CS 150: Dictionaries and Sets

Cynthia Taylor
Oberlin College
April 21st 2014

Test 2

- Returned in Lab this week

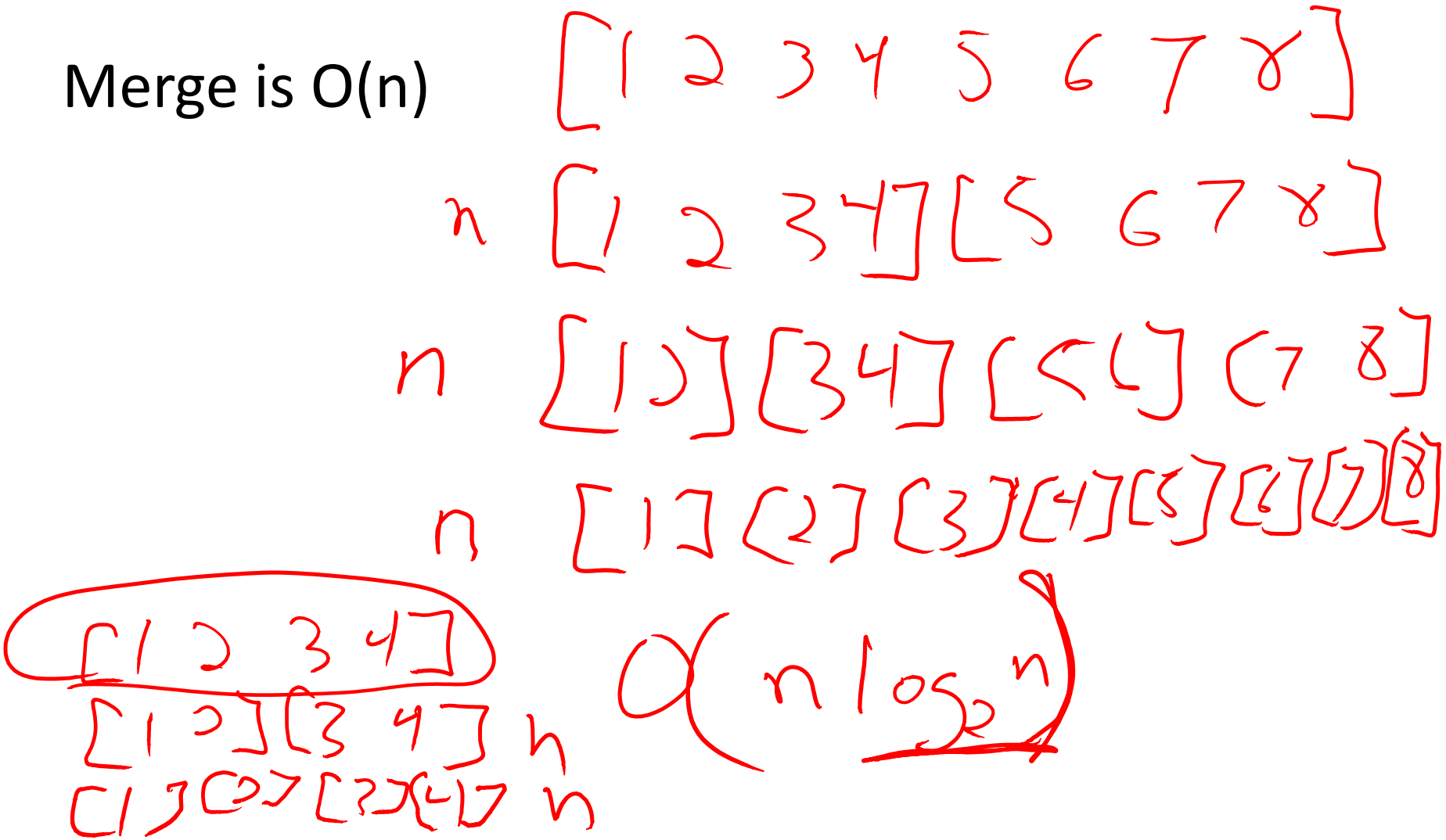
Recall from Last Class

```
def mergeSort(A):
  if len(A) > 1:
     m = len(A)//2
     B = mergeSort(A[:m])
     C = mergeSort(A[m:])

     A = merge(B,C)
   return A
```

Complexity of Mergesort

Merge is O(n)



Bird Watching

- We want to keep track of how many of each bird we have seen
  - Robin: 3, Pigeon: 45, etc

- Could use parallel lists

```
birds = [‘robin’, ‘pigeon’, ‘falcon’]
counts = [3,45,2]
```

Adding a new bird sighting

```
def new_sighting(birds, counts, new_bird):
    if new_bird not in birds:
        birds.append(new_bird)
        missing code
    ind = birds.index(new_bird)
    counts[ind] = counts[ind] + 1
```

index of new_bird

A. `counts.append(0)`
B. `counts.append(1)`
C. `counts.append(new_bird)`
D. No code necessary
E. I don't know

Using Dictionaries

```
bird_dict = {"robin":3,"pigeon:45","falcon":3}

def new_sighting(bird_dict, new_bird):
    if new_bird not in bird_dict:
        bird_dict[new_bird] = 0
    bird_dict[new_bird] = bird_dict[new_bird] + 1
```

