Lecture 14: Midterm Review

July 13th, 2022

Richard Roggenkemper, Laryn Qi, Cooper Bedin

Announcements

- Midterm Tomorrow
  - Seat Assignments will come out later today via email
  - Remote Exam logistics were posted to Piazza last night
- Lab & HW Due Tonight
- No Discussion, Tutoring, or Office Hours Thursday and Friday
- Instructor Office Hours Today (2-5 PM)

Lecture 14.1: Tree Recursion Review

July 13th, 2022

Laryn Qi

Review & Tips

The Essence of Tree Recursion

- **base case(s)**: usually one or more – when have I found a valid path? an invalid one (i.e. there's no possible way I can end up on a valid path)?
  - *count_stair_ways*: at the top of the staircase / stepped past the top
    - return 1 represents valid path, return 0 represents invalid path
  - *count_partitions*: successfully partitioned n fully / exceeded n with my parts OR run out of parts to use
  - *insect_combinatorics*: hit the top-right corner / gone out-of-bounds
- **recursive calls**: multiple, often each represents a choice
  - *c_s_w*: take 1 step or take 2 steps
  - *count_part*: use a part of size k or don't use any parts of size k
  - *insect_comb*: move right or move up
- **recombination**: some function or operation to construct the answer of your original problem from the answer of your subproblems
  - *c_s_w*, *count_part*, *insect_comb*: total num of ways → sum recursive calls
  - *largest_drop*: biggest difference between two digits → call max

Tips for Tree Recursion

- Structure
  - **base case**: what is the simplest input that we know the answer to? when should we stop recursing?
  - **recursive case**: what smaller inputs am I allowed to compute given my original input (*decision-making*)? how can i use their solution to solve my input (*recursive leap of faith*)?
- Consider **domain**, **range**, and **intended behavior** of your function
  - **domain**: what type(s) and set of values can you take in? the edge(s) of your domain could be your base case(s)
  - **range**: most recursive functions return the same type of value in all cases
    - verify that all return statements have matching return types
  - **intended behavior**: if I were given this problem, how would I solve it? → how can I formalize my steps into general instructions a computer could follow?
- Parse the skeleton
  - What is the high-level purpose of each blank line? Of each variable?
- Problem-solving
  - partial solutions can lead you to the full solution (jigsaw puzzle approach)
  - Write down what you know first → what else can you fill out given that the other blanks have been filled?

Fall 2019 Final Q6a-c

Lecture 14.2: Lists and Mutability Review

July 13th, 2022

Cooper Bedin

Quick Review

What's important to know about lists?

You should know:

- How to construct a new list
- How to index elements out of a list
- How to mutate a list by indexing
- How to take a slice of a list
- How to write a list comprehension, and what they do
- You should be familiar with all the list mutation operations—you don't have to memorize them all, because they'll be on the study guide, but you should feel like you've seen each of them and know what they do
  - `append`, `extend`, `pop`, `remove`, `insert`
- What operations create a new list, and which ones mutate an existing list
- How to represent lists in environment diagrams

Making a copy vs mutating

These will create an entirely new list:

- Taking any slice of a list
  - `a[1:3]`
- Writing a list comprehension
- Concatenating lists
  - `a = a + [3, 2]`

These will mutate a list that already exists

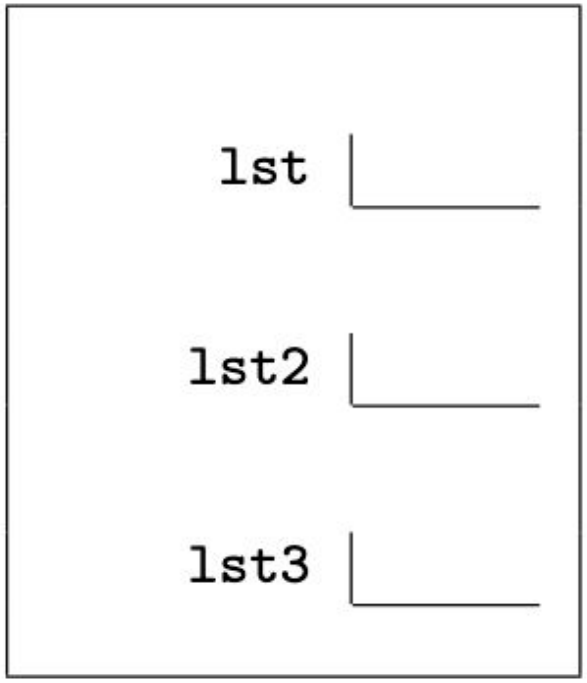
- Any of the mutation functions (see previous slide)
- Bracketing on the right side of an assignment statement
  - `a[0] = 3`
  - `a[1:3] = [3, 4, 5]`
  - **not** `a = [3, 4, 5]`
  - `a += [3, 4, 5]` is a special case in which mutation actually does occur

