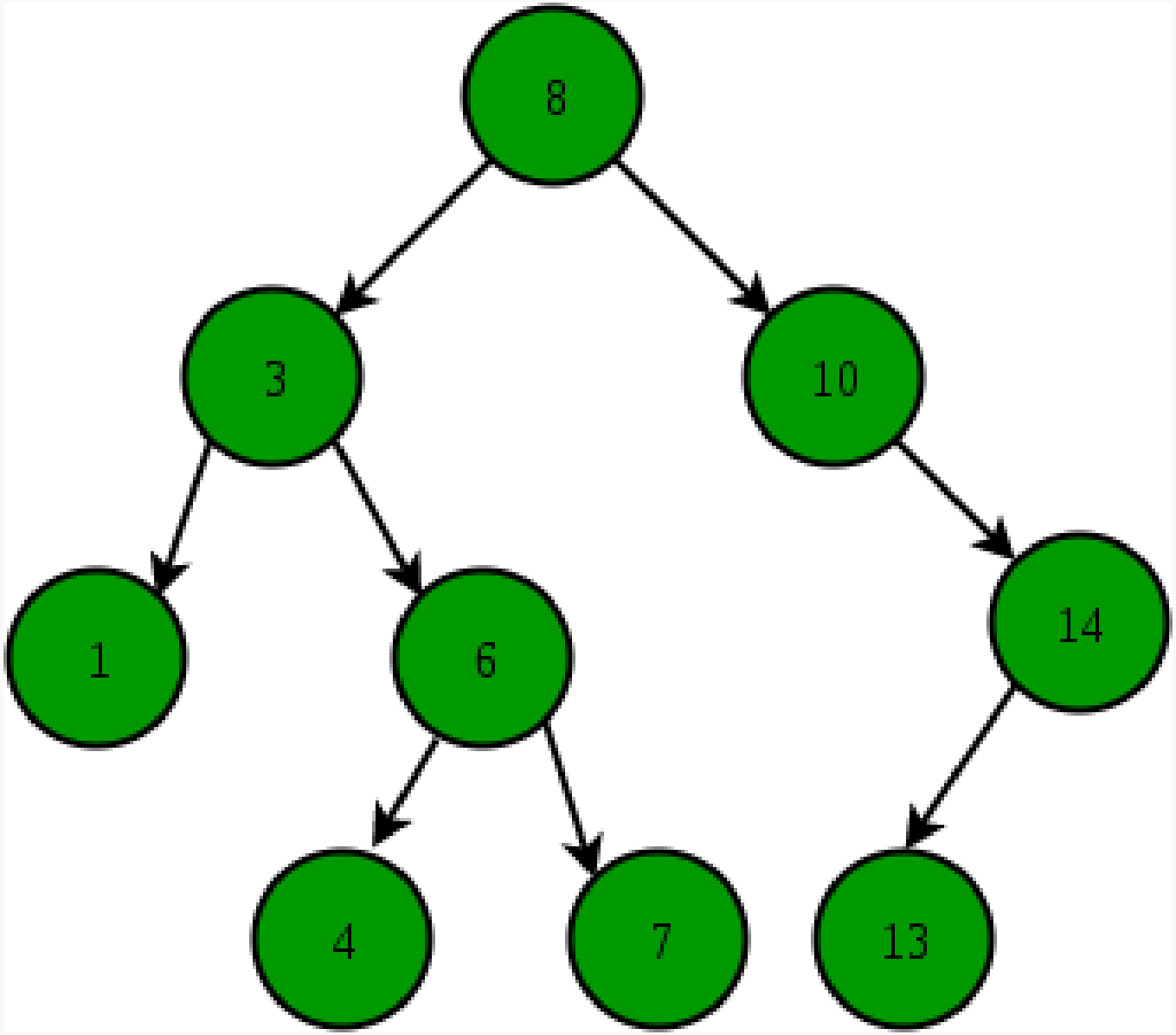
Runtime is $O(n)$

Binary Search Trees

Binary search tree is a binary satisfying the following properties

1. The left subtree of a node contains only nodes with keys lesser than the node's key.
2. The right subtree of a node contains only nodes with keys greater than the node's key.
3. The left and right subtree each must also be a binary search tree.

Binary Search Trees



Operations on Binary Search Trees

1. Search for a node
2. Add a node
3. Delete a node

All these operations must preserve the invariant of a bst.

What to know

- Imperative programming in Racket: vector-set!, begin, Void
- Binary search trees