Algorithmic Approaches for Biological Data, Lecture #15

Katherine St. John

City University of New York
American Museum of Natural History

23 March 2016

Outline

- Sorting by Keys

Outline

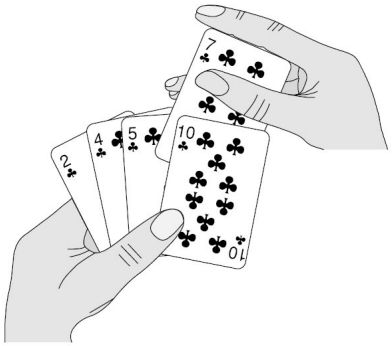
- Sorting by Keys
- Lambda Functions

Outline

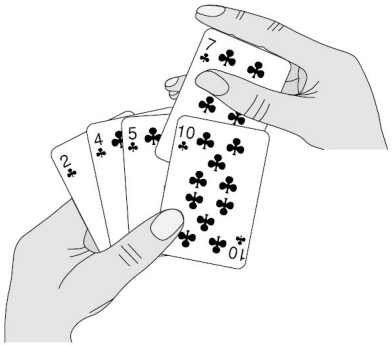
- Sorting by Keys
- Lambda Functions
- Recursion

Sorting

- Input: List of n items

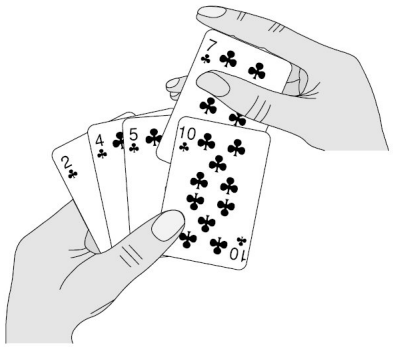


Sorting



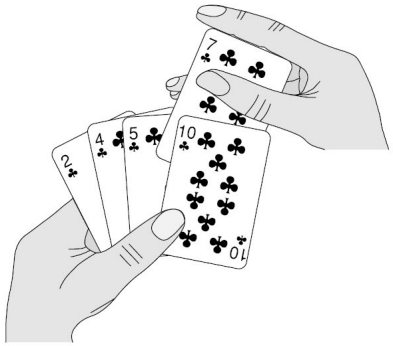
- Input: List of n items
- Output: The n items, in sorted order

Sorting



- Input: List of n items
- Output: The n items, in sorted order
- Recap: Some common approaches:
  - Take the largest card and move to the end. Repeat with next largest... BubbleSort
  - Start a new list and insert each card into it to keep in order...InsertionSort
  - Divide the cards in half. Sort each half and merge sorted results together....MergeSort
- All need to compare values and re-order in some way.

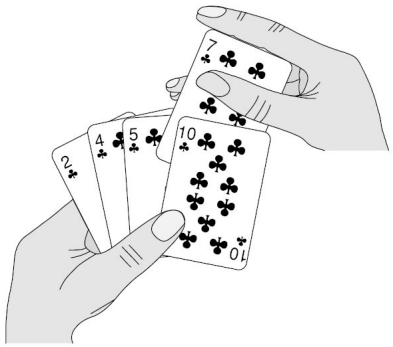
Built-in Sorting Functions



Two useful functions that are built-in to Python:

- `sorted()`

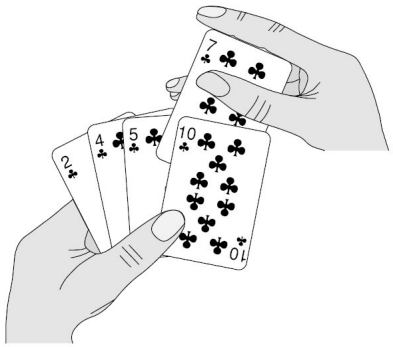
Built-in Sorting Functions



Two useful functions that are built-in to Python:

- `sorted()`
  - Example: `print sorted(myList)`

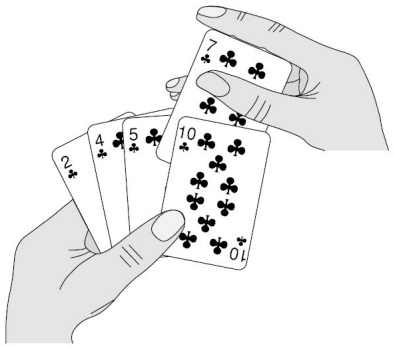
Built-in Sorting Functions



Two useful functions that are built-in to Python:

- `sorted()`
  - Example: `print sorted(myList)`
  - Does not change `myList`.
    Instead, returns a sorted list.