Lists and Mutation in Environment Diagrams

Su19 MT Q3a

```
lst = [2, 4, lambda: lst]
lst2 = lst
lst = lst[2:]
lst3 = lst[0]()
```

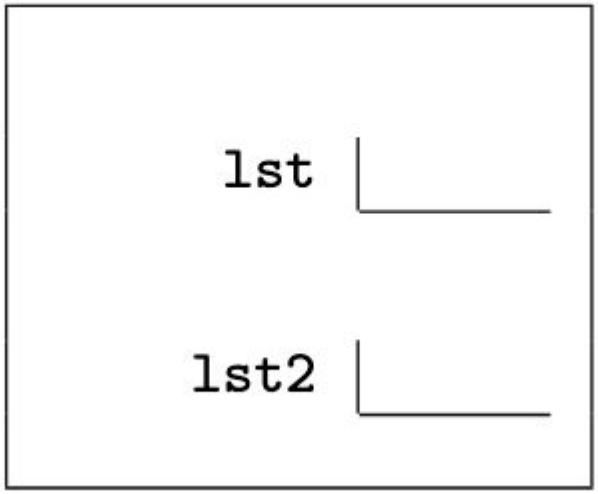
PythonTutor



Su19 MT Q3b

```
lst = [[5], 2, 4, 10]
lst2 = lst[1:3] + lst[:3]
for n in lst2[:2]:
    lst.append(lst2[n])
```

PythonTutor



Su19 MT Q3c

```
lst = ['goodbye', 0, None, 8, 'hello', 1]
while lst.pop():
    x = lst.pop()
    if x:
        lst.pop()
    else:
        lst.append('three')
```

PythonTutor

List Practice Problem

Fa19 MT2 Q6

Definition. A *switch list* `r` for two source lists `s` and `t`, both of length n, is a list where each element `r[i]` for `0 ≤ i < n` is either `s[i]` or `t[i]`. The switch count for `r` is the number of indices `i` for which `r[i]` and `r[i-1]` come from different lists. As a special case, index `0` contributes `0` to the switch count if `r[0]` comes from `s` and 1 if it comes from `t`.

Implement `switch`, which takes two lists of numbers `s` and `t` that have the same length, and a non-negative integer `k`. It returns the switch list for `s` and `t` that has the largest sum and has a switch count of at most `k`.

Fa19 MT2 Q6

```
def switch(s, t, k):
    """Return the list with the largest sum built by switching
    between S and T at most K times.
    >>> switch([1, 2, 7], [3, 4, 5], 0)
    [1, 2, 7]
    >>> switch([1, 2, 7], [3, 4, 5], 1)
    [3, 4, 5]
    >>> switch([1, 2, 7], [3, 4, 5], 2)
    [3, 4, 7]
    >>> switch([1, 2, 7], [3, 4, 5], 3)
    [3, 4, 7]
    """
1   if k == 0 or len(s) == 0:
2       return s
3   else:
4       a = switch(_____, _____, k-1)
5       b = __________
6       return max(a, b, key=sum)
```

Constraints:

- The blanks on line 4 must be filled with one of the following combinations:
  - `s, t`
  - `t, s`
  - `s[1:], t`
  - `t, s[1:]`
  - `s[1:], t[1:]`
  - `t[1:], s[1:]`
- The blank on line 5 must use one of the following templates:
  - `_______ + switch(_______, _______, k - 1)`
  - `_______ + switch(_______, _______, k)`
  - `switch(_______, _______, k - 1)`
  - `switch(_______, _______, k)`

Fa19 MT2 Q6 - Solution

```
def switch(s, t, k):
    """Return the list with the largest sum built by switching
    between S and T at most K times.
    >>> switch([1, 2, 7], [3, 4, 5], 0)
    [1, 2, 7]
    >>> switch([1, 2, 7], [3, 4, 5], 1)
    [3, 4, 5]
    >>> switch([1, 2, 7], [3, 4, 5], 2)
    [3, 4, 7]
    >>> switch([1, 2, 7], [3, 4, 5], 3)
    [3, 4, 7]
    """
1   if k == 0 or len(s) == 0:
2       return s
3   else:
4       a = switch(_____, _____, k-1)
5       b = __________
6       return max(a, b, key=sum)
```

Constraints:

- The blanks on line 4 must be filled with one of the following combinations:
  - [redacted]
  - [redacted]
  - [redacted]
  - `t, s[1:]`
  - [redacted]
  - [redacted]
- The blank on line 5 must use one of the following templates:
  - `_______ + switch(_______, _______, k - 1)`
  - `_______ + switch(_______, _______, k)`
  - [redacted]
  - [redacted]

