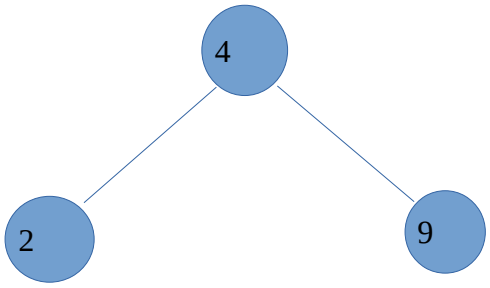
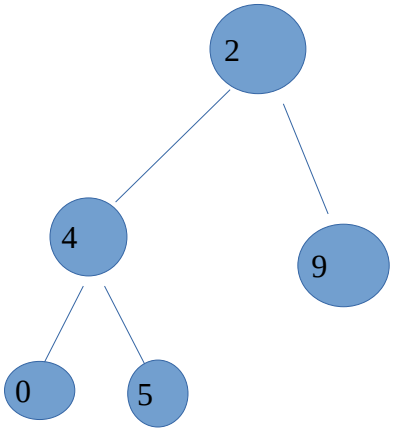
2, 4, **9**, 0 , 5: Binary Search Tree [**Search**]

4, 2, 9: New order of data. AVL, it changed the order of input.
So that we could get a balanced binary tree. (complete binary tree)



The tree is not balanced.



1, places: 2^1 => 2.

Almost Complete Binary Tree. => converted to complete.
(There is always place to insert in the existing inner node.)

[2][4][9][]

Try to insert, from left to right child,
Increase height if and only if, there is
complete binary tree.

Memoization

>> Dynamic Programming

>> You want to remember partial computation.

>> path finding

Fibonacci ??

compute n^{th} term.

40^{th} fib => 39^{th}, 38^{th}, ??

39^{th} => 38^{th}, 37^{th}

38^{th} => 37^{th}, 36^{th} (I can use this value)

37^{th} => 36^{th}, 35^{th}

=> declare memoization unit.
=> fib (n+2) => fib(n) + fib(n+1)

```
fib(n) => getOrElse(memoization(n), computeFib(n))
computeFib(n) => computeIt(n) & put(memoize, n)
```

int a[100]; ?? execute ??

int a[32767] ?? does this execute ??

Dynamic ?? Array

100 elements a[1000]
1001 elements
How do I incorporate the 1001th elements in **a**.

f(x) => x^2 + 10x % period (so that we can get the hash value in our predefined range 0, 10)

Good: minimize collision

    Distribution ?? (Uniform)
    two primes.

[1 … 10000]: randomization, crypto math.

67??