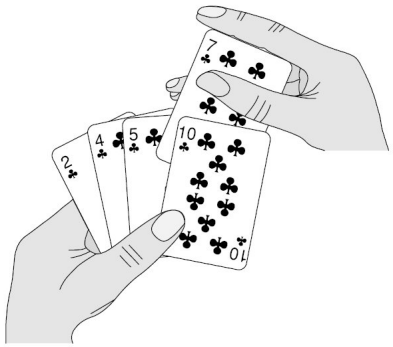
Built-in Sorting Functions



Two useful functions that are built-in to Python:

- `sorted()`
  - Example: `print sorted(myList)`
  - Does not change `myList`.
    Instead, returns a sorted list.
- `sort()`

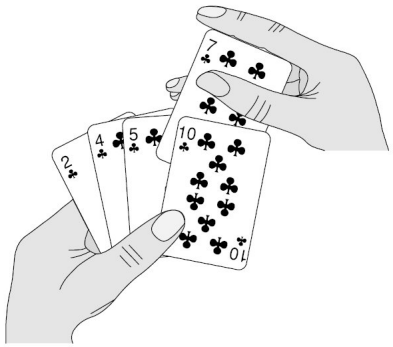
Built-in Sorting Functions



Two useful functions that are built-in to Python:

- `sorted()`
  - Example: `print sorted(myList)`
  - Does not change `myList`.
    Instead, returns a sorted list.
- `sort()`
  - Example: `myList.sort()`

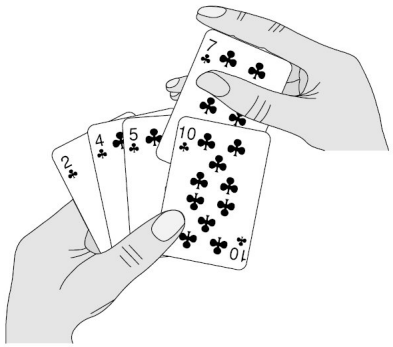
Built-in Sorting Functions



Two useful functions that are built-in to Python:

- `sorted()`
  - Example: `print sorted(myList)`
  - Does not change `myList`.
    Instead, returns a sorted list.
- `sort()`
  - Example: `myList.sort()`)
  - Does change `myList`.

Built-in Sorting Functions



Two useful functions that are built-in to Python:

- `sorted()`
  - Example: `print sorted(myList)`
  - Does not change `myList`.
    Instead, returns a sorted list.
- `sort()`
  - Example: `myList.sort()`)
  - Does change `myList`.
- Can use `sorted()` on tuples
  (Cannot use `.sort()` since tuples are not mutable.)

Lambda Functions

- Python supports ‘anonymous functions.’

Lambda Functions

- Python supports ‘anonymous functions.’
- Called lambda functions

Lambda Functions

- Python supports 'anonymous functions.'
- Called lambda functions
- Example:

```
f = lambda x:  x**2
print f(3) #Prints 9
```

- They are very useful for embedding into other functions.

Lambda Functions

- Python supports 'anonymous functions.'
- Called lambda functions
- Example:

```
f = lambda x:  x**2
print f(3) #Prints 9
```

- They are very useful for embedding into other functions.
- Example:

```
rooTuples = [('eleanor', 'A', 1884),
             ('teddy', 'A', 1858),
             ('alice', 'B', 1884)]
```

Lambda Functions

- Python supports 'anonymous functions.'
- Called lambda functions
- Example:

```
f = lambda x:  x**2
print f(3) #Prints 9
```

- They are very useful for embedding into other functions.
- Example:

```
rooTuples = [('eleanor', 'A', 1884),
             ('teddy', 'A', 1858),
             ('alice', 'B', 1884)]
sorted(rooTuples, key=lambda r:  r[2])
```

Lambda Functions

- Python supports 'anonymous functions.'
- Called lambda functions
- Example:

```
f = lambda x:  x**2
print f(3) #Prints 9
```

- They are very useful for embedding into other functions.
- Example:

```
rooTuples = [('eleanor', 'A', 1884),
             ('teddy', 'A', 1858),
             ('alice', 'B', 1884)]
sorted(rooTuples, key=lambda r:  r[2])
# Output:
[('teddy','A',1858), ('eleanor','A',1884), ('alice','B',1884)]
```