Fa19 MT2 Q6 - Solution

```
def switch(s, t, k):
    """Return the list with the largest sum built by switching
    between S and T at most K times.
    >>> switch([1, 2, 7], [3, 4, 5], 0)
    [1, 2, 7]
    >>> switch([1, 2, 7], [3, 4, 5], 1)
    [3, 4, 5]
    >>> switch([1, 2, 7], [3, 4, 5], 2)
    [3, 4, 7]
    >>> switch([1, 2, 7], [3, 4, 5], 3)
    [3, 4, 7]
    """
1   if k == 0 or len(s) == 0:
2       return s
3   else:
4       a = switch(t, s[1:], k-1)
5       b = ___________
6       return max(a, b, key=sum)
```

Constraints:

- The blanks on line 4 must be filled with one of the following combinations:
  - [redacted]
  - [redacted]
  - [redacted]
  - `t, s[1:]`
  - [redacted]
  - [redacted]
- The blank on line 5 must use one of the following templates:
  - `_______ + switch(_______, _______, k - 1)`
  - `_______ + switch(_______, _______, k)`
  - [redacted]
  - [redacted]

Fa19 MT2 Q6 - Solution

```
def switch(s, t, k):
    """Return the list with the largest sum built by switching
    between S and T at most K times.
    >>> switch([1, 2, 7], [3, 4, 5], 0)
    [1, 2, 7]
    >>> switch([1, 2, 7], [3, 4, 5], 1)
    [3, 4, 5]
    >>> switch([1, 2, 7], [3, 4, 5], 2)
    [3, 4, 7]
    >>> switch([1, 2, 7], [3, 4, 5], 3)
    [3, 4, 7]
    """
1   if k == 0 or len(s) == 0:
2       return s
3   else:
4       a = switch(t, s, k-1)
5       b = ___________
6       return max(a, b, key=sum)
```

Constraints:

- The blanks on line 4 must be filled with one of the following combinations:
  - `s, t`
  - `t, s`
  - `s[1:], t`
  - `t, s[1:]`
  - `s[1:], t[1:]`
  - `t[1:], s[1:]`
- The blank on line 5 must use one of the following templates:
  - `_______ + switch(_____, _____, k - 1)`
  - `_______ + switch(_____, _____, k)`
  - `switch(_____, _____, k - 1)`
  - `switch(_____, _____, k)`

Fa19 MT2 Q6 - Solution

```
def switch(s, t, k):
    """Return the list with the largest sum built by switching
    between S and T at most K times.
    >>> switch([1, 2, 7], [3, 4, 5], 0)
    [1, 2, 7]
    >>> switch([1, 2, 7], [3, 4, 5], 1)
    [3, 4, 5]
    >>> switch([1, 2, 7], [3, 4, 5], 2)
    [3, 4, 7]
    >>> switch([1, 2, 7], [3, 4, 5], 3)
    [3, 4, 7]
    """
1   if k == 0 or len(s) == 0:
2       return s
3   else:
4       a = switch(t, s, k-1)
5       b = _____ + switch(_____, _____, k)
6       return max(a, b, key=sum)
```

Constraints:

- The blanks on line 4 must be filled with one of the following combinations:
  - `s, t`
  - `t, s`
  - `s[1:], t`
  - `t, s[1:]`
  - `s[1:], t[1:]`
  - `t[1:], s[1:]`
- The blank on line 5 must use one of the following templates:
  - `_____ + switch(_____, _____, k - 1)`
  - `_____ + switch(_____, _____, k)`
  - `switch(_____, _____, k - 1)`
  - `switch(_____, _____, k)`

Fa19 MT2 Q6 - Solution

```
def switch(s, t, k):
    """Return the list with the largest sum built by switching
    between S and T at most K times.
    >>> switch([1, 2, 7], [3, 4, 5], 0)
    [1, 2, 7]
    >>> switch([1, 2, 7], [3, 4, 5], 1)
    [3, 4, 5]
    >>> switch([1, 2, 7], [3, 4, 5], 2)
    [3, 4, 7]
    >>> switch([1, 2, 7], [3, 4, 5], 3)
    [3, 4, 7]
    """
1   if k == 0 or len(s) == 0:
2       return s
3   else:
4       a = switch(t, s, k-1)
5       b = [s[0]] + switch(s[1:], t[1:], k)
6       return max(a, b, key=sum)
```

Constraints:

- The blanks on line 4 must be filled with one of the following combinations:
  - `s, t`
  - `t, s`
  - `s[1:], t`
  - `t, s[1:]`
  - `s[1:], t[1:]`
  - `t[1:], s[1:]`
- The blank on line 5 must use one of the following templates:
  - `_____ + switch(_____, _____, k - 1)`
  - `_____ + switch(_____, _____, k)`
  - `switch(_____, _____, k - 1)`
  - `switch(_____, _____, k)`