Buyer Beware: Hollywood Special Effects Now Permeate Property Listings

https://www.wsj.com/articles/home-sellers-doctored-photos-challenge-buyers-bots-11551708001

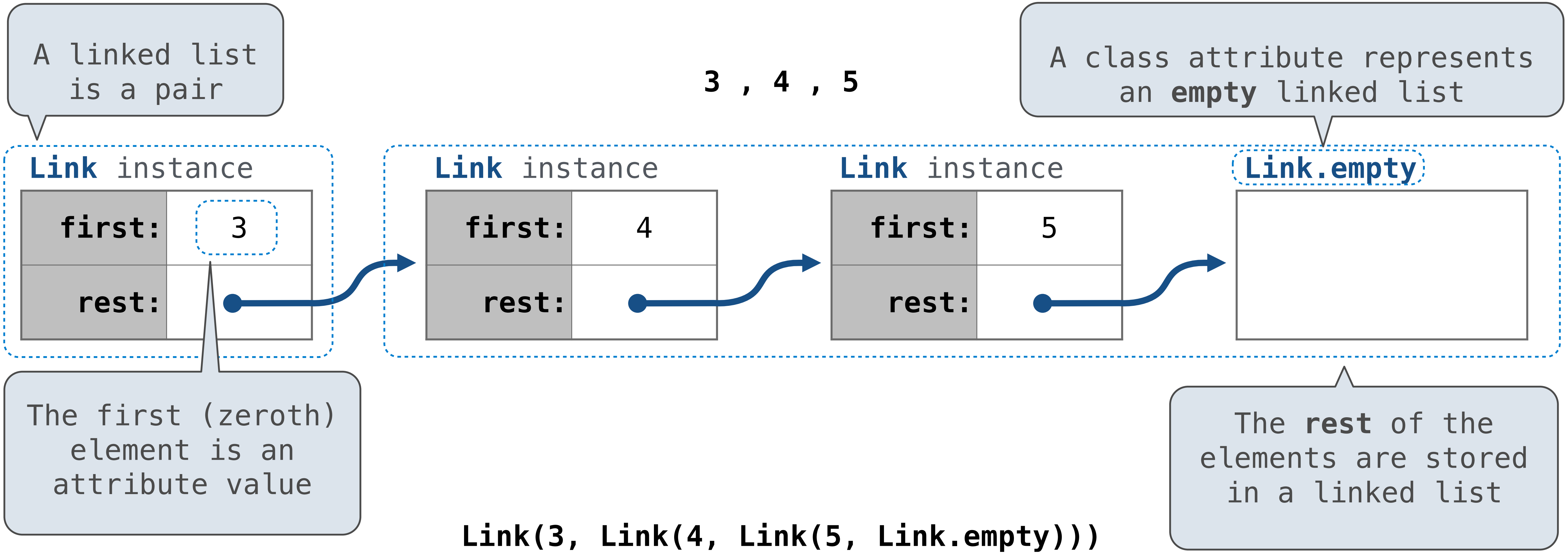
"**Computer-generated imagery** traditionally reserved for Hollywood **has now become so cheap and easy to use that home sellers are digitally removing walls, hiding ugly paneling, and even adding swimming pools to online listings**. In addition, most home searches begin online, and deals are often reached without in-person showings, especially among investors who are putting photos through their own algorithms to price properties. The technology allows sellers to make brown lawns look green, and stage rooms with virtual furniture. **However, these techniques could result in buyer disappointment when they arrive for in-person showings, or lead to miscalculations in renovation budgets**. The ease and extent to which images can be altered has agents and the organizations that monitor real estate listings wondering how to create guidelines for the technology."

Announcements

Linked Lists

Linked List Structure

A linked list is either empty **or** a first value and the rest of the linked list



