بسم الله الرحمن الرحیم

ساختمان‌های داده

جلسه ۱۶

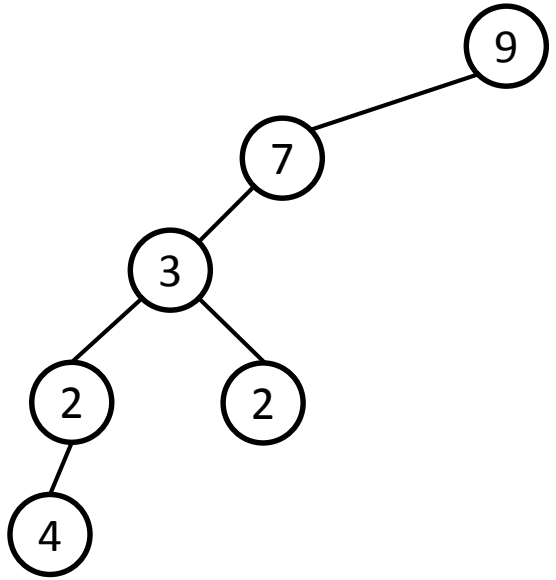
مجتبی خلیلی
دانشکده برق و کامپیوتر
دانشگاه صنعتی اصفهان

مثال

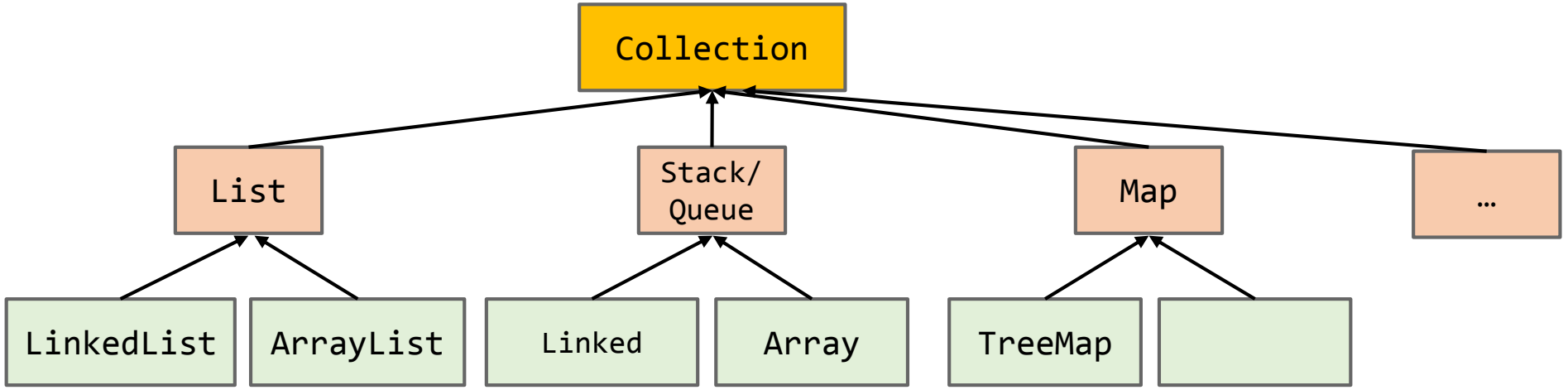
- کمترین ارتفاع ممکن برای درخت باینری با n گره از چه مرتبه ای است؟

مثال

- حافظه؟



ساختارهای داده



Binary Search Trees

Maps

- Map

Entry ADT

- An entry stores a key-value pair (k,v)
- Methods:
  - key(): return the associated key
  - value(): return the associated value
  - setKey(k): set the key to k
  - setValue(v): set the value to v

- We call this “item” or “element” or “record” exchangeably.

