Lecture 13, 26 September 2024

Linked lists

In [1]:
```
class List:
    def __init__(self):
        self.value = None
        self.next = None
        return

    def isempty(self):
        return(self.value == None)

    def append(self,v):   # append, iterative
        if self.isempty():
            self.value = v
            return

        temp = self
        while temp.next != None:
            temp = temp.next

        temp.next = List()
        temp.next.value = v
        return

    def insert(self,v):
        if self.isempty():
            self.value = v
            return

        newnode = List()
        newnode.value = v

        # Exchange values in self and newnode
        (self.value, newnode.value) = (newnode.value, self.value)

        # Switch links
        (self.next, newnode.next) = (newnode, self.next)

        return


    def __str__(self):
        # Iteratively create a Python list from linked list
        # and convert that to a string
        selflist = []
        if self.isempty():
            return(str(selflist))

        temp = self
        selflist.append(temp.value)

        while temp.next != None:
          temp = temp.next
          selflist.append(temp.value)

        return(str(selflist))
```

In [2]:
```
l = List()
l.append(5)
print(l)
```

```
[5]
```

In [3]:
```
l.append(7)
print(l)
```

```
[5, 7]
```

In [4]:
```
l.append(9)
print(l)
```

```
[5, 7, 9]
```

In [5]:
```
l.insert(4)
print(l)
```

```
[4, 5, 7, 9]
```

Change the constructor

In [6]:
```
class List:
    def __init__(self,initlist = []):
        self.value = None
        self.next = None
        for x in initlist:
            self.append(x)
        return

    def isempty(self):
        return(self.value == None)

    def append(self,v):   # append, iterative
        if self.isempty():
            self.value = v
            return

        temp = self
        while temp.next != None:
            temp = temp.next

        temp.next = List()
        temp.next.value = v
        return

    def insert(self,v):
        if self.isempty():
            self.value = v
            return

        newnode = List()
        newnode.value = v

        # Exchange values in self and newnode
        (self.value, newnode.value) = (newnode.value, self.value)

        # Switch links
        (self.next, newnode.next) = (newnode, self.next)

        return


    def __str__(self):
        # Iteratively create a Python list from linked list
        # and convert that to a string
        selflist = []
        if self.isempty():
            return(str(selflist))

        temp = self
        selflist.append(temp.value)

        while temp.next != None:
          temp = temp.next
          selflist.append(temp.value)

        return(str(selflist))
```

Some performance measurement

In [7]:
```
l = List([11,12,13])
print(l)
```

```
[11, 12, 13]
```

In [8]:
```
l.append(14)
print(l)
```

```
[11, 12, 13, 14]
```

In [9]:
```
l.insert(10)
print(l)
```

```
[10, 11, 12, 13, 14]
```

In [10]:
```
import time
```

Insert items at the start of a linked list, multiples of 10^5, linear blowup

In [11]:
```
for i in range(1,5):
    l1 = List()
    start = time.perf_counter()
    for j in range(i*100000):
        l1.insert(j)
```

```
    elapsed = time.perf_counter() - start
    print(i*100000,elapsed)
```

```
100000 0.141151190000528
200000 0.1800829439998779
300000 0.2989669700000377
400000 0.48523681000006036
```

Insert items at the start of a Python list, multiples of 5×10^4, quadratic blowup

In [12]:
```
for i in range(1,5):
    l2 = []
    start = time.perf_counter()
    for j in range(i*50000):
        l2.insert(0,j)
    elapsed = time.perf_counter() - start
    print(i*50000,elapsed)
```

```
50000 0.3145574010013661
100000 1.3161295789977885
150000 3.008215244000894
200000 7.537719673997344
```

Append items at the end of a linked list, multiples of 10^4, quadratic blowup

In [13]:
```
for i in range(1,5):
    l1 = List()
    start = time.perf_counter()
    for j in range(i*10000):
        l1.append(j)
    elapsed = time.perf_counter() - start
    print(i*100000,elapsed)
```

```
100000 2.9528205919996253
200000 10.977468514996872
300000 26.732917231998726
400000 46.65877005100265
```

Append items at the start of a Python list, multiples of 10^6, linear blowup

In [14]:
```
for i in range(1,5):
    l2 = []
    start = time.perf_counter()
    for j in range(i*1000000):
        l2.append(j)
    elapsed = time.perf_counter() - start
    print(i*50000,elapsed)
```

```
50000 0.09546786200007773
100000 0.21856623599887826
150000 0.2940401890009525
200000 0.37559778400100186
```