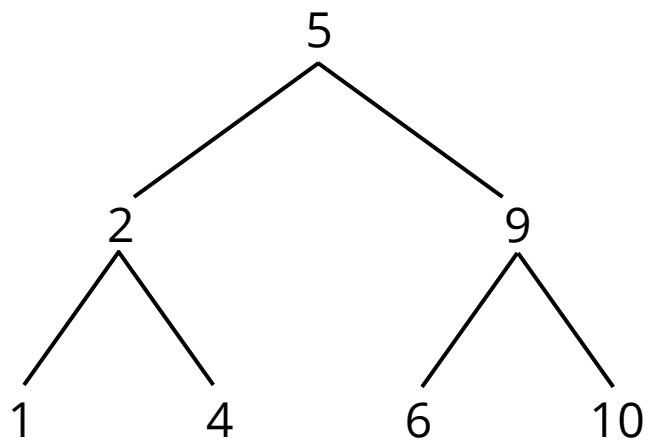
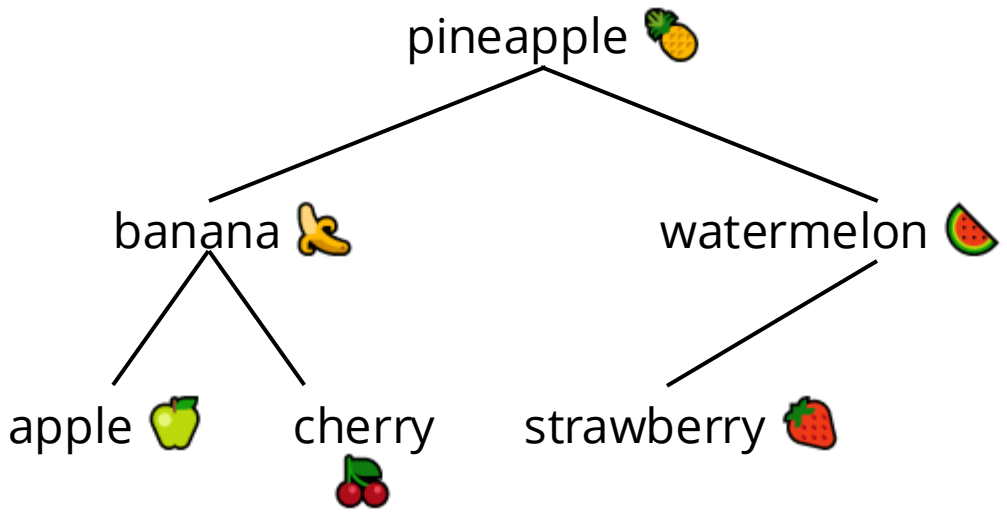
Binary Search Tree

left value < root < right value





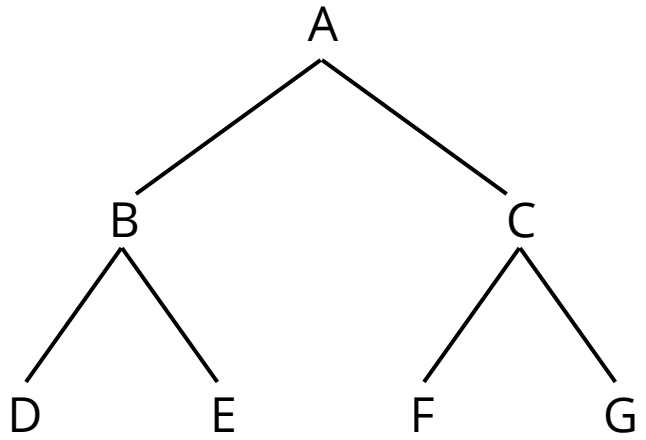
Binary Search Tree

- **Pre-order:**
  - Visit current node
  - Recurse left
  - Recurse right

  A ➔ B ➔ D ➔ E ➔ C ➔ F ➔ G
- **Post-order:**
  - Recurse left
  - Recurse right
  - Visit current node

  D ➔ E ➔ B ➔ F ➔ G ➔ C ➔ A
- **In-order:**
  - Recurse left
  - Visit current node
  - Recurse right

  D ➔ B ➔ E ➔ A ➔ F ➔ C ➔ G



Ex10.08: Draw a BST

Ex10.08: Draw a BST

Construct (draw) a binary search tree on strings by inserting the signs of the Zodiac in their sequence order into a tree that is initially empty: Aquarius, Pisces, Aries, Taurus, Gemini, Cancer, Leo, Virgo, Libra, Scorpio, Sagittarius, Capricorn.

Then add in the twelve animals of the Chinese zodiac: Rat, Ox, Tiger, Rabbit, Dragon, Snake, Horse, Goat, Monkey, Rooster, Dog, Boar.