- Then, MAP stores multiple a collection of Entries

The Map ADT

- find(k): if the map M has an entry with key k, return and iterator to it; else, return special iterator end
- put(k, v): if there is no entry with key k, insert entry (k, v), and otherwise set its value to v. Return an iterator to the new/modified entry
- erase(k): if the map M has an entry with key k, remove it from M

- size(), empty()
- begin(), end(): return iterators to beginning and end of M

Ordered Maps

- Keys come from a total order
- New operations:
  - Each returns an iterator to an entry:
  - firstEntry(): smallest key in the map
  - lastEntry(): largest key in the map
  - floorEntry(k): largest key $\leq$ k
  - ceilingEntry(k): smallest key $\geq$ k
  - All return end if the map is empty

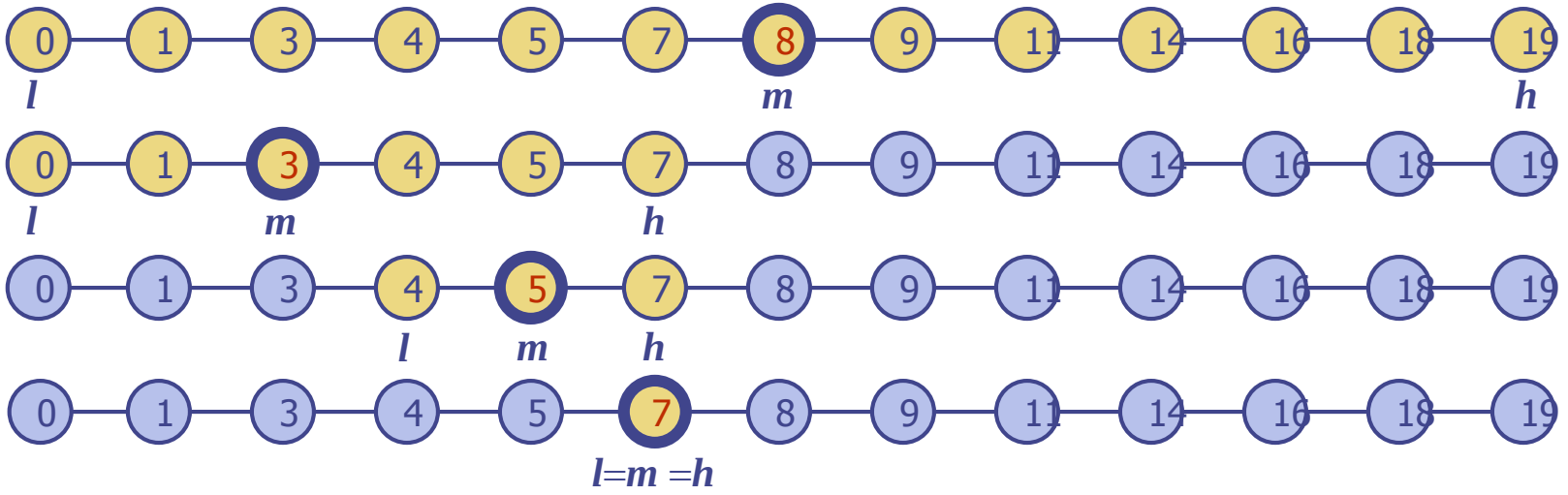
- Important Issue?

  Using what data structure and algorithm, we implement Ordered Map?

Binary Search

- Binary search can perform operations get, floorEntry and ceilingEntry on an ordered map implemented by means of an array-based sequence, sorted by key
  - similar to the high-low game
  - at each step, the number of candidate items is halved
  - terminates after O(log n) steps

- Example: find(7)



Search Tables

- A search table is an ordered map implemented by means of a sorted sequence
  - We store the items in an array-based sequence, sorted by key
  - We use an external comparator for the keys (for any arbitrary comparison)

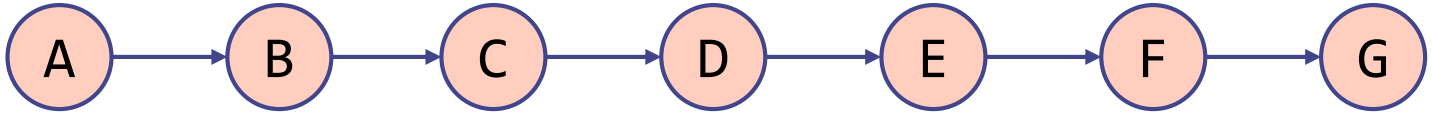
- Performance:
  - find, floorEntry and ceilingEntry take $O(\log n)$ time, using binary search
  - insert (put) takes $O(n)$ time since in the worst case we have to shift $n/2$ items to make room for the new item
  - erase (delete) take $O(n)$ time since in the worst case we have to shift $n/2$ items to compact the items after the removal