A linked list is a pair

A class attribute represents an **empty** linked list

The first (zeroth) element is an attribute value

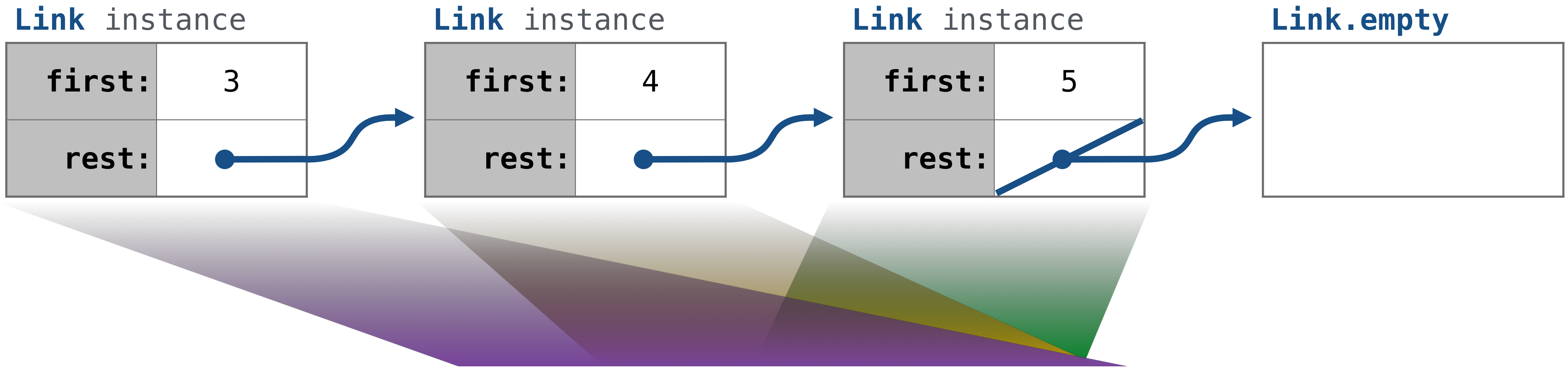
The **rest** of the elements are stored in a linked list

```
Link(3, Link(4, Link(5, Link.empty)))
```

Linked List Structure

A linked list is either empty **or** a first value and the rest of the linked list

3 , 4 , 5



`Link(3, Link(4, Link(5, Link.empty)))`

Linked List Class

Linked list class: attributes are passed to `__init__`

```
class Link:

    empty = ()

    def __init__(self, first, rest=empty):
        assert rest is Link.empty or isinstance(rest, Link)
        self.first = first
        self.rest = rest
```

Some zero-length sequence

Returns whether rest is a Link

help(isinstance): Return whether an object is an instance of a class or of a subclass thereof.

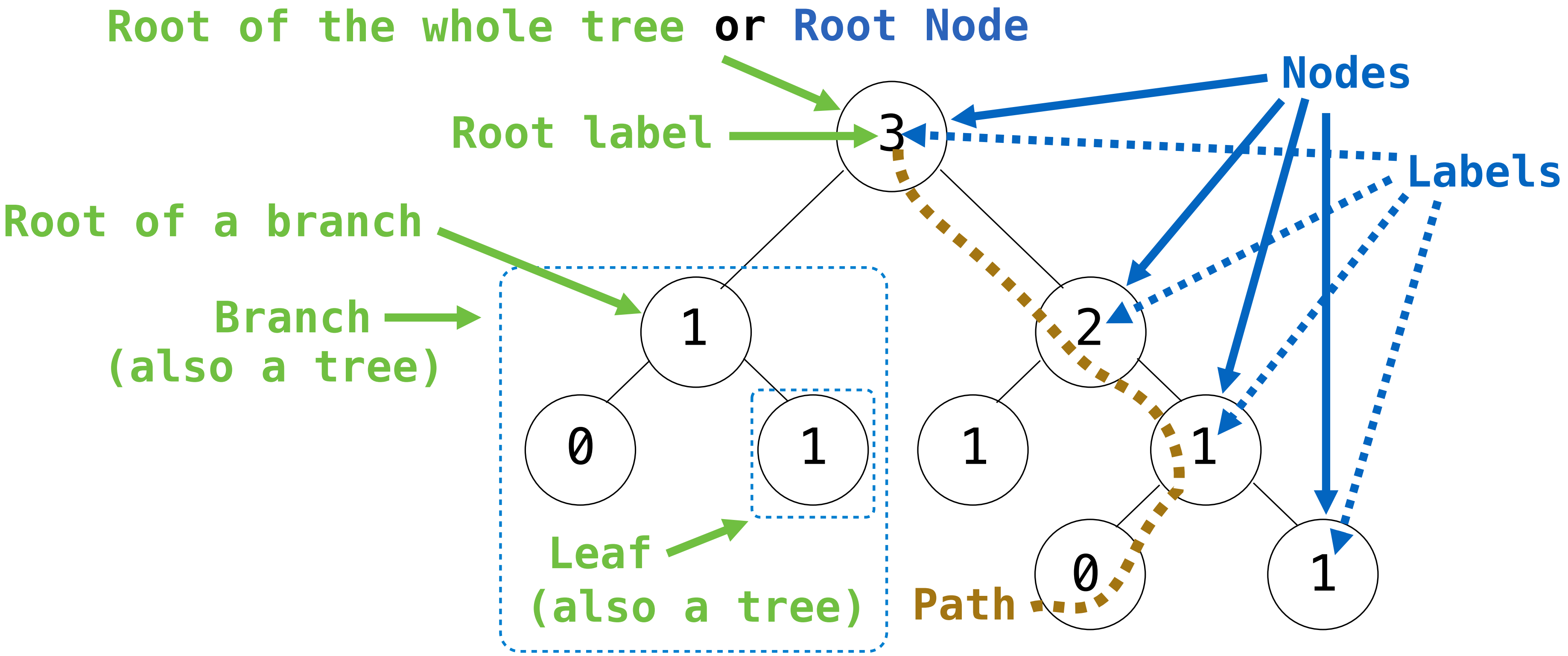
```
Link(3, Link(4, Link(5            )))
```