Lambda Functions

- Python supports 'anonymous functions.'
- Called lambda functions
- Example:

```
f = lambda x:  x**2
print f(3) #Prints 9
```

- They are very useful for embedding into other functions.
- Example:

```
rooTuples = [('eleanor', 'A', 1884),
             ('teddy', 'A', 1858),
             ('alice', 'B', 1884)]
sorted(rooTuples, key=lambda r:  r[2])
# Output:
[('teddy','A',1858), ('eleanor','A',1884), ('alice','B',1884)]
sorted(rooTuples, key= lambda r :  (r[2],r[0]))
```

Lambda Functions

- Python supports 'anonymous functions.'
- Called lambda functions
- Example:

```
f = lambda x:  x**2
print f(3) #Prints 9
```

- They are very useful for embedding into other functions.
- Example:

```
rooTuples = [('eleanor', 'A', 1884),
             ('teddy', 'A', 1858),
             ('alice', 'B', 1884)]
sorted(rooTuples, key=lambda r:  r[2])
# Output:
[('teddy','A',1858), ('eleanor','A',1884), ('alice','B',1884)]
sorted(rooTuples, key= lambda r :  (r[2],r[0]))
# Output:

[('teddy','A',1858), ('alice','B',1884), ('eleanor','A',1884)]
```

In Pairs

In pairs/triples, work out (and then try at the shell, `moreSorting.py` on webpage):

1. How do you sort a hand of cards, first by suit, then by rank? Write your algorithm in pseudocode.
2. What does the following do:

```
anon = lambda x:  x+"mouse "
print anon("a ")
print anon( anon("anony") )

add = lambda x,y:  x+y
print add("a", "mouse")
print add(3,5)

def cat(s, f):
    return f(s)

print cat("mighty", anon)
print cat("mighty",
          lambda s:  add(s,"mouse"))
```

3. What does the following do:

```
print sorted("A man a plan a canal:
       Panama".split(), key=str.lower)
```

4. What does the following do:

```
#We'll use 11 = Jack, 12 = Queen,
# 13 = King, 1 = Ace, and
# C = clubs, D = diamonds, H = hearts, and
# S = spades

cards = [(10, 'H'), (12, 'C'), (2, 'D'),
       (4, 'H'), (2, 'C'), (12, 'S'),
       (13, 'S'), (1, 'C'), (5, 'D'),
       (7, 'H'), (8,'D'), (7,'C'),
       (9,'D')]

print "Original list:", cards
print "Plain sorted:", sorted(cards)
print "Sorted by last value:", sorted(cards,
key = lambda card:  card[-1])
print "Sorted by suit, then rank:",
   sorted(cards,
          key=lambda card:(card[1],card[0]))
```

Recursion

- Recursion: a function that calls itself.

Recursion

- Recursion: a function that calls itself.
- Very useful for traversing trees and graphs (networks).

Recursion

- Recursion: a function that calls itself.
- Very useful for traversing trees and graphs (networks).
- Any function that can be written with recursion can be written with iteration (and vice versa).

Recursion

- Recursion: a function that calls itself.
- Very useful for traversing trees and graphs (networks).
- Any function that can be written with recursion can be written with iteration (and vice versa).
- Caveat: Often easier to think recursively but can have high overhead in some programming languages.
- Today's lab: running timing factorial written with recursion versus iteration:

Recursion

- Recursion: a function that calls itself.
- Very useful for traversing trees and graphs (networks).
- Any function that can be written with recursion can be written with iteration (and vice versa).
- Caveat: Often easier to think recursively but can have high overhead in some programming languages.
- Today's lab: running timing factorial written with recursion versus iteration:

```
def factR(n):
    if n ¡= 1:
        return 1
    else:
        return n*factR(n-1)
```

```
def factI(n):
    prod = 1
    for i in range(2,n):
        prod = i*prod
    return prod
```

Recap

- Recap: Built-in sorts, lambda functions, and recursion.

Recap

- Recap: Built-in sorts, lambda functions, and recursion.
- Today's lab: measuring running times of sorts and recursive functions.

Recap

- Recap: Built-in sorts, lambda functions, and recursion.
- Today's lab: measuring running times of sorts and recursive functions.
- Email lab reports to `kstjohn@amnh.org`.

Recap

- Recap: Built-in sorts, lambda functions, and recursion.
- Today's lab: measuring running times of sorts and recursive functions.
- Email lab reports to `kstjohn@amnh.org`.
- Challenges available at `rosalind.info`.