Binary Search Trees

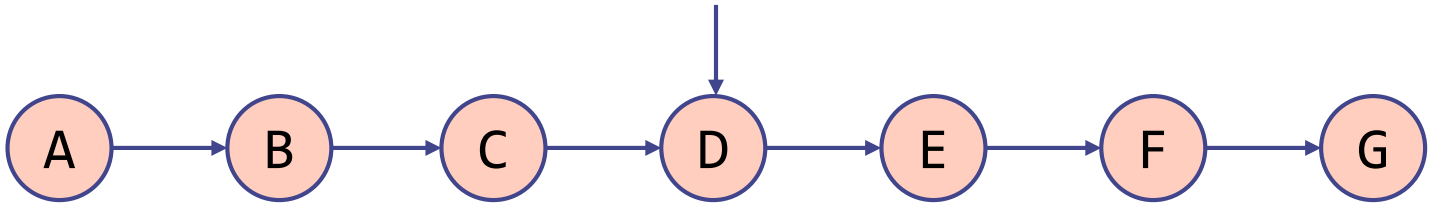
- Insert, find, delete?

Binary Search Tree

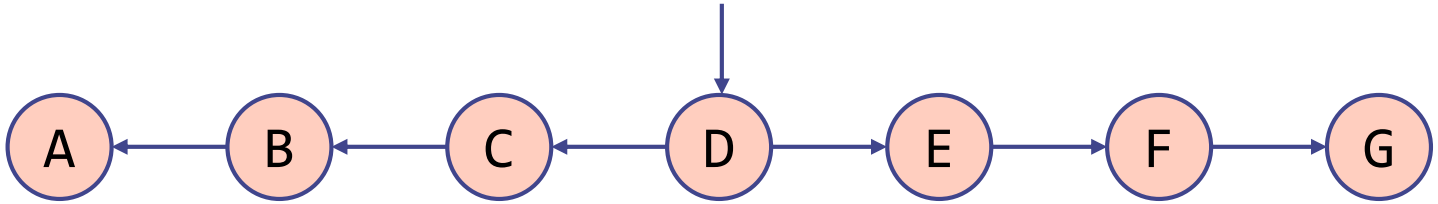
- مرتب شده



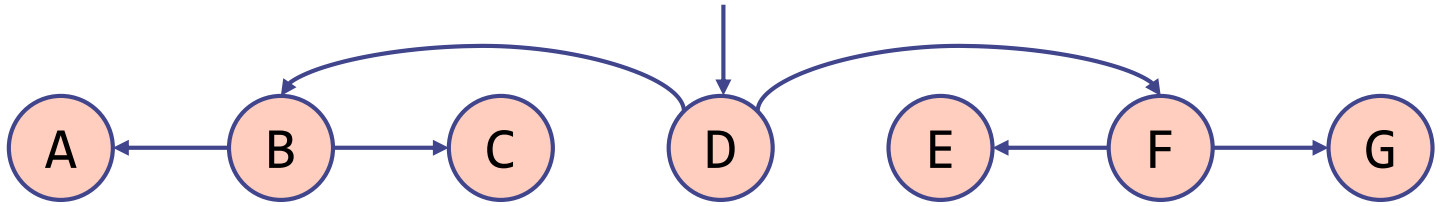
Binary Search Tree



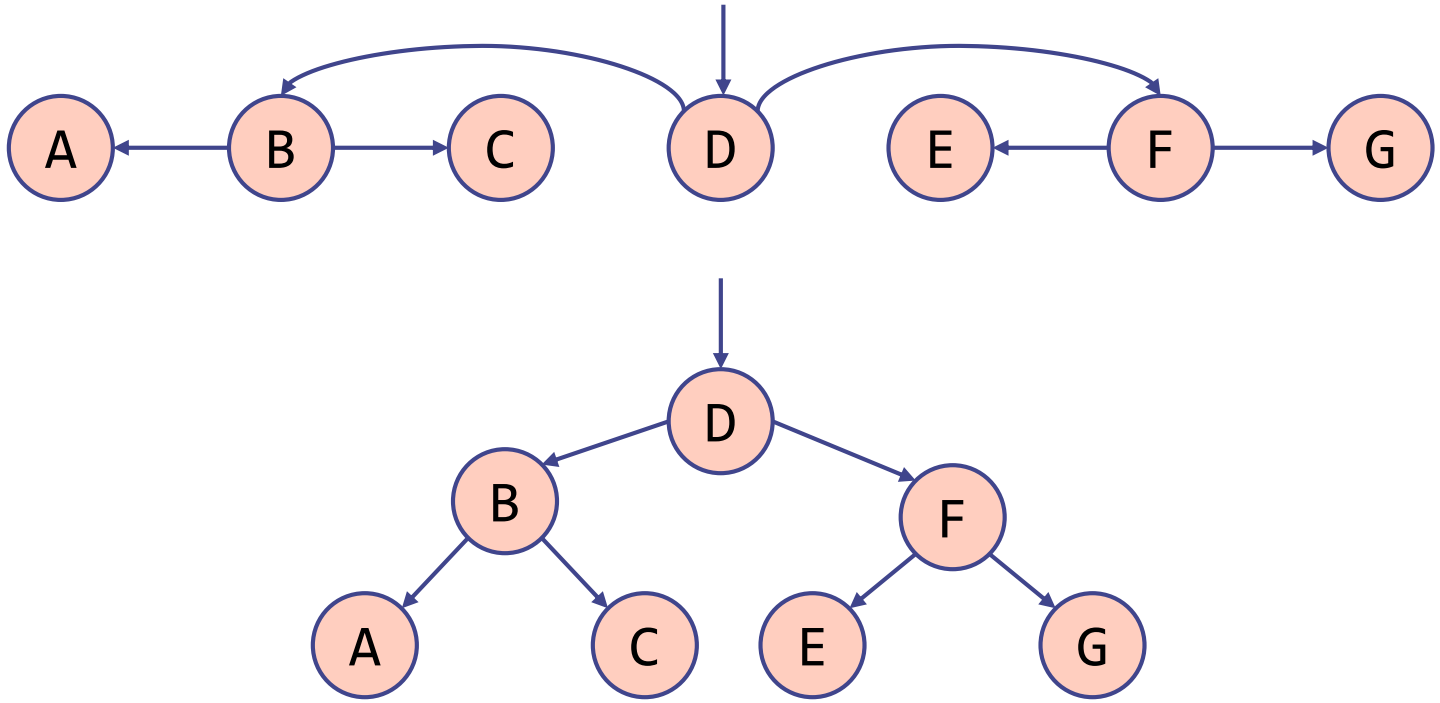
Binary Search Tree



Binary Search Tree

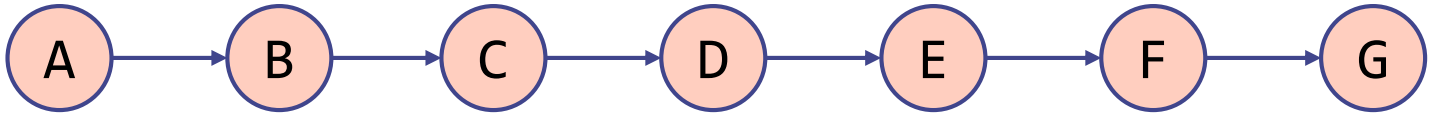


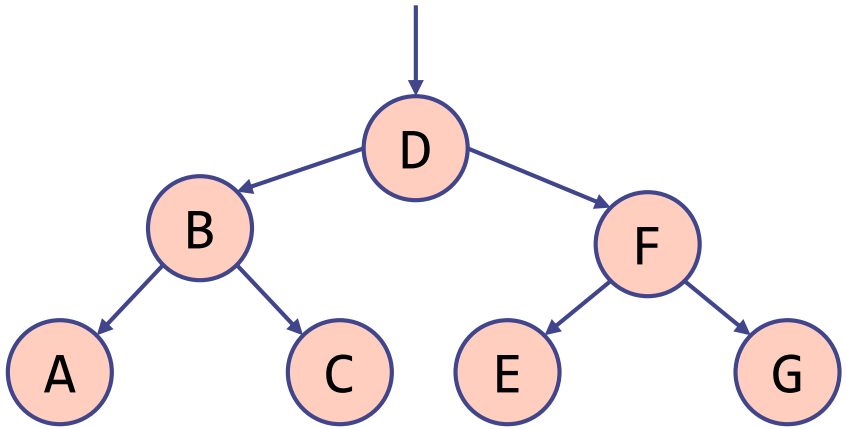
Binary Search Tree



Binary Search Tree

