Algorithms and Data Structures for Data Science

Lists Implementation Day 2

CS 277
Brad Solomon

February 7, 2024

Department of Computer Science
Exam 0: 2/13 — 2/15

Two coding questions in 50 minutes

Practice exam has four total questions

Content is Python fundamentals and lists

Pretend it's a real exam.

Review the content rather than specifics.
Office Hours Time Change

Not available Thursday @ 11 AM

For this week only: Friday @ 10 - 11 AM
for qlen in [5, 10, 15, 20, 25]:
    print("For query length: {}".format(qlen))
    time_list = []
    for pow in [15, 16, 17, 18, 19, 20]:
        rowcount = 2**pow
        query = random.sample(string.ascii_uppercase, qlen)
        random.shuffle(query)
        start = timeit.default_timer()
        checkPathExists("data/graph_r{}_c8.txt".format(rowcount), query)
        end = timeit.default_timer()
        time = end-start
        time_list.append(time)
    print("Runtime with increasing edge count:")
    print(time_list)
Learning Objectives

Build a linked list implementation in Python

Discuss list implementation strategies in the context of Big O

Extend lists to multi-dimensions using built-in and NumPy
def convert_1D_to_2D(inList, rowSize):
    listLen = len(inList)
    numRows = math.ceil(listLen/rowSize)
    outList = []
    count = 0
    ops = 0
    for i in range(numRows):
        tempList = []
        for j in range(rowSize):
            if count >= listLen:
                tempList.append(-1)
            else:
                tempList.append(inList[count])
                ops+=1
            count+=1
        outList.append(tempList)
    print(ops)
    return outList
(Theoretical) List Implementations

1. Array List
   - Evenly spaced (fixed size)
   - Allocate chunk of memory

2. Linked List
### Array Implementation

<table>
<thead>
<tr>
<th></th>
<th>Array</th>
<th>Singly Linked List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look up given an input <strong>position</strong></td>
<td></td>
<td>(O(1))</td>
</tr>
<tr>
<td>Search given an input <strong>value</strong></td>
<td></td>
<td>(O(n))</td>
</tr>
<tr>
<td>Insert/Remove at <strong>front</strong></td>
<td></td>
<td>(O(n))</td>
</tr>
<tr>
<td>Insert/Remove at <strong>arbitrary</strong> location</td>
<td></td>
<td>(O(n))</td>
</tr>
</tbody>
</table>
class Node:
    def __init__(self, data, next=None):
        self.data = data
        self.next = next
class Node:
    def __init__(self, data, next=None):
        self.data = data
        self.next = next

n1 = Node(3)
n2 = Node(5)
n3 = Node(7)
n1.next = n2
n2.next = n3
curr = n1
print(curr.next.next.data)
class Node:
    def __init__(self, data, next=None):
        self.data = data
        self.next = next

class linkedList:
    def __init__(self, head=None):
        self.head = head

    def __str__(self):
    def __len__(self):
    def __getitem__(self):
    def add(self):
    def insert(self):
    def delete(self):
    def remove(self):
Linked List: add()}

```python
ll = LinkedList()
for i in range(3):
    ll.add(i)
```

1) Make a new Node object

2) Add node to LL by:
2.5) Setting new Node's next equal to head

Set head to new node
Linked List: add()

```
ll = linkedList()
for i in range(3):
    ll.add(i)
```

```
class linkedList:
    def add(self, data):
        # Implementation of add method
```

Add 0:
- Head: None

Add 1:
- Created new Node (data, head)
- Set head to new Node

Add 2:
- Created new Node (data, head)
- Set head to new Node

Add 3:
- Created new Node (data, head)
- Set head to new Node

Add 4:
- Created new Node (data, head)
- Set head to new Node

Add 5:
- Created new Node (data, head)
- Set head to new Node
Linked List: add()

```python
ll = linkedList()
for i in range(3):
    ll.add(i)
```

```python
class linkedList:
    def add(self, data):
        temp = self.head
        self.head = Node(data, temp)
```

Line 3:
keep track of old head node

Line 4:
create new Node object w/ data & next pointing to old head

& we assign to head
Linked List: `__len__()`

```python
ll = linkedList()
for i in range(3):
    ll.add(i)
print(len(ll))
```

How to walk down chain?

Create a new var "current"

```
len = 0
curr = head
while (curr.next != None):
    len += 1
    curr = curr.next
```

```
head
2 1 0
```

When `curr.next` None?

