Hash Tables

A real-world example: let's say I stay at a hotel for a few days for a vacation. I inquire if there are any messages for me at the front desk when I go back to the hotel at the end of the day. There's a cabinet behind the receptionist which has 26 compartments behind it. She removes 2 letters from the space marked M since she knows my first name. One is for me and one is for some other person whose first name starts with M. Only 2 letters needed to be read by the receptionist. If there had been just one mailbox, how many letters would she have to check?

In [1]:
```python
import random
import string

def create_empty_hash_table():
    """Creates hashtable with 26 empty buckets"""
    return [[] for x in range(26)]

def hashfunc(key):
    """Takes a key as an input and returns the appropriate index"""
    wordtoindex = {"a": 0, "b": 1, "c": 2, "d": 3, "e": 4, "f": 5, "g": 6, "h": 7, "i": 8, "j": 9,
                  "k": 10 , "l": 11, "m": 12, "n": 13, "o": 14, "p": 15, "q": 16, "r": 17, "s": 18,
                   "t": 19, "u": 20, "v": 21, "w": 22, "x": 23, "y":24, "z":25}
    #or we can use ord(c[0])-ord('a')
    return wordtoindex[key[0].lower()]

def htable_put(table, key, value):
    """Inserts (name, age) tuple in appropriate buckets"""
    bucket = table[hashfunc(key)]
    bucket.append((key, value))

def populate_hashtable(table, name_age_list):
    """Iterates through a list of (name, age) tuples and places them in appropriate buckets"""
    for name, age in name_age_list:
        htable_put(table, name, age)
    return table

def htable_search(table, key):
    """Searches a specific bucket in the hashtable and returns (name, age) tuple"""
    bucket = table[hashfunc(key)]
    return linear_search(bucket, key)

def linear_search(name_list, key):
    """Basic linear search which searches a list of names and returns (name, age) tuple"""
    for name, age in name_list:
        if name == key:
            return (name, age)

def read_names():
    """Reads names from names.txt and returns a list of (name, age) tuple"""
    with open('names.txt') as f:
        names = f.read().split('\n')
    return [(name, random.randint(1, 100)) for name in names]
```

In [6]:
```python
name_age_list = read_names()
name_to_search = name_age_list[-random.randint(1,99)][0]
name_to_search
```

Out[6]:
```
'Halle'
```

In [3]:
```python
%%time
tym = [linear_search(name_age_list, name_to_search) for i in range(50000)]
```

```
CPU times: user 1.69 s, sys: 11 ms, total: 1.7 s
Wall time: 1.7 s
```

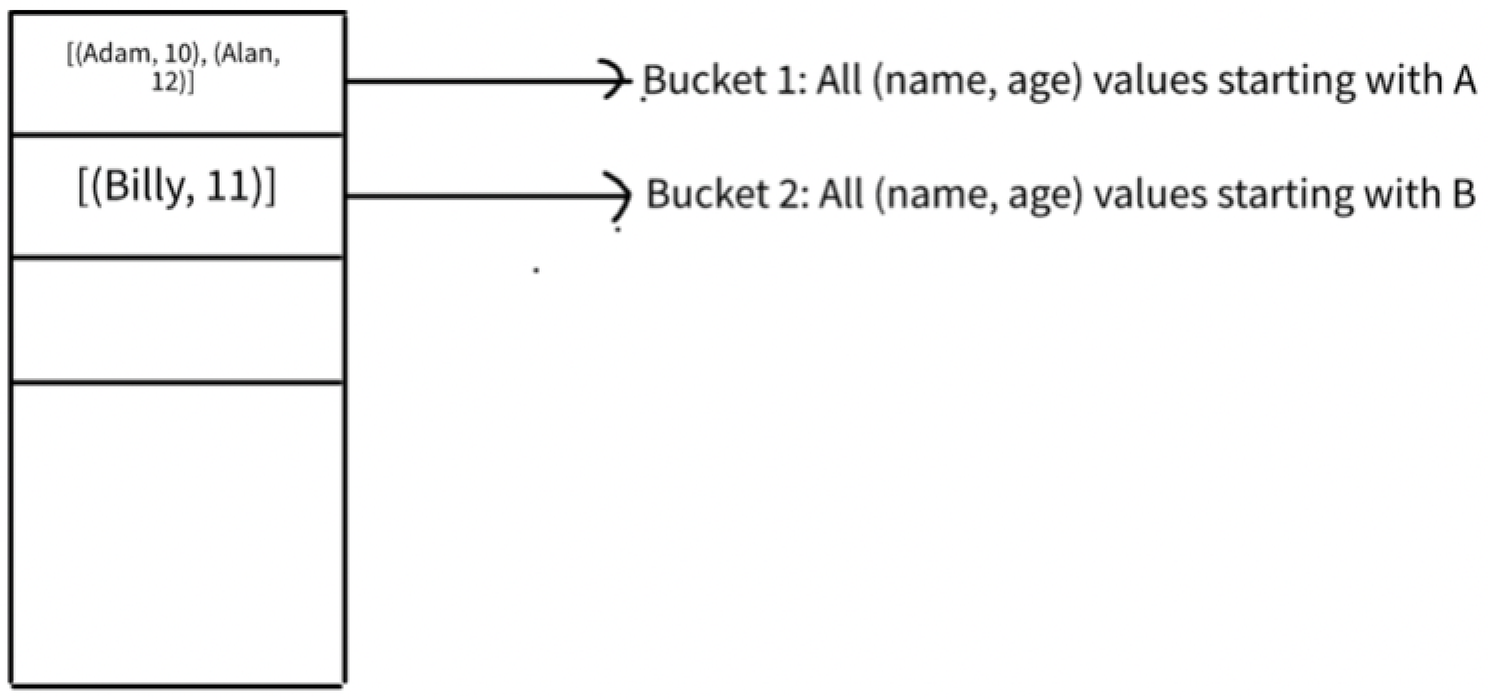
[(Adam, 10), (Alan, 12)]
[(Billy, 11)]
Bucket 1: All (name, age) values starting with A
Bucket 2: All (name, age) values starting with B

Fig: A hashtable divided into buckets.

In [4]:
```python
table = create_empty_hash_table()
table = populate_hashtable(table, name_age_list)
```

In [5]:
```python
%%time
tym2= [htable_search(table, name_to_search) for i in range(50000)]
```

```
CPU times: user 135 ms, sys: 3.93 ms, total: 138 ms
Wall time: 137 ms
```