- Only one dictionary
- Instead of looking for index, look up by *key*

Keys and Values

- Keys are immutable
- Values are mutable

- Use `d[k] = v` to add key k with value v to dictionary d
- If k is already present, its value is overwritten

After this code, d will be

```
d = {"a":1,"b":2}
d["c"] = 3
d["b"] = 4
```

A. {“a”:1, “b”:2, “c”:3}

B. {“a”:1, “b”:4, “c”:3}

C. {“a”:1, “b”:2, “b”:4, “c”:3}

D. This will cause an error

E. I don’t know

Getting Values from Dictionaries

- `d[k]` will return the value associated with key k in dictionary d
  - If k does not exist, this causes an error
- `d.get(k)` will also return the value associated with key k in dictionary d
  - Returns None if k does not exist
  - If a second parameter is included `d.get(k,v)`, then v is returned instead of None if k is not found

What is d at the end of this code?

```
d = {3:4}
d[5] = d.get(4, 8)
d[4] = d.get(3, 9)
```

A. {3:4, 5:8, 4:9}

B. {3:4, 5:8, 4:4}

C. {3:4, 5:4, 4:3}

D. Error caused by get

E. I don't know

Keys, Values and Items

- `d.keys()` returns a dictionary's keys
- `d.values()` returns a dictionary's values
- `d.items()` returns a dictionary's key-value pairs
- These are similar to lists, but NOT lists. To turn into a list, `list(d.keys())`

Sets

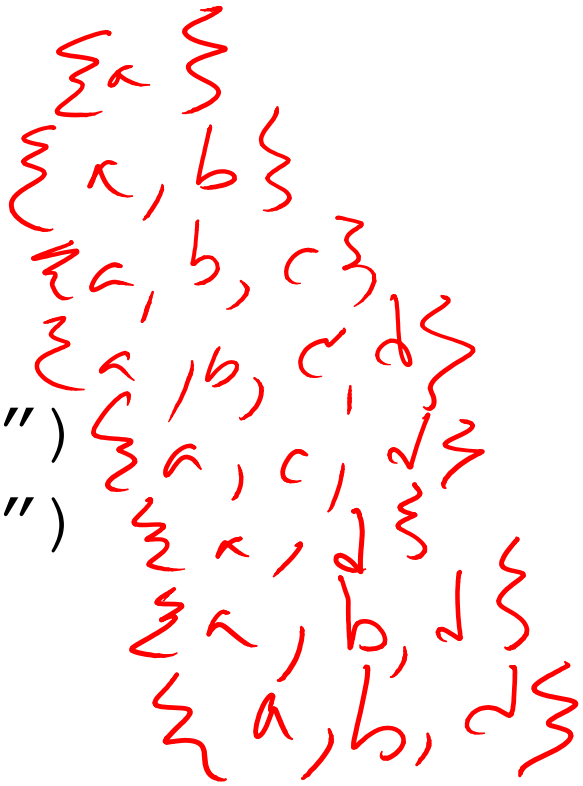
- Set of values
- Unlike lists, not ordered
- Unlike dictionaries, no keys
- No duplicate items – adding an element that already exists to a set has no effect

Set Functions

- `s = {a,b}` creates a set with elements a and b
- `s.add(v)` adds a value to a set
- `s.remove(v)` removes a value from a set
- `For v in s` iterates over the set

After this code, alpha will be

```
alpha = {"a"}
alpha.add("b")
alpha.add("c")
alpha.add("d")
alpha.remove("b")
alpha.remove("c")
alpha.add("b")
alpha.add("d")
```



A. {"b","d"}

B. {"a","b","d"}

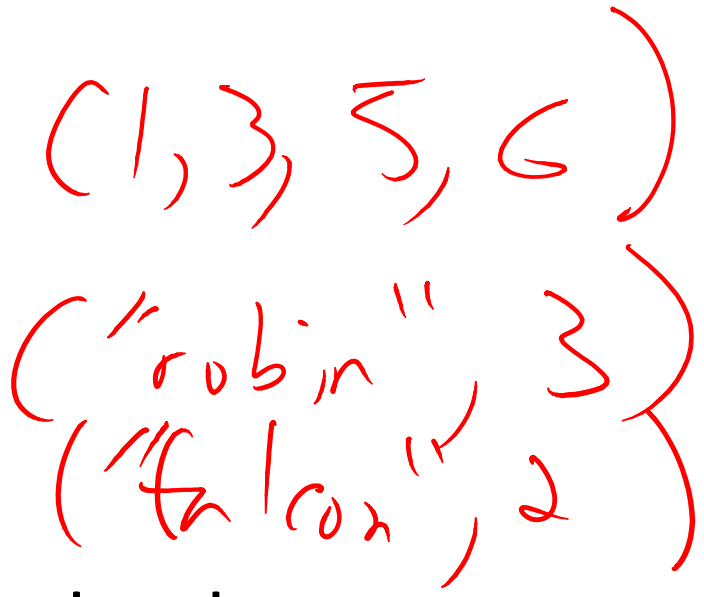
C. {"a","b","c","d"}

D. {"a","b","b","d"}

E. I don't know

Tuples

- (“a”,10)
- Immutable
- Usually have some structure



Next Time

- Multiple Processes

- Lab 10 – Tuesday at 10 pm