`curr = curr.next`
Linked List: \texttt{\_\_len\_\_( )}

```
ll = linkedList()
for i in range(3):
    ll.add(i)
print(len(ll))
```

```
class linkedList:
    def \_\_len\_\_(self):
        i = 0
        curr = self.head
        while(curr):
            curr = curr.next
            i+=1
        return i
```
In-Class Exercise: Linked List `__getitem__()`

```python
def LinkedList():
    pass

ll = LinkedList()
for i in range(5):
    ll.add(i)
print(ll[3])
```

```
head

0 -> 1 -> 2 -> 3 -> 4 -> None
```
In-Class Exercise: Linked List \_\_getitem\_\_( )

```python
ll = linkedList()
for i in range(5):
    ll.add(i)
print(ll[3])
```
**Linked List: `__str__()`**

```python
class linkedList:
    def __str__(self):
        curr = self.head
        out="[
            while(curr):
                out+="{},".format(curr.data)
                curr = curr.next
        
        if out[-1]==",":
            out = out[:-1]
        out = out +"]\n"
        return out
```

```python
ll = linkedList()
for i in range(3):
    ll.add(i)
print(ll)
```

```
[0, 1, 2]
```

```
head
```

```
0 1 2 None
```

On your own: Linked List find()

```
#initialize ll
node = ll.find("B")
if(node):
    print("Exists!")
else:
    print("Doesn't exist!")
```
Linked List: insert( )

```python
for i in range(5):
    ll.add(i)
ll.insert("Value", 2)
print(ll)
```

0(1)

1) Make new Node
2) Set new Node next = current node at position i
3) Change previous node's next to be new node

0(n)

head

1) O(1)
2) O(1) + O(n)
3) O(1) + O(n)

None

find node curr O(n)
In-Class Exercise: Linked List insert()

```python
for i in range(5):
    ll.add(i)
ll.insert("Value", 2)
print(ll)
```

```
[0, 1, 2, 3, 4, Value]
```

```
head -> 0 -> 1 -> 2 -> 3 -> 4 -> None
```
Linked List: delete()

```python
for i in range(5):
    ll.add(i)
ll.delete(1)
print(ll)
```

```
head

0 -> 1 -> 2 -> 3 -> 4 -> None
```
Linked List: delete()

```python
for i in range(5):
    ll.add(i)

ll.delete(1)
print(ll)
```

![Diagram of a linked list with elements 0 to 4 and an arrow indicating deletion of the second element, resulting in a list from 0 to 3.](image-url)
Linked List: remove()
Optional Exercise: remove()

```
ll = linkedList()
for i in range(5):
    ll.add(i)
ll.add(i)
ll.remove(4)
```
<table>
<thead>
<tr>
<th></th>
<th>Singly Linked List</th>
<th>Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look up given an input <strong>position</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search given an input <strong>value</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert/Remove at <strong>front</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert/Remove at <strong>arbitrary</strong> location</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Programming Toolbox: Multidimensional Lists

How can we make a matrix in Python?
Programming Toolbox: Multidimensional Lists

How is a matrix in Python indexed?
```python
 outerList = []
 for i in range(5):
     innerList = []
     for j in range(5):
         innerList.append(i+j)
     outerList.append(innerList)
 print(outerList)
 print(outerList[3][1])
```
Python 2D lists

```
outerList = []
for i in range(5):
    innerList = []
    for j in range(5):
        innerList.append(i+j)
    outerList.append(innerList)
print(outerList)
print(outerList[3][1])
```
NumPy is optimized for multidimensional arrays of numbers

```python
import numpy as np

# Convert list to np list
nl = np.array([1, 2, 3, 4, 5, 6])
print(nl)

print(nl.shape)

# Modify list shape
nl2 = nl.reshape(3, 2)
print(nl)
print(nl2)

# Create a new list
nl3 = np.arange(15).reshape(5, 3)
nl4 = np.zeros((2, 5))

print(nl3)
print(nl4)
```
Programming Toolbox: NumPy

Basic operations are applied **elementwise** (to each item of a list)

```
nl = np.arange(4).reshape(2, 2)
print(nl)
nl2 = nl * 2
print(nl2)
# Matrix multiplication
# 0*0+1*4     0*0+1*6
# 2*0+3*4     2*2+3*6
print(nl.dot(nl2))
```

Explore on your own: [https://numpy.org/devdocs/](https://numpy.org/devdocs/)