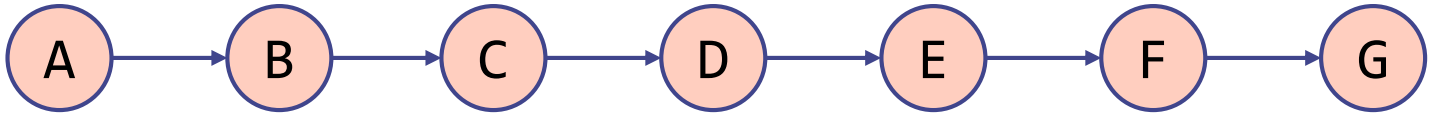
- دسترسی در مقایسه با لیست و آرایه؟

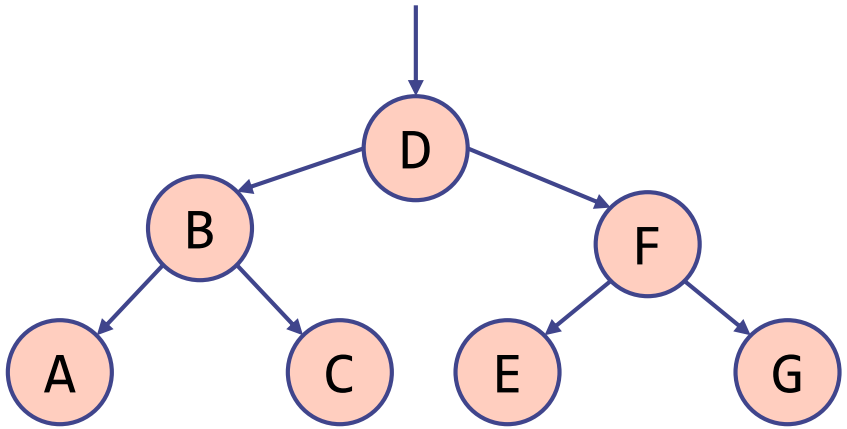




Binary Search Tree

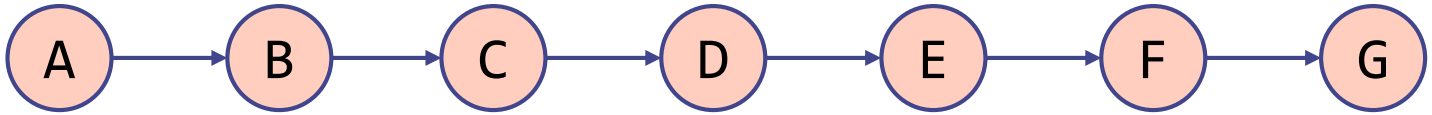
- جستجو در مقایسه با لیست و آرایه؟

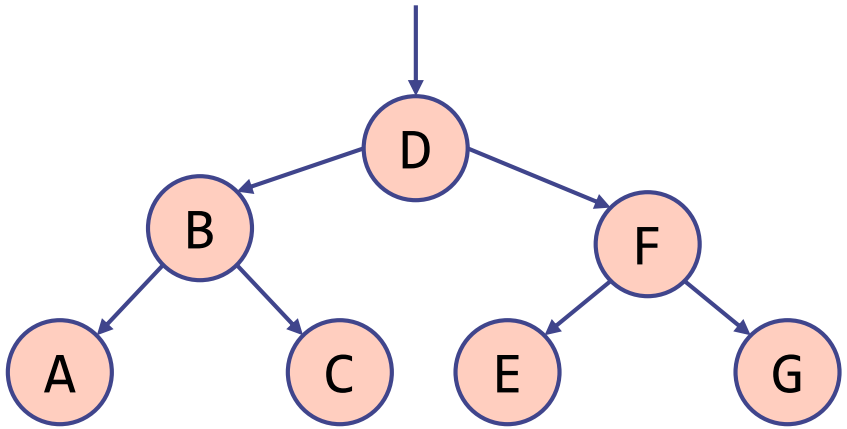




Binary Search Tree

- حذف در مقایسه با لیست و آرایه؟

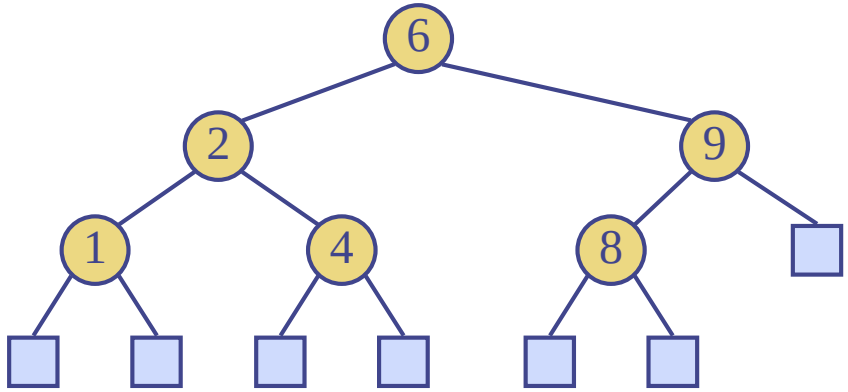




Binary Search Trees