(Demo)

Property Methods

Tree Class

Tree Abstraction (Review)



Recursive description (wooden trees):

A **tree** has a **root label** and a list of **branches**

Each **branch** is a **tree**

A **tree** with zero **branches** is called a **leaf**

A **tree** starts at the **root**

Relative description (family trees):

Each location in a tree is called a **node**

Each **node** has a **label** that can be any value

One node can be the **parent/child** of another

The top node is the **root node**

People often refer to labels by their locations: "each parent is the sum of its children"

Tree Class

A Tree has a label and a list of branches; each branch is a Tree

```
class Tree:
    def __init__(self, label, branches=[]):
        self.label = label
        for branch in branches:
            assert isinstance(branch, Tree)
        self.branches = list(branches)
```

```
def fib_tree(n):
    if n == 0 or n == 1:
        return Tree(n)
    else:
        left = fib_tree(n-2)
        right = fib_tree(n-1)
        fib_n = left.label + right.label
        return Tree(fib_n, [left, right])
```

```
def tree(label, branches=[]):
    for branch in branches:
        assert is_tree(branch)
    return [label] + list(branches)
def label(tree):
    return tree[0]
def branches(tree):
    return tree[1:]
```

```
def fib_tree(n):
    if n == 0 or n == 1:
        return tree(n)
    else:
        left = fib_tree(n-2)
        right = fib_tree(n-1)
        fib_n = label(left) + label(right)
        return tree(fib_n, [left, right])
```

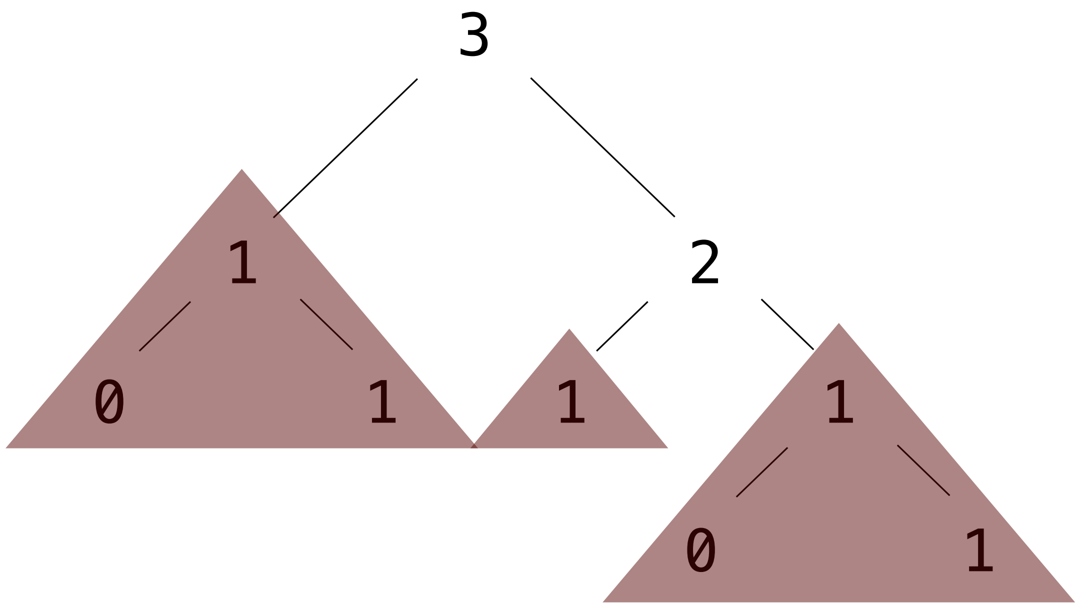
(Demo)

