61A Lecture 20

Announcements

Sets

Sets

One more built-in Python container type

- Set literals are enclosed in braces
- Duplicate elements are removed on construction
- Sets have arbitrary order, just like dictionary entries

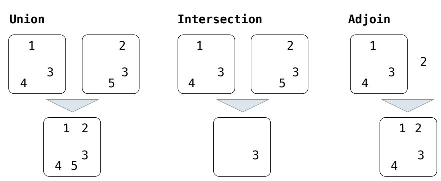
```
>>> s = {3, 2, 1, 4, 4}
>>> s
{1, 2, 3, 4}
>>> 3 in s
True
>>> len(s)
4
>>> s.union({1, 5})
{1, 2, 3, 4, 5}
>>> s.intersection({6, 5, 4, 3})
{3, 4}
>>> s
{1, 2, 3, 4}
```

(Demo)

Implementing Sets

What we should be able to do with a set:

- **Membership testing:** Is a value an element of a set?
- **Union:** Return a set with all elements in set1 or set2
- **Intersection:** Return a set with any elements in set1 and set2
- **Adjoin:** Return a set with all elements in s and a value v



Sets as Linked Lists

Sets as Unordered Sequences

Proposal 1: A set is represented by a linked list that contains no duplicate items.

Time order of growth

```
def empty(s):
    return s is Link.empty
```

$\Theta(1)$

```
def contains(s, v):
    """Return whether set s contains value v.

    >>> s = Link(1, Link(3, Link(2)))
    >>> contains(s, 2)
    True
    """
```

Time depends on whether & where v appears in s

$\Theta(n)$

Assuming v either does not appear in s ***or*** *appears in a uniformly distributed random location*

(Demo)

Sets as Unordered Sequences

Time order of growth

```
def adjoin(s, v):
    if contains(s, v):
        return s
    else:
        return Link(v, s)
```

$\Theta(n)$ — The size of the set

```
def intersect(set1, set2):
    in_set2 = lambda v: contains(set2, v)
    return filter_link(in_set2, set1)
```

Return elements x for which in_set2(x) returns a true value

$\Theta(n^2)$ — If sets are the same size

```
def union(set1, set2):
    not_in_set2 = lambda v: not contains(set2, v)
    set1_not_set2 = filter_link(not_in_set2, set1)
    return extend_link(set1_not_set2, set2)
```

Return a linked list containing all elements in set1_not_set2 followed by all elements in set2

$\Theta(n^2)$

Sets as Ordered Linked Lists

Sets as Ordered Sequences

Proposal 2: A set is represented by a linked list with unique elements that is ***ordered from least to greatest***

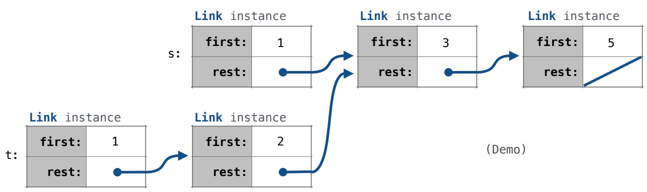
Parts of the program that...	Assume that sets are...	Using...
Use sets to contain values	Unordered collections	`empty`, `contains`, `adjoin`, `intersect`, `union`
Implement set operations	Ordered linked lists	`first`, `rest`, `<`, `>`, `==`

Different parts of a program may make different assumptions about data

Searching an Ordered List

```
>>> s = Link(1, Link(3, Link(5)))
>>> contains(s, 1)
True
>>> contains(s, 2)
False
>>> t = adjoin(s, 2)
```

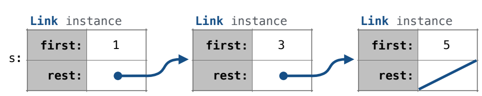
Operation	Time order of growth
contains	$\Theta(n)$
adjoin	$\Theta(n)$



(Demo)

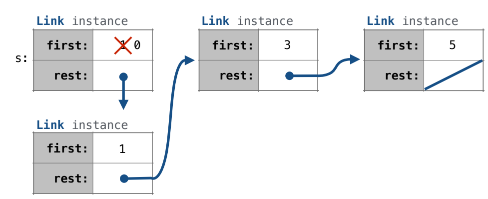
Set Mutation

Adding to an Ordered List



add(s, 0)

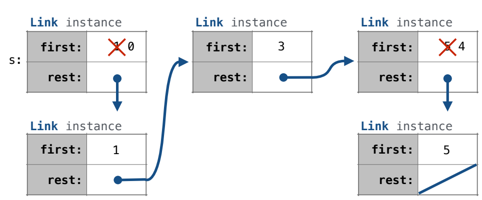
Adding to an Ordered List



add(s, 3)

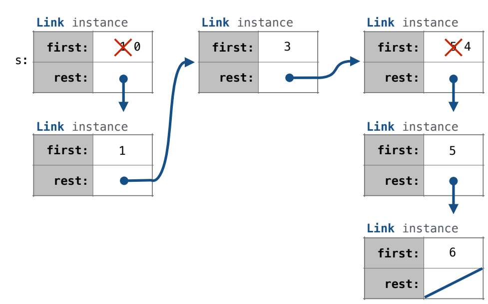
add(s, 4)

Adding to an Ordered List



add(s, 6)

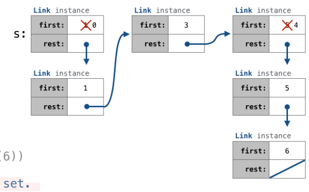
Adding to an Ordered List



Adding to an Ordered List

```
def add(s, v):
    """Add v to a set s and return s.

    >>> s = Link(1, Link(3, Link(5)))
    >>> add(s, 0)
    Link(0, Link(1, Link(3, Link(5))))
    >>> add(s, 3)
    Link(0, Link(1, Link(3, Link(5))))
    >>> add(s, 4)
    Link(0, Link(1, Link(3, Link(4, Link(5)))))
    >>> add(s, 6)
    Link(0, Link(1, Link(3, Link(4, Link(5, Link(6))))))
    """
    assert not empty(s), "Cannot add to an empty set."
    if s.first > v:
        s.first, s.rest = ____v____ , ____Link(s.first, s.rest)____
    elif s.first < v and empty(s.rest):
        s.rest = ____Link(v, s.rest)____
    elif s.first < v:
        ____add(s.rest, v)____
    return s
```



Set Operations

Intersecting Ordered Linked Lists

Proposal 2: A set is represented by a linked list with unique elements that is ***ordered from least to greatest***

```
def intersect(set1, set2):
    if empty(set1) or empty(set2):
        return Link.empty
    else:
        e1, e2 = set1.first, set2.first
        if e1 == e2:
            return Link(e1, intersect(set1.rest, set2.rest))
        elif e1 < e2:
            return intersect(set1.rest, set2)
        elif e2 < e1:
            return intersect(set1, set2.rest)
```

Order of growth? $\Theta(n)$ (Demo)