Data Structures

Probing Implementation

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

PhoneHashTable DS

- We will use a single list to maintain the elements
- Use a private object to mark deleted locations

```
class OurDictPropbing:
    _DELETED_MARK = object()

    def __init__(self, table_size):
        self.table_size = table_size
        self.table = [None] * table_size
        self.total_elements = 0
```

Useful utility

- Develop a function that takes a key and search the table and return one of 2 cases
- 1) Item found
  - Return its index
- 2) Item NOT found
  - Return the first available index (can be empty or deleted)
- We then use this utility to code the hash table trivially
- Try to code it

```python
# Critical function - commonly wrongly implemented
# Don't just return the first of: available or found
# First make sure it is NOT found, before returning first valid idx
def _find_idx(self, key):
    hkey = hash(key) % self.table_size
    first_available = None

    for step in range(self.table_size):
        item = self.table[hkey]
        if item is None or item == self._DELETED_MARK:
            if first_available is None:
                first_available = hkey

            if item is None:
                break   # We are done with the block

        elif item[0] == key:
            return hkey, True

        hkey = (hkey + 1) % self.table_size  # move 1 step (linear proping)

    # NOT found. Here is an insertion slot
    return first_available, False
```

```
def add(self, key, value):
    assert self.total_elements < self.table_size, 'Table is FULL'
    hkey, found = self._find_idx(key)
    self.table[hkey] = [key, value]
    self.total_elements += not found
    print(self.total_elements, key)

def get(self, key):
    hkey, found = self._find_idx(key)
    assert found, f'No such item {key}'
    return self.table[hkey][1]

def exists(self, key):
    hkey, found = self._find_idx(key)
    return found

def remove(self, key):
    hkey, found = self._find_idx(key)
    if found:
        self.table[hkey] = self._DELETED_MARK
        self.total_elements -= 1
    return found
```

```
dct = OurDictPropbing(table_size=9)

dct.add('Mostafa', 1)
dct.add('Ziad', 2)
dct.add('Ali', 5)
dct.add('Belal', 10)
dct.add('Ashraf', 4)
dct.add('Ziad', 555)    # reassign

dct.print()
'''
0: ['Belal', 10]
1: ['Ashraf', 4]
2: E
3: ['Ziad', 555]
4: E
5: E
6: E
7: ['Mostafa', 1]
8: ['Ali', 5]
'''
```

```
dct.remove('Belal')
dct.remove('Ali')
dct.print()
'''
0: D
1: ['Ashraf', 4]
2: E
3: ['Ziad', 555]
4: E
5: E
6: E
7: ['Mostafa', 1]
8: D
'''
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."