CSC148 winter 2016

hashes, memory, review week 12

Danny Heap heap@cs.toronto.edu BA4270 (behind elevators)

http://www.cdf.toronto.edu/~csc148h/winter/ 416-978-5899

April 8, 2016





Outline

hash tables

tracing code

aliasing

summary/review



why hash

lists are contiguous (adjacent) sequences of references to objects, so access to a list position is fast (just arithmetic)

what if we could convert — hash — other data to a suitable integer for a list index, we'd want:

- ▶ fast
- ▶ deterministic: the same (or equivalent values) gets hashed to the same integer each time.
- well-distributed: We'd like a typical set of values to get hashed pretty uniformly over the available list positions.



you can't hash everything!

```
>>> list1 = [0]
>>> id(list1)
3069263116
>>> list1.append(1)
>>> id(list1)
3069263116
```

oops!



hash to hash table (dictionary)...

Once you have hashed an object to a number, you can easily use part of that number as an index into a list to store the object, or something related to that object. If the list is of length n, you might store information about object o at index hash(o) % n.

collisions

even a well-distributed hash function will have a surprising number of collisions...

how many people do you need to poll before you find two with the same birthday (out of 366 possibilities, including leap-year)?

the mathematics is a bit counter-intuitive... the probability of a non-collision for 23 birthdays is:

$$p = \frac{366}{366} \times \frac{365}{366} \times \cdots \times \frac{344}{366} \approx 0.493$$





chaining or probing

a couple of tactics for dealing with two different keys ending up at the same index

chaining: keep a small (one hopes) list at that index

probing: explore, in a systematic way, until the next open index

either tactic has costs, so keep collisions to a minimum by keeping the list partly empty



Python dictionaries are implemented using hash tables and probing. The cost of collisions is kept small by enlarging the underlying table when necessary, and the cost of enlarging is amortized over many dictionary accesses.

The result is that access to a dictionary element is O(1), essentially the time it takes to access a list element.

One downside is that extra work is required to order the keys or values of a dictionary. What is their "natural" order?





know your code

```
...inside out, left to right

def f(n):
    return n + 1

def g(m):
    return f(m) + 1

print(f(g(f(1) * 2) + 2))
```



know more code

```
... bottom to top
class A:
    def g(self, n):
        return n + 1
    def f(self, m):
        return self.g(m)
class B(A):
    def g(self, n):
        return 2 * n
if __name__ == "__main__":
    b = B()
    print(b.f(2))
    a = A()
    print(a.f(2))
```

...and more code

```
...think locally...
x = 7

def f():
    y = x
    print(y)
    if False:
        x = 2

if __name__ == "__main__":
    f()
```

aliases

```
>>> L = [[]] * 3
>>> L
[[], [], []]
>>> L[0].append(1)
>>> L
```

persistent values

default values are created when a function is defined...!

bare bones

▶ 3 hours

▶ 7 questions

comprehensive

▶ no aid sheet

UNIVERSITY OF TOR Faculty of Arts and Sc	April 2016 Examinat	CSC 148H1S Duration — 3 hou	
PLEAST	HA	M OIN	

Last Name: First Name:

Student Number:

Do not turn this page until you have received the signal to start. (In the meantime, please fill out the identification section above, and read the instructions below.)

#1:	#2 / 6	#3	#4:	# 55	# 68 / 77 #	TOTAL: /54	
This exam consists of 7 questions on 15 pages (including this one).	When you receive the signal to start, please make sure that your copy of the exam is complete.	Please arewer questions in the space provided. You will carn 20% for	any question you kave blank or write "I camot answer this question," on. You may earn substantial part marks for writing down the outline	of a solution and indicating which steps are missing. You must achieve 40% of the marks on this final ecam, to pass this	course. Write your student number at the bottom of pages $2 \cdot 15$ of this exam.	There is a Python API at the end of this exam that you may tear off for reference.	

Good Luck!

Total Pages = 15

Page 1

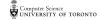
CONTD





topics

- object-oriented programming and design: lecture slides and example code, in-class exercises, weeks 1-3, lab #1 and lab #2, and assignment #1
- abstract data types, stacks, queues: lecture slides and example code, weeks 3 and 4, lab #3
- linked lists: lecture slides and example code, in-class exercise, weeks 4 and 5, and lab #4
- reading, writing recursion on nested lists: lecture slides and example code, in-class exercise, week 6, lab #5 and lab #6





more topics

- recursion on general trees: lecture slides and example code week 7, in-class exercises on contains and leaf, lab #7
- recursion on binary trees: lecture slides and example code week 8, lab #8, and assignment #2
- binary search trees, insertion, deletion, mutation: lecture slides and example code week #9, in-class exercise
- efficiency, recursion, recursive structures: lecture slides and example code week #10, lab #9
- big-Oh, hash table: lecture slides and example code week #11 hash table, tracing and traps: lecture slides and example code week #12
- office hours: Mondays 3-5 p.m., April 11, 18, 25, BA4270.