- A binary search tree is a binary tree storing keys (or key-value entries) at its internal nodes and satisfying the following property:
  - Let u, v, and w be three nodes such that u is in the left subtree of v and w is in the right subtree of v. We have $key(u) \leq key(v) \leq key(w)$

- External nodes do not store items (GTM version)

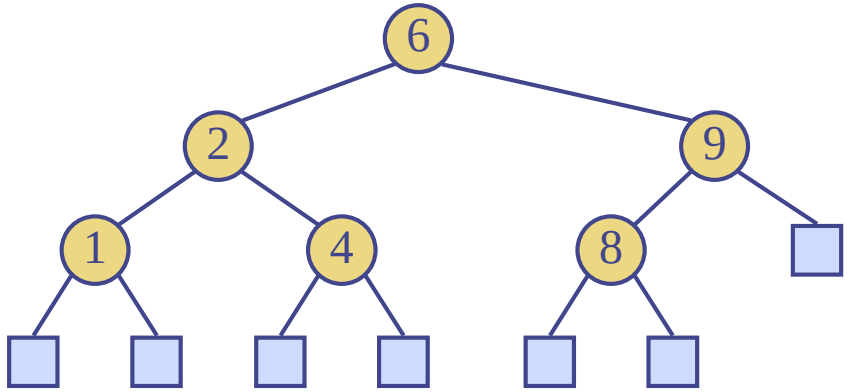


Binary Search Trees

- چگونه از BST یک لیست sort شده بسازیم؟

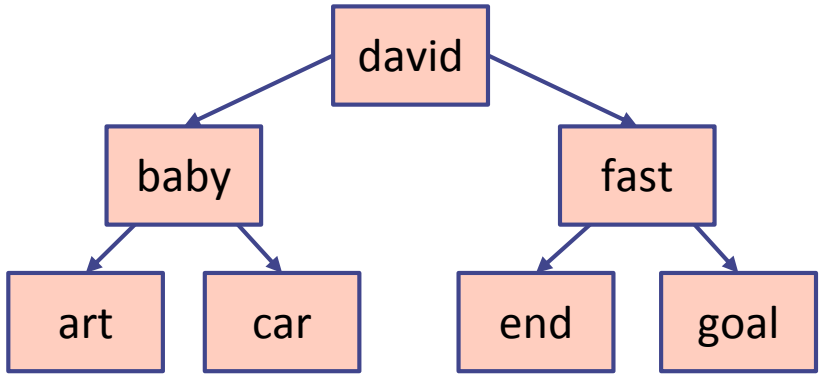
Binary Search Trees

- A binary search tree is a binary tree storing keys (or key-value entries) at its internal nodes and satisfying the following property:
  - Let u, v, and w be three nodes such that u is in the left subtree of v and w is in the right subtree of v. We have $key(u) \leq key(v) \leq key(w)$
- External nodes do not store items (GTM version)
- An inorder traversal of a binary search trees visits the keys in increasing order



