Lecture 12, 17 September 2026

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming and Data Structures with Python

Inductive defns & recursive computation

Define functions inductively

$fact(0) = 1$ — Base

$fact(n) = n * fact(n-1)$ — Define $f(k)$ in terms of $f(j), j<k$

$fib(0) = 0$

$fib(1) = 1$

$fib(n) = fib(n-1) + fib(n-2)$

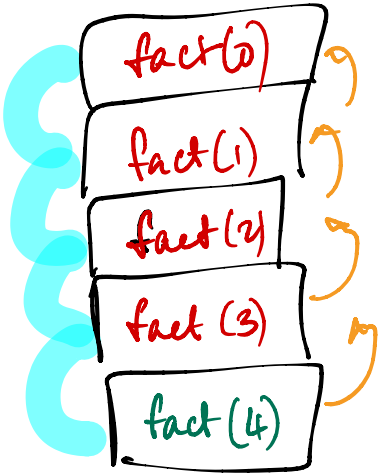
0	1	2	3	4	5	6	...
0	1	1	2	3	5	8	

Exercise

Define $\binom{n}{k}$ inductively

Computation **Recursion**

fact(n) will evaluate fact(n-1)


fact(0)
fact(1)
fact(2)
fact(3)
fact(4)


w()
h()
g()
f()

Typical problem

Gap in definition

Infinite sequence of recursive calls

Like messing up a while loop

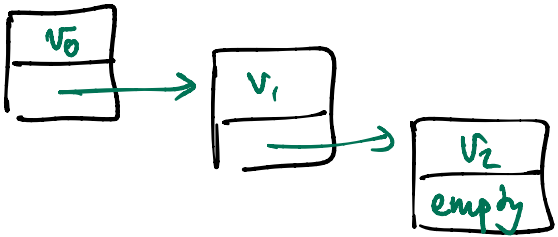
Python — recursion limit

Inductive data structures

List is

either empty

or value attached to a list $v : l$



Length of list

```
length emptylist = 0
length (v : l) = 1 + length l
```

Dictionary implementation

`d = {}` – Empty list

```
def emptylist(l):
    return l == {}
```

`d["val"]`

`d["nxt"]`

```
def length(l):
    if emptylist(l):
        return 0
    else:
        return 1 + length(l["nxt"])
```

Python list

`[]` empty list

`l[0]` = "val"

`l[1:]` = "rest"

```
def mylen(l):
    if l==[]:
        return 0
    else:
        return 1 +
          mylen(l[1:])
```

appendlist

if l is empty

    create a value here

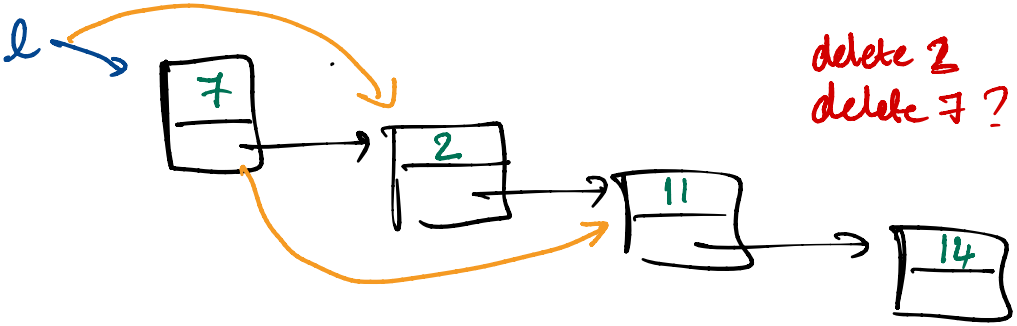
else

    appendlist(l["next"], v)

delete v from l

What if no v in l ? — Do nothing

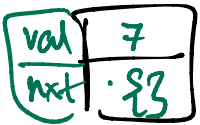
What if multiple copies of v in l? — First v is deleted



delete 2

delete 7 ?

del(l[k]) – deletes key k



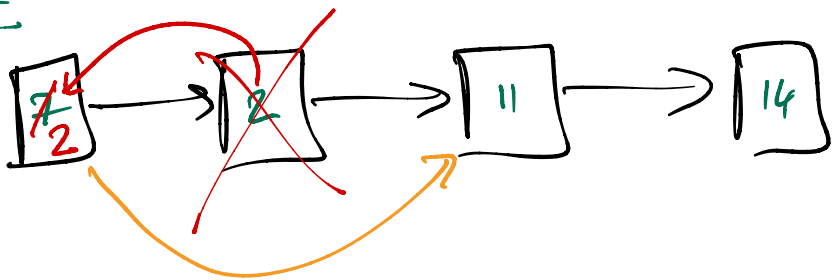
delete 7

{}

del(l['nxt'])

del(l['val'])

Delete 7



In general - make previous node point to next node

Inductive definition?

delete ({}, v) ⟶ {}

delete (l, v) where l["next"] = {}

if l["value"] ≠ v — return l

else "nullify" l del(l["value"])
del(l["next"])

del(l, v) — not empty, has a next node

① l["value"] == v

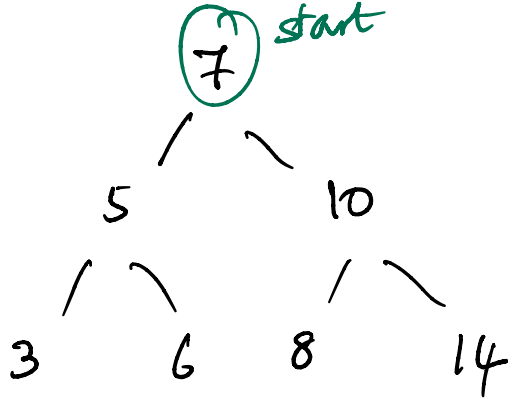
Copy l["next"]["value"] here

Set l["next"] to l["next"]["next"]

② l["value"] ≠ v

del(l["next"], v)

Two dimensional struct



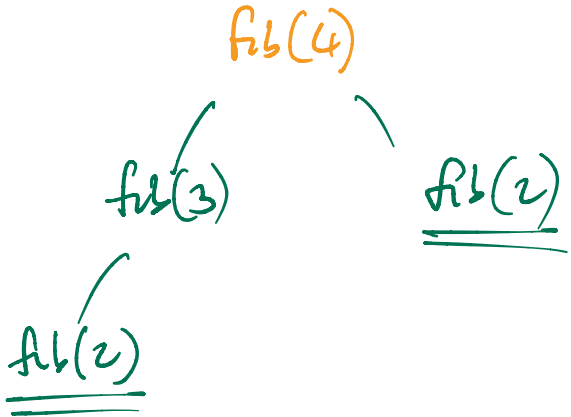
"Tree"

"Search tree"

return ~~fact(n)~~ = n * fact(n-1)

"Tail recursion"

return ~~fact(n)~~ = fact(n-1) * n


fib(4)
fib(3)
fib(2)
fib(2)