CISS450: Artificially Intelligence
Lecture 17: Linked Lists

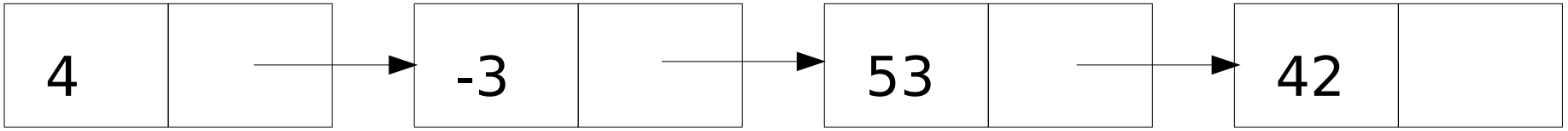
Yihsiang Liow

Agenda

- Study linked lists. This is our first lecture on data structures.
- Assume you have CISS350.

What are Linked Lists?

- Here's the big picture: A linked list is a list of thing and they are related by links. The links follow a line. Right now, do ***not*** think of programming.
- Here is an example of a linked list of integers:



- Each entity contains an integer and a link will be called a node.

What is a Linked List?

- You can think of the above linked list as a C/C++ array or Python list in the sense that you can keep things in the list. In this case, integers. The only difference is the presence of links. Think of links as letting you travel from one thing to another.
- For C/C++ arrays or Python list, there are no links. In fact the index of the C/C++ array or Python list will let you move around the cells. For a linked list, the only way to move around is to follow the links. This means that you have to have a node to begin with.

Realization

- Here's one possible implementation in Python:
- We use a dictionary to model each node. Each dictionary has keys “data” and “next”. The value of “data” is the integer for that node and the “next” value is the next dictionary. (You can also use classes and objects.)
- The following code models the above:

```
d = {'data':42, 'next':None}
c = {'data':53, 'next':d}
b = {'data':-3, 'next':c}
a = {'data':4,  'next':b}
```

Traversing

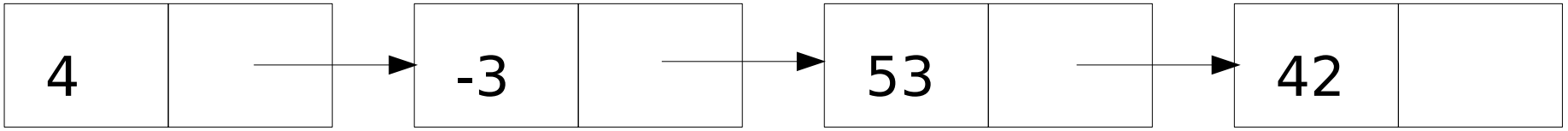
- Recall again that you must have a node to begin with. So let's say we have the node dictionary a. (Forget the names of the rest). Write a short Python snippet of code to print all the integers in the list:

Traversing

- But you say ... ok, but I can store all these integers in a Python list (or for C/C++ an array) and print everything out easily with a for-loop. What's the point??

Add

- A linked list being a container of things should allow one to add things.
- How would you add 100 to the linked list so that 100 is between 53 and 42? (Don't worry about the code yet)



Add

- Write some Python code to search for the node with 53 and then add a node with integer 100 after it:

Delete

- Also, you should be able to delete things in the linked list. How would you do that at a high level (use a picture)? How would you do it in Python?

Why?

- It's pretty obvious that, just like traversing the whole linked list, adding and deleting is just as painful. Why not use Python's list (or arrays for C/C++)?
- Any ideas?

Singly Linked List class

Exercise. Complete the obvious singly linked list class. No sentinel.

```
class SLNode:
    def __init__(self, key, next_ = None):
      self.key__ = key
      self.next__ = next_
    def get_key(self):
        return self.key__
    def get_next(self):
        return self.next__
```

Singly Linked List class

```
class SLList:
    def __init__(self):
        self.head__ = None
        self.len__ = 0
    def get_head(self):
        pass
    def insert_head(self, key):
        pass
    def delete_head(self):
        pass
    def __str__(self):
        pass
```

- After you are done:
  - Add properties so that you can do list.head instead of list.get_head().
  - Allow len(list).
  - Implement a version that uses sentinel node

Doubly Linked Lists

- The above linked list is more specifically a ***singly*** linked list – it's a linked list where each node has a ***single link***. The link allows us to travel from one end to the other.
- At a high level, how would you modify the structure so that you can travel backwards? This is the ***doubly*** linked list.

Singly and Doubly Linked Lists

- How would you implement a doubly linked list of integers in Python? How would you implement add, delete, etc?