Algorithms and complexity

using python.

Iván Andrés Trujilllo Abella

Facultad de Ingenieria
Pontificia Universidad Javeriana

trujilloiv@javeriana.edu.co

Algorithms and complexity

The time of execution of a program could be important when large input data.

Sorting algorithms

The problem of sort an unsorted **list** is a recurrent problem to study the complexity and the efficient in the way how we can solve a problem.

Bubble sort algorithm

this strategy consist in compare the adjacent values and replace the adjacent only if is grater than the previous one.

```
lista = [a,b,c,d,e]
```

Assume that $a < b < c < d < e$
then the first iteration we have

```
[b,c,d,e,a]
```

translate the first element for all list until get the last position, in the worst case.
the following list must be iterate over n-1.

Bubble sort algorithm

Python

```
data = [99,98,97,96,95]
iter = len(data)
while iter >0: # or for i in range(0,iter-1):
  for i in range(0,iter-1):
    if data[i]> data[i+1]:
      data[i+1], data[i] = data[i], data[i+1]
  iter = iter -1
print(data)
```

Note that there are only needed $n-1$ iterations due to $n-1$ iteration the first element is the lesser value of the n values.

Preliminar 1

Remember that we can concatenate two list

```
listaOne=[1,2,3]
listaTwo=[4,5,6]
listaThree = listaOne+ListaTwo
print(listaThree)
```

will return:

```
[1,2,3,4,5,6]
```

Divide and conquer

This is a strategy to resolve different kind of problems, among them the reduction of iterations.

Quick Sort

We can think in the following method:

```
lista = [a,b,c,d,e,f]
```

lista contain random numbers, take the first of the last and called pivot, and try to sorted to the

Insights

for code implementation

The logic behind this is two positional index, beginning in 0-index, one index to keep position and the other to iterate over the data, the idea is keep the values greater than pivot with position index, and replace by first lesser value in the index iterator, after make the first swap then we need increment the positional index for the next more greater value than pivot. for this reason we can interchange the last positional index with the pivot value, due it is the first value greater than pivot.

Recursive instance

Note that we can broke up the list.

Call stack

what is call stack in recursion? what is recursive leap of faith

```
def loop(k):
  if k ==10:
    print('dont exceed')
  else:
    print(f"loop:{k}")
    loop(k+1)
    print(k)
loop(0)
```

Recursion

$$a * b = b + b + b + ... + b(a - times) \tag{1}$$

```
def product(a,b):
  if a==1:
    return b
  else:
    return b + product(a-1,b)
```

Recursion

$$\frac{n(n+1)}{2} = \sum_{i=1}^{n} i \tag{2}$$

in a recursion way?

```
def sum(k):
    if k==0:
        return 0
    else:
    return k + sum(k-1)
```

Fibonnaci serie

$$f(9) = 1, 1, 2, 3, 5, 8, 13, 21, 34 \tag{3}$$

```
def fibonacci(x):
  i = 0
  a,b = 0,1
  while i < x:
    a,b = b,a+b
    i +=1
    print(a)
```

In a recursively way:

```
def fibor(k):
  if k<=2:
    return 1
  else:
    return fibor(k-1) + fibor(k-2)
```

Power function

$$f(n) = a^n = a * a * a * a ... * a(n - times)$$
$$f(n) = a * a^{n-1} \tag{4}$$

```
def powerr(a,n):
  if n==0:
    return 1
  else:
    return a*powerr(a,n-1)
```

Recursion ideas

Recursion define a base case, and a recursive case for instance in factorial the base case is $factorial(0) = 1$ and for $x > 0$ $n * (n-1)! = factorial(n)$ that is recursive case.

stack

is a data structure, store information in a FILO (first in last outUntitled3) way A list is a stack if only uses the methods append and pop.

Kind of recursion

tail recursion:

Hanoi

Rules: you can only move a disk at time and only a smaller disk could be top of a bigger disk.
Suppose do you have three pegs and N disk. 'A" B" C'
the process could be

```
To carry out the 1 (biggest) disk to last  tower (C),
you must carry out the N-1 disk's to the next tower(B),
to carry out the N-1 disk to tower(B),
you must carry out the N-2 disk's to tower(C),
to carry out N-3 disk to C,
carry out N-3 disk to B,
and so on, repeat
to carry out to reach the smallest disk.
```

Hanoi

We said a stack of n disk as Sn.

```
To carry out Sn to C,
you must carry out Sn-1 to B,
to carry out Sn-1 to B,
you must carry out Sn-2 to C,
to carry out Sn-2 to C,
you must carry out Sn-3 to B... and so on to reach S1
```

Find the maximun value in a list

∗args

```
def max(*args):
    aux = args[0]
    for n in args[1:]:
        if n > aux:
            aux = n
    return res
```

Local variable, global variable, and recursion


IHS
Pontificia Universidad
JAVERIANA
Bogotá

Divide and conquer

Insertion sort

Merge sort

Time complexity and sorting

The worst case

- Selection sort $O(n^2)$
- Insertion sort $O(n^2)$
- Merge sort $O(nlog(n))$
- Quick sort $O(n^2)$

Turtle introduction in Colab

Not need be install it,

```
!pip3 install ColabTurtle
from ColabTurtle.Turtle import *
initializeTurtle()
backward(px) # pixel to back
forward(px) # pixel to forward
left(angles)
right(angles)
```

Turtle

Draw a rectangle

```
initializeTurtle()
right(90)
forward(100)
left(90)
forward(100)
left(90)
forward(100)
left(90)
forward(100)
```

Turtle

Not line

```
initializeTurtle()
right(90)
penup()
forward(100)
left(90)
forward(100)
pen(down)
left(90)
forward(100)
left(90)
forward(100)
```

You can uses another parameters as **speed([1-13])** to change the speed of drawing.

Loops in turtle

```
for x in range(4):
  right(90)
  forward(100)
```

there are another methods in turtle

- color
- pensize
- width

set_pos(x, y) image the screen in halves and representing the x-y coordenate system.