Infinity (and beyond)

Help Center

Often times when constructing algorithms we want to set variables to a symbolic value to signify a special state in our algorithm. For example, if we want to compute the distance between two nodes in a graph, what should that value be when no path connects them? 0? -1? `None`? Infinity? Each has advantages and drawbacks. Consider the case of finding the minimum value in a list of numbers:

```
def find_minimum(list_of_numbers):
    smallest_seen = None
    for i in list_of_numbers:
        if i < smallest_seen:
            smallest_seen = i

    return smallest_seen
```

What do you expect the output of `find_minimum([2, 3, 2, 1, 5])` to be? Interestingly enough, this invocation yields `None`. Why? Because in Python `None < 1` is true. So what first seems like a logical choice (we have never seen any value yet, so clearly the smallest seen number is nonexistent) can turn out to be problematic in our implementation. Great care and thought must be taken when selecting these symbolic values as their choice may impact your program's correctness.

```
def find_minimum(list_of_numbers):
    smallest_seen = None
    for i in list_of_numbers:
        if (smallest_seen == None) or (i < smallest_seen):
            smallest_seen = i

    return smallest_seen
```

Using `None` therefore required an awkward check that must occur every time through the loop, even though `smallest_seen` will never be `None` after the first iteration. So, what value then should we initially assign to `smallest_seen`? Infinity seem like a good choice. Anything inside the list must be smaller than infinity (unless infinity appears in the list. Is that an issue for this algorithm?). We can represent infinity in Python using `float('inf')`:

```
def find_minimum(list_of_numbers):
    smallest_seen = float('inf')
    for i in list_of_numbers:
        if i < smallest_seen:
            smallest_seen = i

    return smallest_seen
```

Now `find_minimum([2, 3, 2, 1, 5])` produces `1`. What though is the output of `find_minimum([])` and `find_minimum([float('inf')])` ? Both are `float('inf')` so how do we know if the input list was empty or contained only one element? As you can see, no implementation is perfect. The important part though is to know the limitations of your implementation, understand their impact on your algorithm and compensate for those cases (for example, we could check for a zero length list before calling `find_minimum`).

Created Sun 24 Aug 2014 12:42 AM BST

Last Modified Sun 24 Aug 2014 4:20 AM BST