Daily Coding Problem #134

Problem

This problem was asked by Facebook.

You have a large array with most of the elements as zero.

Use a more space-efficient data structure, SparseArray, that implements the same interface:

- `init(arr, size)`: initialize with the original large array and size.
- `set(i, val)`: updates index at `i` with `val`.
- `get(i)`: gets the value at index `i`.

Solution

Since the original array is mostly zeroes, we should be able to manage our array with a lot less space by only keeping track of the non-zero values and indices. We can use a dictionary to keep track of those, and default to zero when the key is not found in our dictionary.

Remember to also check the bounds when setting or getting `i`, and to clean up any indices if we're setting an index to zero again.

```
class SparseArray:
    def __init__(self, arr, n):
        self.n = n
        self._dict = {}
        for i, e in enumerate(arr):
            if e != 0:
                self._dict[i] = e

    def _check_bounds(self, i):
        if i < 0 or i >= self.n:
            raise IndexError('Out of bounds')
```

```
    def set(self, i, val):
        self._check_bounds(i)
        if val != 0:
            self._dict[i] = val
            return
        elif i in self._dict:
            del self._dict[i]

    def get(self, i):
        self._check_bounds(i)
        return self._dict.get(i, 0)
```

This will use as much space as there are non-zero elements in the array.