**Algorithms**

- Englishness...
- Classifying life
- Removing/Sorting and Jotto!

**HW 3**

HW #3 due Monday, 11:59
- Sound Lab!
- Several algorithms ...

---

**BR 5 Snczx**

Edward Frenkel

Is the Universe a Simulation?

FEB. 14, 2016

---

**Warm-up + cool-down from lab...**

\[2^x \text{ for } x \text{ in } L\]

\[\text{L} = [1, 11, 21]\]

\[N = \text{len(L)}\]

\[2^\text{L[i]} \text{ for } i \text{ in range(N)}\]

---

**But why use indices?!**

\[??? + x \text{ for } x \text{ in } L\]

\[\text{L} = [1, 11, 21]\]

\[\text{N} = \text{len(L)}\]

\[\text{L[i-1]} + \text{L[i]} \text{ for } i \text{ in range(N)}\]

---

**Design...**

**Top-down design**

- Visualize
- Split into parts
- Build each part
- Combine
- Test

**remAll(\(e, L\))**

- remove all \(e\)'s from \(L\)

**remAll(42, [5, 7, 42, 8, 42])**

- \([5, 7, 8]\)

**remAll('q', ['aaqqqiiqqqijqqqns'])**

- \('aliiiens'\)

---

**Use it!**

**Keep L[0]**

+ \(e\) from the rest

**Lose it!**

**Drop L[0]**

+ \(e\) from the rest
Design ~ code

```
def remAll(e, L):
    if len(L) == 0:
        return L
    elif L[0] != e:
        return L[0:1] + remAll(e, L[1:])
    else:
        return remAll(e, L[1:])
```

Top-down design
Visualize
Split into parts
Build each part
Combine
Test

Subsequences

```
def subseq(s, sbig):
    s is the subsequence to find (or not)
    sbig is the bigger string in which we are looking for s
```

```
subseq('', 'cataga') ➞ True
subseq('ctg', 'cataga') ➞ True
subseq('ctg', 'taggta') ➞ False
subseq('aliens', 'always frighten dragons') ➞ False
subseq('trogdor', 'that dragon is gone for good') ➞ True
```

other rem examples...

```
remAll(8, [7, 8, 9, 8]) ➞ [7, 9]  remAll
remOne(8, [7, 8, 9, 8]) ➞ [7, 9, 8]  remOne
remUpto(8,[7, 8, 9, 8]) ➞ [9, 8]  remUpto
remAll('d', 'coded') ➞ 'coe'  remAll
remOne('d', 'coded') ➞ 'coed'  remOne
remUpto('d', 'coded') ➞ 'ed'  remUpto
```

from remAll to remOne

```
def remAll(e, L):
    """Removes all e’s from L.""
    if len(L) == 0:
        return L
    elif L[0] != e:
        return L[0:1] + remAll(e, L[1:])
    else:
        return remAll(e, L[1:])
```

```
remOne(8, [7, 8, 9, 8]) ➞ [7, 9, 8]  remOne
remOne('d', 'coded') ➞ 'coed'
```
from \texttt{remOne} to \texttt{remUpto}

def \texttt{remOne}(e, L):
    """Returns sequence \(L\) with one \(e\) removed."
    """
    if \(\text{len}(L) = 0\):
        return \(L\)
    elif \(L[0] \neq e\):
        return \(L[0:1] + \text{remOne}(e, L[1:])\)
    else:
        return \(L[1:]\)

\texttt{remUpto}(8, \{7, 8, 9, 8\}) \Rightarrow \{9, 8\}
\texttt{remUpto}(\text{'d'}, \text{'coded'}) \Rightarrow \text{'ed'}

Subseq \sim \text{trying (coding) it out...}

def \texttt{subseq}(s, sbig):
    """Returns True if \(s\) is a subsequence of \(sbig\); False otherwise. Both are strings."""
    """
    if \(s == ''\):
        return True
    if \(s == '\text{'\textbf{.}\textbackslash'}\)"
    return True

\texttt{subseq('alg', 'magical')} \Rightarrow \text{False}
\texttt{subseq('alg', 'twobrillig')} \Rightarrow \text{True}

Subseq \sim \text{thinking it out...}

\texttt{subseq}(s, \texttt{subbig})
\texttt{subseq('ctg', 'tacggta')}

hw3pr2: \textit{use it or lose it}

Longest Common Subsequence \texttt{LCS(S, T)}

\texttt{'HUMAN'} \hspace{1cm} \texttt{'CGCTGAGCTAGGCA...'}
\texttt{'CHIMPANZEE'} \hspace{1cm} \texttt{'ATCCTAGGTAACTG...'}
also in hw4pr2: *Jotto*!

\[ \text{jscore}(S, T) \]

'robot' \hspace{1em} 'otter'

\[ \text{jscore}('robot', 'otter') \rightarrow \]

Let's try it!

**Brainstorm algorithms for these problems -- what helper functions might help each one?**

- \( \text{exact_change}(t, L) \):
  - \( \text{exact_change}(42, [25, 30, 2, 5]) \) → True
  - \( \text{exact_change}(42, [20, 16, 3, 2, 17]) \) → True
  - \( \text{exact_change}(20, [16, 3, 2, 17]) \) → False

- **sort(\(\mathbf{L}\))**
  - \( \text{sort}([42, 5, 7]) \rightarrow [5, 7, 42] \)
  - \( \text{sort}([42, 7]) \rightarrow [7, 42] \)
  - \( \text{sort}([42]) \rightarrow [42] \)
  - \( \text{sort}([]) \rightarrow [0, 0, 1] \)
  - \( \text{bisort}(1, 0, 1) \) →

- **LCS(S, T)**
  - LCS('ctga', 'tagc') → 'tga'
  - LCS('tgac', 'tagc') → 'tgac'
  - LCS('tgac', 'a') → 'a'
  - LCS('gattaca', 'ctctgcat') →

**LCS**

\[ ? \]

What was gained (or lost) here?

**Phylogeny**
def remAll(e, L):
    """Removes all e's from L.""
    if len(L) == 0:
        return L
    elif L[0] != e:
        return L[0:1] + remAll(e, L[1:])
    else:
        return remAll(e, L[1:])

def subseq(s, sbig):
    """Returns True if s is a subsequence of sbig, False otherwise. Both are strings."
    if s == '':
        return True
    elif
        subseq('alg', 'magical')  # False
        subseq('alg', 'twasbrillig')  # True