Lect 21 – Performance of Python Data Structures

Rob Capra

INLS 490-172

Add to a list

```
from timeit import Timer

def test1():
    l = []
    for i in range(1000):
        l = l + [i]

def test2():
    l = []
    for i in range(1000):
        l.append(i)

def test3():
    l = [i for i in range(1000)]

def test4():
    l = list(range(1000))

t1 = Timer("test1()", "from __main__ import test1")
print "concat ",t1.timeit(number=1000), "ms"
t2 = Timer("test2()", "from __main__ import test2")
print "append ",t2.timeit(number=1000), "ms"
t3 = Timer("test3()", "from __main__ import test3")
print "comprehension ",t3.timeit(number=1000), "ms"
t4 = Timer("test4()", "from __main__ import test4")
print "list range ",t4.timeit(number=1000), "ms"
```

Table 2: Big-O Efficiency of Python List Operators

Operation	Big-O Efficiency
index []	O(1)
index assignment	O(1)
append	O(1)
pop()	O(1)
pop(i)	O(n)
insert(i,item)	O(n)
del operator	O(n)
iteration	O(n)
contains (in)	O(n)
get slice [x:y]	O(k)
del slice	O(n)

Dictionaries

Table 3: Big-O Efficiency of Python Dictionary Operations

operation	Big-O Efficiency
copy	O(n)
get item	O(1)
set item	O(1)
delete item	O(1)
contains (in)	O(1)
iteration	O(n)

Dictionaries vs. Lists (contains)

```
import timeit
import random

for i in range(10000,1000001,20000):
    t = timeit.Timer("random.randrange(%d) in x"%i,
                     "from __main__ import random,x")
    x = list(range(i))
    lst_time = t.timeit(number=1000)
    x = {j:None for j in range(i)}
    d_time = t.timeit(number=1000)
    print("%d,%10.3f,%10.3f" % (i, lst_time, d_time))
```

Output:

```
10000, 0.113, 0.001
30000, 0.366, 0.001
50000, 0.609, 0.001
70000, 0.904, 0.001
90000, 1.129, 0.002
110000, 1.399, 0.001
130000, 1.724, 0.004
```

Dictionaries vs. Lists
(contains)