CS 2302 Data Structures - Dictionaries

Author:
Olac Fuentes
Computer Science Department
University of Texas at El Paso
Last modified: 12/21/2022

Sets allow us to perform insertions, deletions, and membership queries in constant time.

Dictionaries are another data stucture that uses hash tables to perform operations efficiently.

A dictionary is a Python container used to describe associative relationships. A dictionary is represented by the dict object type. A dictionary associates (or "maps") keys with values. A key is a term that can be located in a dictionary, such as the word "cat" in the English dictionary. A value describes some data associated with a key, such as a definition. A key can be any immutable type, such as a integer, floating point number, or string; a value can be any type.

Building a dictionary

We can create an empty dictionary as follows:

```
D= {}
```

To add a key, value pair, to the dictionary, we do: D[key] = value

```
D[2302] = 'Python'
D[2401] = 'Java'
D[1301] = 'Java'
```

```
D
```

```
{2302: 'Python', 2401: 'Java', 1301: 'Java'}
```

```
D['UTEP'] = 123344
```

```
D['UTEP']
```

```
123344
```

keys must be unique but two different keys may have the same value, as shown above.

If we assign a value to a key that is already in the dictionary, its value is changed.

```
D[1301] = 'C'
```

```
D
```

```
{2302: 'Python', 2401: 'Java', 1301: 'C', 'UTEP': 123344}
```

```
D['CS']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-9-5c485601a563> in <module>
----> 1 D['CS']

KeyError: 'CS'
```

SEARCH STACK OVERFLOW

We can also access the values in a dictionary from a key value usiong the get function.

```
for k in [2302, 2401, 1301, 'UTEP']:
  print(k, D.get(k))
```

```
2302 Python
2401 Java
1301 C
UTEP 123344
```

An advantage of the *get* function is that you can specify a default value to return if the specified key does not exist, which by default is *None*.

```
for k in [2302, 2401, 1301, 'UTEP','Miners']:
  print(k, D.get(k))
```

```
2302 Python
2401 Java
1301 C
UTEP 123344
Miners None
```

```
for k in [2302, 2401, 1301, 'UTEP','Miners']:
  print(k, D.get(k,'Pascal'))
```

```
2302 Python
2401 Java
1301 C
UTEP 123344
Miners Pascal
```

We can retrieve the keys in a dictionary using the keys function

```
D.keys()
```

```
dict_keys([2302, 2401, 1301, 'UTEP'])
```

We can test if a key is in a dictionary in the same way we check for list and set membership.

```
2302 in D.keys()
```

```
True
```

```
3331 in D.keys()
```

```
False
```

We remove a key,value pair from a dictionary using the del operation

---

```
del D[2302]
```

```
D
```

```
{2401: 'Java', 1301: 'C', 'UTEP': 123344}
```

If we try to delete a key that is not in the dictionary, we get an error.

```
del D[2302]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-18-cba506733b78> in <module>
----> 1 del D[2302]

KeyError: 2302
```

SEARCH STACK OVERFLOW

We can build dictionaries using comprehension:

```
square = {i:i**2 for i in range(13)}
print(square)
```

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121, 12: 144}
```

```
for i in range(13):
  print(i, square[i])
```

```
0 0
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
11 121
12 144
```

```
square = {i:i**2 for i in [1,2,4,8,16,32,64]}
print(square)
```

```
{1: 1, 2: 4, 4: 16, 8: 64, 16: 256, 32: 1024, 64: 4096}
```

```
square = {i:i**2 for i in [9,1,2,6,4,5,0,7,8] if i%2 == 0}
print(square)
```

```
{2: 4, 6: 36, 4: 16, 0: 0, 8: 64}
```

Like set operations, dictionary operations use hashing, thus they are very efficient.

```
import time
import numpy as np
```

```
n=50000
L = np.random.permutation(n)

start = time.time()
for i in range(2*n):
  t = i in L # Check membership and store in variable
elapsed_time1 = time.time() - start
print('elapsed time using list', elapsed_time1,'secs')

D = {i:' ' for i in L}

start = time.time()
for i in range(2*n):
  t = i in D.keys()  # Check membership and store in variable
elapsed_time2 = time.time() - start
print('elapsed time using dictionary', elapsed_time2,'secs')

print('ratio',elapsed_time1/elapsed_time2)
```

```
elapsed time using list 2.632946252822876 secs
elapsed time using dictionary 0.04049515724182129 secs
ratio 65.0187931633392
```

Example: The following code constructs a dictionary where a country's name is the key and its capital is the value.

