Final Examples

Announcements

Trees

Tree-Structured Data

Implement `bigs`, which takes a tree instance `t` containing integer labels. It returns the number of nodes in `t` whose labels are larger than any labels of their ancestor nodes.

```python
def bigs(t):
    """Return the number of nodes in `t` that are larger than all their ancestors.
    >>> a = Tree(1, [Tree(4, [Tree(4), Tree(5)]), Tree(3, [Tree(0, [Tree(2)])])])
    >>> bigs(a)
    4
    """
    def f(a, x):
        if node.label > max_ancestors:
            return 1 + sum(f(b, node.label) for b in a.branches)
        else:
            return sum(f(b, x) for b in a.branches)
    return f(t, 0)
```

A tree can contain other trees:

```
[5, [6, 7], 8, [[9], 10]]
```

```
S (NP (JJ Short) (NNS cuts))
```

```
VP (VBP make) (NP (JJ long) (NNS delays))
```

```
ul
<li><b>Midterm</b> 1</li>
<li><b>Midterm</b> 2</li>
</ul>
```

Tree processing often involves recursive calls on subtrees:

Recursive Accumulation

Solving Tree Problems
Function Template

Work through examples that illustrate the function's purpose.

Functional Examples

Translate the data definitions into an outline of the function.

Function Definition

Fill in the gaps in the function template. Exploit the purpose statement and the examples.

Signature, Purpose Statement, Header

Designing a Function

Implement `bigs`, which takes a Tree instance `t` containing integer labels. It returns the number of nodes in `t` whose labels are larger than any labels of their ancestor nodes.

```python
def bigs(t):
    """Return the number of nodes in t that are larger than all their ancestors."""
    n = 0
    def f(a, x):
        nonlocal n
        if a.is_leaf():
            return
        else:
            n += 1
            for b in a.branches:
                if b.label < min(a.label, x):
                    n += 1
                f(b, x)

    f(t, t.label)
    return n
```

Implement `smallss`, which takes a Tree instance `t` containing integer labels. It returns the non-leaf nodes in `t` whose labels are smaller than any labels of their descendant nodes.

```python
def smallss(t):
    """Return the non-leaf nodes in t that are smaller than all their descendants."
    def f(a, x):
        nonlocal n
        if a.is_leaf():
            return
        else:
            n = 1
            for b in a.branches:
                if b.label > max(a.label, x):
                    n += 1
                f(b, x)

    f(t, t.label)
    return n
```

How to Design Programs

From Problem Analysis to Data Definitions

Identify the information that must be represented and how it is represented in the chosen programming language. Formulate data definitions and illustrate them with examples.

Signature, Purpose Statement, Header

State what kind of data the desired function consumes and produces. Formulate a concise answer to the question what the function computes. Define a stub that lives up to the signature.

Functional Examples

Work through examples that illustrate the function's purpose.

Function Template

Translate the data definitions into an outline of the function.

Function Definition

Fill in the gaps in the function template. Exploit the purpose statement and the examples.

Testing

Articulate the examples as tests and ensure that the function passes all. Doing so discovers mistakes. Tests also supplement examples in that they help others read and understand the definition when the need arises-and it will arise for any serious program.

Privacy Policies and Laws

Mark Zuckerberg in San Francisco, January 8, 2010

"People have really gotten comfortable not only sharing more information and different kinds, but more openly and with more people. That social norm is just something that has evolved over time."

Tim Cook in Brussels, October 24, 2018

"We at Apple are in full support of a comprehensive federal privacy law in the United States. Here, and everywhere, it should be rooted in four essential rights:

- First, the right to have personal data minimized. Companies should challenge themselves to de-identify customer data, or not to collect it in the first place.
- Second, the right to knowledge. Users should always know what data is being collected and what it is being collected for. This is the only way to empower users to decide what collection is legitimate and what isn’t. Anything less is a sham.
- Third, the right to access. Companies should recognize that data belongs to users, and we should all make it easy for users to get a copy of, correct, and delete their personal data.
- And Fourth, the right to security. Security is foundational to trust and all other privacy rights."
Perils of Sharing

A persistent source of privacy breaches: sending a message to an unintended recipient

Automated Decision Making

What should the self-driving car do?