Tree Mutation

Example: Pruning Trees

Removing subtrees from a tree is called *pruning*

Prune branches before recursive processing



```
def prune(t, n):

    """Prune sub-trees whose label value is n."""
                        b                                      b.label != n
    t.branches = [______________ for b in t.branches if _____________________]

    for b in t.branches:
                             b                                  n
        prune(_______________________________, _______________________________)
```

(Demo)

Example: Pruning Trees

Removing subtrees from a tree is called *pruning*

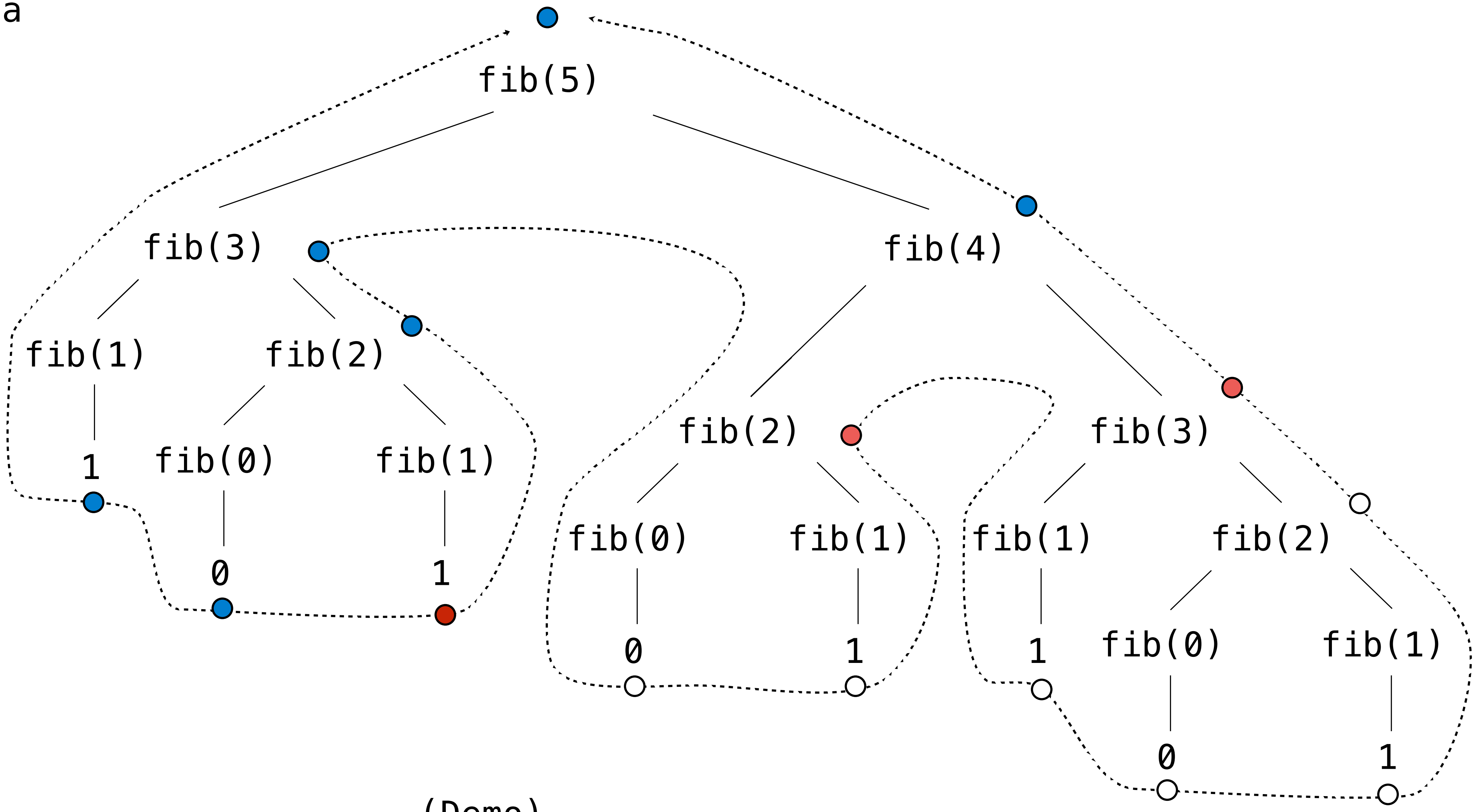
Prune branches before recursive processing

Memoization:

🔵 Returned by fib

🔴 Found in cache

○ Skipped



(Demo)