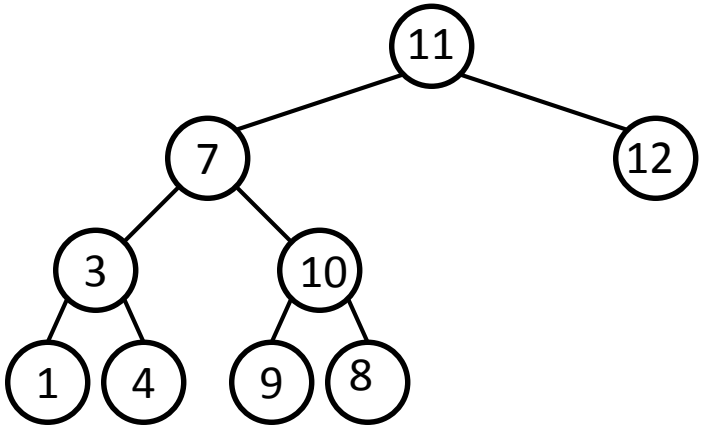
Binary Search Trees

- کلید تکراری نداریم (ترتیب به طور شفاف و گذرا):



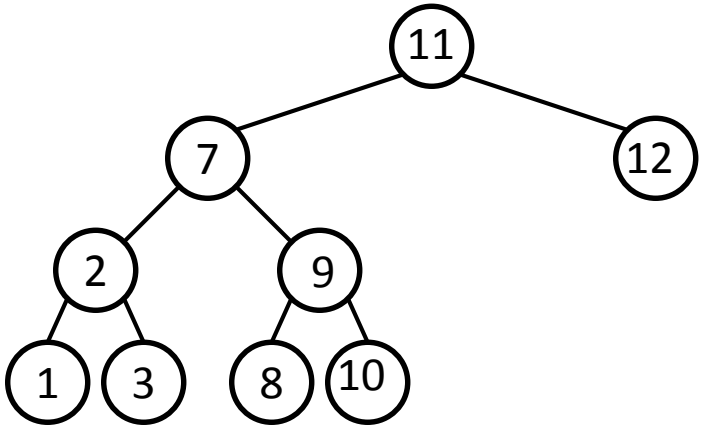
مثال

- BST؟



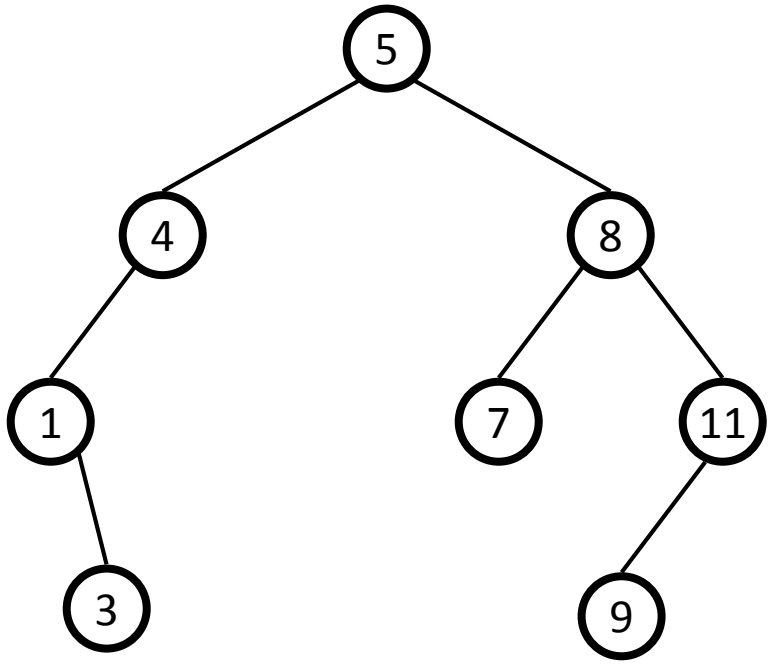
مثال

- BST؟



مثال

- BST؟



مثال

- چه مقداری؟