Lecture 37: Sorting/Python

CS 51G
Spring 2018
Kim Bruce

Announcements

- Test program 2 now live
  - Design due Tuesday, April 24
    - It will not be returned before program is due!
    - Keep a copy for yourself!
  - Due last day of classes
- Apples lab this Friday
  - Focus on files and strings
- Exercise 19.3.8

Merge Sort

- Divide list in half,
  - Sort first half
  - Sort second half
  - Merge two sorted halves together
  - See sort demo:
  - http://www.cs.pomona.edu/classes/cs051G/demos/SearchSort/sort.grace

Complexity of Merge Sort

- Merge two lists of total size n takes $\leq n-1$ compares
- Let $T(n)$ = # comparisons to merge sort list of size n.
- $T(0) = T(1) = 0$. Why?
- $T(n) \leq T(n/2) + T(n/2) + (n-1)$
- Claim: $T(n) < n \log_2 n$

QuickSort

- Another divide and conquer sort
  - not in sort demo Grace program
  - Move all small elements to left side of list, all large elements on left.
  - Sort small and then sort large
  - Done!
  - Also takes about n log n compares on average
    - Though worst case is roughly n^2.
    - Happens when list already sorted in either direction

Which sort when?

- Short lists (50 or fewer elements):
  - Selection sort or insertion sort are faster.
  - If partially sorted, insertion can be much faster than selection
- Long lists (50 or more)
  - QuickSort is fastest on average
    - But worst case is worse than selection/insertion
  - Merge sort always roughly n log n, so better if can’t afford long delays.
  - Merge sort takes more space (extra list of size n)

Python

Python

- Python is designed as a scripting language
  - Short programs to glue together calls to powerful libraries.
- Python is relatively slow compared to languages like Java, C, C++, etc.
  - but has highly optimized libraries written in other languages.
- Designed by BDFL Guido Van Rossum
  - Python 1 (1990), Python 2(2000), Python 3 (2008

Python Resources

- Python for Java Programmers
  - http://python4java.necaiseweb.org
- Think Python 2e (free text) *for novices*
  - http://greenteapress.com/wp/think-python-2e/

Key Points of Python

- Indenting is significant (like Grace)
  - use spaces not tabs — *don't mix them!!*
  - Line breaks are important. Statements extending onto the next line are problematic. Surround by parens so Python knows it is a continuation!
    - Can also use backslash \ at end to signal next line is continuation
- No curly braces (blocks headed with ":" instead)
- No type declarations

Running Python

- Use PyCharm CE
  - Get from Applications folder and drag to dock
    - https://www.jetbrains.com/pycharm-edu/download/
    - See on-line documentation
  - Can use interactive mode in console or
  - Write programs as usual

Getting started

- print ("hello world")
  - parentheses are required!
- count = 10
  - assignment
- count = "countString"
  - no type associated with names, can change on fly
- Comments start with #
  - x = 0 #assigns value 0 to x

Python programming

- Blocks use “:”
  - indentation counts!

```
i = 10
while i > 0:
    print(i)
    i = i - 1
print "That's it!", i

if i > 0:
    print "oops, terminated too soon!", i
elif i < 0:
    print "terminated too late", i
else:
    print 'terminated just right!', i
```

Defining functions

```
# Defines a "repeat" function that takes 2 arguments.
def repeat(s, exclaim):
    result = s + s + s
    if exclaim:
        result = result + '!!!'
    return result
```

Primitive Types

- Numbers: Integers and ﬂoating point
  - diﬀerent kinds of division
- Boolean: False, True
- String: “hello” or ‘hello’
- list: [0, 2, 4, ”hello”] — heterogeneous
- Tuple (immutable): (1,2,’a’)

Questions?