NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 3, Lecture 8

Madhavan Mukund, Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Inductive definitions

Many arithmetic functions are naturally defined inductively

- Factorial
  - $0! = 1$
  - $n! = n \times (n-1)!$
- Multiplication — repeated addition
  - $m \times 1 = m$
  - $m \times n = m + (m \times (n-1))$

Inductive definitions …

- Define one or more base cases
- Inductive step defines f(n) in terms of smaller arguments

Recursive computation

- Inductive definitions naturally give rise to recursive programs

```
def factorial(n):
  if n == 0:
    return(1)
  else:
    return(n * factorial(n-1))
```

Recursive computation

- Inductive definitions naturally give rise to recursive programs

```
def multiply(m,n):
  if n == 1:
    return(m)
  else:
    return(m + multiply(m,n-1))
```

Inductive definitions for lists

- Lists can be decomposed as
  - First (or last) element
  - Remaining list with one less element
- Define list functions inductively
  - Base case: empty list or list of size 1
  - Inductive step: f(l) in terms of smaller sublists of l

Inductive definitions for lists

- Length of a list

```
def length(l):
  if l == []:
    return(0)
  else:
    return(1 + length(l[1:])
```

Inductive definitions for lists

- Sum of a list of numbers

```
def sumlist(l):
  if l == []:
    return(0)
  else:
    return(l[0] + sumlist(l[1:])
```

Recursive insertion sort

- Base case: if list has length 1 or 0, return the list
- Inductive step:
  - Inductively sort slice `l[0:len(l)-1]`
  - Insert `l[len(l)-1]` into this sorted slice

Recursive insertion sort

```
def InsertionSort(seq):
  isort(seq,len(seq))

def isort(seq,k): # Sort slice seq[0:k]
  if k > 1:
    isort(seq,k-1)
    insert(seq,k-1)

def insert(seq,k): # Insert seq[k] into sorted seq[0:k-1]
  pos = k
  while pos > 0 and seq[pos] < seq[pos-1]:
    (seq[pos],seq[pos-1]) = (seq[pos-1],seq[pos])
    pos = pos-1
```

Recursion limit in Python

- Python sets a recursion limit of about 1000

```
>>> l = list(range(1000,0,-1))
>>> InsertionSort(l)
. . .
RecursionError: maximum recursion depth
exceeded in comparison
```

- Can manually raise the limit

```
>>> import sys
>>> sys.setrecursionlimit(10000)
```

Recursive insertion sort

- $T(n)$, time to run insertion sort on length n
  - Time $T(n-1)$ to sort slice `seq[0:n-1]`
  - $n-1$ steps to insert `seq[n-1]` in sorted slice
- Recurrence
  - $T(n) = n-1 + T(n-1)$
    $T(1) = 1$
  - $T(n) = n-1 + T(n-1) = n-1 + ((n-2) + T(n-2)) = \ldots =$
    $(n-1) + (n-2) + \ldots + 1 = n(n-1)/2 = O(n^2)$

$O(n^2)$ sorting algorithms

- Selection sort and insertion sort are both $O(n^2)$
- $O(n^2)$ sorting is infeasible for n over 5000
- Among $O(n^2)$ sorts, insertion sort is usually better than selection sort
  - What happens when we apply insertion sort to an already sorted list?
- Next week, some more efficient sorting algorithms