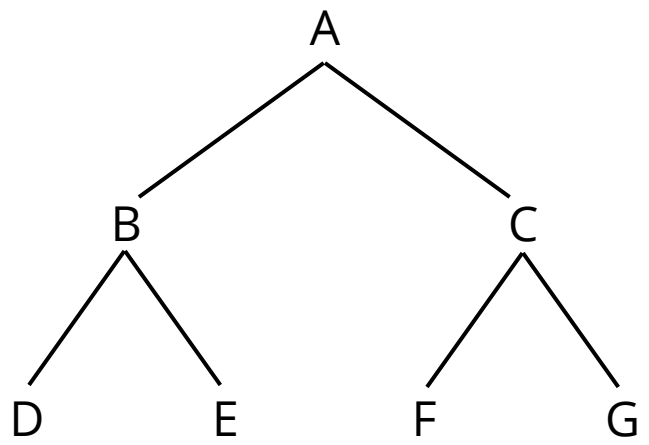
Then add in your name and the name of some friends.

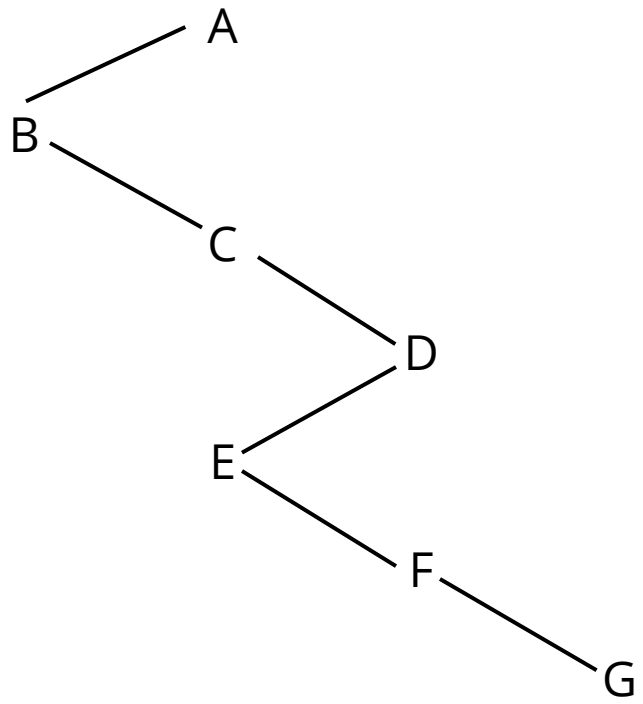
Finally, calculate the average depth of the items in the tree, with the root node of the tree at depth 1, its children at depth 2, and so on.

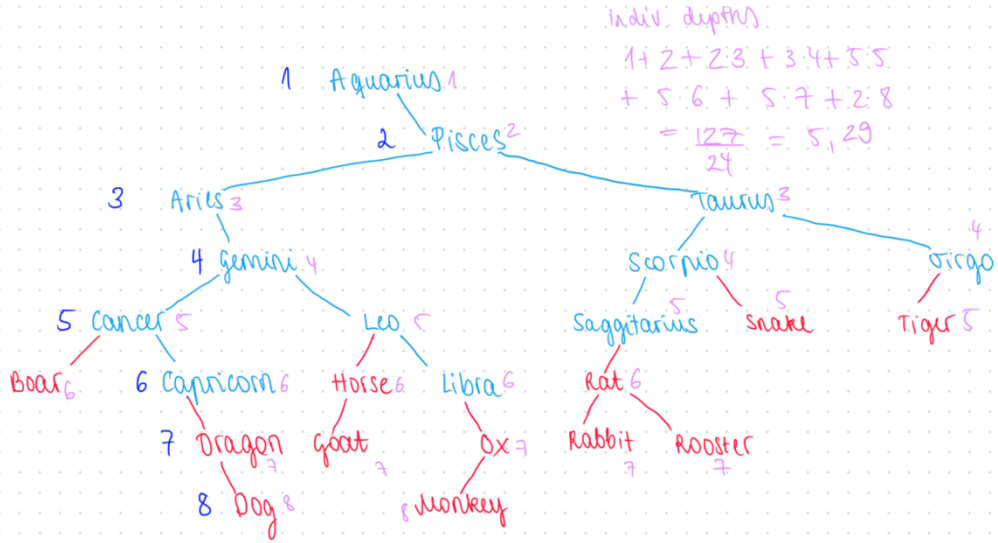
Verify that for "typical" input sequences like this one, the average depth of the items in the tree is much smaller than the number of objects in the tree.

Binary Search Tree

- Time complexities of ordered **insert, delete** and **search** for a binary search tree:
  - Best case/Average case (balanced tree): $O(\log n)$
  - Worst case (stick): $O(n)$







indiv. depths
1+2+2·3+3·4+5·5
+5·6+5·7+2·8
= 127/24 = 5,29

24 entries
but only depth of 8

Time complexities of insert, delete, search for BST?

all: O(log(n)) average/best case (balanced tree )
O(n) worst case (stick )

56%

merge sort

Merge Sort

[20, 1, 3, 8, 4, 10, 2, 1, 21]

we split the larger part on the right

split

merge

[20, 1, 3, 8] [4, 10, 2, 1, 21]

[20, 1] [3, 8]

[20] [1]

[1, 20]

[3] [8]

[3, 8]

[1, 3, 8, 20]

[4, 10] [2, 1, 21]

[4] [10]

[4, 10]

[2] [1, 21]

[1] [21]

[1, 21]

[1, 2, 21]

[1, 2, 4, 10, 21]

[1, 1, 2, 3, 4, 8, 10, 20, 21]