```
countries = ['Russia','Canada', 'USA', 'Brazil', 'Australia', 'China','Spain','France']
capitals = ['Moscow','Ottawa', 'Washington', 'Brasilia', 'Canberra', 'Beijing','Madrid','Paris']

cap_dict ={} # Create empty dictionary

for i in range(len(countries)):
  cap_dict[countries[i]] = capitals[i]
```

```
cap_dict
```

```
{'Russia': 'Moscow',
 'Canada': 'Ottawa',
 'USA': 'Washington',
 'Brazil': 'Brasilia',
 'Australia': 'Canberra',
 'China': 'Beijing',
 'Spain': 'Madrid',
 'France': 'Paris'}
```

Now to retrieve the value given the key, we do the following:

```
print(cap_dict['France'])
```

```
Paris
```

```
print(cap_dict['Russia'])
```

```
Moscow
```

```
cap_dict.keys()
```

```
dict_keys(['Russia', 'Canada', 'USA', 'Brazil', 'Australia', 'China', 'Spain', 'France'])
```

Example: Write a function that receives a list and builds a dictionary containing the position in the list where every item first appeared.

```
def first_occurrence(L):
  D ={}
  for i, item in enumerate(L):
    if not item in D:
      D[item] = i
  return D
```

```
import numpy as np
np.random.seed(0)
L = list(np.random.randint(0,10,size=30))
print(L)

D = first_occurrence(L)
for i in set(L):
  print('first occurrence of {} at index {}'.format(i,D[i]))
```

```
[5, 0, 3, 3, 7, 9, 3, 5, 2, 4, 7, 6, 8, 8, 1, 6, 7, 7, 8, 1, 5, 9, 8, 9, 4, 3, 0, 3, 5, 0]
first occurrence of 0 at index 1
first occurrence of 1 at index 14
first occurrence of 2 at index 8
first occurrence of 3 at index 2
first occurrence of 4 at index 9
first occurrence of 5 at index 0
first occurrence of 6 at index 11
first occurrence of 7 at index 4
first occurrence of 8 at index 12
first occurrence of 9 at index 5
```

Example: Write a function that receives a list and builds a dictionary containing the position in the list where every item last appeared.

```
def last_occurrence(L):
  D ={}
  for i, item in enumerate(L):
    D[item] = i
  return D
```

```
np.random.seed(0)
L = list(np.random.randint(0,10,size=30))
print(L)

D = last_occurrence(L)
for i in set(L):
  print('last occurrence of {} at index {}'.format(i,D[i]))
```

```
[5, 0, 3, 3, 7, 9, 3, 5, 2, 4, 7, 6, 8, 8, 1, 6, 7, 7, 8, 1, 5, 9, 8, 9, 4, 3, 0, 3, 5, 0]
last occurrence of 0 at index 29
last occurrence of 1 at index 19
last occurrence of 2 at index 8
last occurrence of 3 at index 27
last occurrence of 4 at index 24
last occurrence of 5 at index 28
last occurrence of 6 at index 15
last occurrence of 7 at index 17
last occurrence of 8 at index 22
last occurrence of 9 at index 23
```

Example: Write a function that receives a list and builds a dictionary containing the position where each unique item in the list appears.

For example, if L = [20,30,20,50,60,30,10,50], D[20] should be the list [0, 2], since 20 appears in positions 0 and 2 in L.

```
def all_occurrences(L):
  D = {item:[] for item in set(L)}
  for i, item in enumerate(L):
    D[item].append(i)
  return D


np.random.seed(0)
L = list(np.random.randint(0,10,size=30))
print(L)

D = all_occurrences(L)
for i in set(L):
  print('all occurrences of {}:'.format(i),D[i])
```

```
[5, 0, 3, 3, 7, 9, 3, 5, 2, 4, 7, 6, 8, 8, 1, 6, 7, 7, 8, 1, 5, 9, 8, 9, 4, 3, 0, 3, 5, 0]
all occurrences of 0: [1, 26, 29]
all occurrences of 1: [14, 19]
all occurrences of 2: [8]
all occurrences of 3: [2, 3, 6, 25, 27]
all occurrences of 4: [9, 24]
all occurrences of 5: [0, 7, 20, 28]
all occurrences of 6: [11, 15]
all occurrences of 7: [4, 10, 16, 17]
all occurrences of 8: [12, 13, 18, 22]
all occurrences of 9: [5, 21, 23]
```