Data Structures Discipline with Python

@fmasanori
I love teaching
CS Professor at FATEC
https://about.me/fmasanori
http://pycursos.com/python-para-zumbis/
DATA STRUCTURES FAIL YOU WILL
FINISHED DATA STRUCTURES ASSIGNMENT
Algoritmos

Eng. de Software

Estrutura de dados

MOTHER OF GOD
**Difficult with C language**

<table>
<thead>
<tr>
<th>Programming then and now</th>
<th>Reading specification</th>
<th>Starting dev tools</th>
<th>Coding...</th>
<th>Coding...</th>
<th>Debugging</th>
<th>Finished.</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

"A C program is like a fast dance on a newly waxed dance floor by people carrying razors."

*Waldi Ravens. Programmer.*
But I love Data Structures
Data Structures are cool

O \( N \log N \)
Data Structures with Python at FATEC
Retention 2008

85% retained
15% approved
Summary

Data Structures with C:
- 85% retained (2008)

Python > C:
- 12% retained (2014)
- 10% retained (2015)
- ENADE grade 5/5 (max) (2013)
- 1st Programming Contest InterFATECs (1st/62) (2014)
Details

• Lab Only – 4 classes/week
• 4 Lab Projects (Python)
• Big Brother (some of the best students could help the other students as coaches)
• Algorithms in C (few) and Python (mainly)
Why Python?

Number of top 39 U.S. computer science departments that use each language to teach introductory courses

Analysis done by Philip Guo (www.pgbovine.net) in July 2014, last updated 2014-07-29

http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/fulltext
Usability is a problem for DS also...

"Results show that many aspects of traditional C-style syntax, while it has influenced a generation of programmers, exhibits problems in terms of usability for novices".

"Perl and Java did not accuracy rates significantly higher than a language with randomly generated keywords"

The most common fault in computer classes is to emphasize the rules of specific programming languages, instead of to emphasize the algorithms that are being expressed in those languages. D. Knuth interview at People of ACM, June, 2014.
Talk is cheap.
Show me the code.

Linus Torvalds
Variables are just names (references)

```python
>>> a = 42
>>> id(a)
1518584480
>>> id(42)
1518584480
>>> a = 'Python'
>>> id(a)
15542496
```
References == “pointers”

```python
>>> a = [1, 2, 3]
>>> b = a
>>> id(a)
48767184
>>> id(b)
48767184
>>> a[0] = 42
>>> a
[42, 2, 3]
>>> b
[42, 2, 3]
```

```python
>>> a = [4, 5, 6]
>>> b = list(a)
>>> id(a)
48772240
>>> id(b)
49151472
>>> a[0] = 42
>>> a
[42, 5, 6]
>>> b
[4, 5, 6]
```
Big integers

```
>>> 2 ** 1024
1797693134862315907729305190789024733617976978942
3065727343008115773267580550096313270847732240753
6021120113879871393357658789768814416622492847430
6394741243777678934248654852763022196012460941194
5308295208500576883815068234246288147391311054082
7237163350510684586298239947245938479716304835356
329624224137216
```
>>> 1 / 2
0.5

#include <stdio.h>

int main(void) {
    printf ("%f\n", 1 / 2);
    system ("pause");
}

Pressione qualquer tecla para continuar. . .
```python
>>> a = 42
>>> b = 'avocado'
>>> a, b = b, a
>>> a
'avocado'
>>> b
42

>>> name, share, price, (year, month, day) = ['ACME', 50, 91.1, (2015, 12, 21)]
>>> first, *middle, last = [-1, 1, 2, 3, 4, 5, -1]
>>> name, email, *fones = ('masanori', 'fmasanori@gmail.com', '3923-3858', '8113-5934', '3905-4851')
```
"The programming activity should be viewed as a process of creating works of literature, written to be read. “

--D.E. Knuth
for (i = 0; i < 10; i++);
printf("Ten times Hello World!");

if (x < y)
  if (pred(x))
    printf("One");
else if (x == y)
  printf("Two");
else
  printf("Tree");
SSL public key verification added and removed here

but mostly removed 😊
"To understand recursion, one must first understand recursion."

--folklore

"To solve the problem, I found barriers within barriers. So, I adopted a recursive solution."

--a student

def fib(n):
    print ('fib(%d)' % n)
    if n <= 2:
        return 1
    #bad O(2**n)
    return fib(n-1) + fib(n-2)

print (fib(5))
Recursion

```python
fibcache = {}
def fib(n):
    if n <= 2:
        return 1
    if n in fibcache:
        return fibcache[n]
    fibcache[n] = fib(n-1) + fib(n-2)
    return fibcache[n]

print (fib(100))
```

```
>>> 354224848179261915075
```
Recursion

```python
from functools import lru_cache

@lru_cache(maxsize=None)
def fib(n):
    if n <= 2:
        return 1
    else:
        return fib(n-1) + fib(n-2)

print(fib(100))

>>> 354224848179261915075
```
def dec2bin(n):
    if n == 0:
        return ''

    return dec2bin(n//2) + str(n%2)

print (dec2bin(18))

>>> 10010
```c
#include <stdio.h>
#include <stdlib.h>

struct cel {
    int cargo;
    struct cel *next;
};

typedef struct cel celula;

void Print (celula *lst) {
    celula *p;
    for (p = lst->next; p != NULL; p = p->next)
        printf ("%d\n", p->cargo);
}
```
void Insert (int y, celula *p) {
    celula *nova;
    nova = malloc (sizeof (celula));
    nova->cargo = y;
    nova->next = p->next;
    p->next = nova;
}

int main (void) {
    celula head;
    celula *lst;
    lst = &head;
    head.next = NULL;
    Insert (3, lst);
    Insert (2, lst);
    Insert (1, lst);
    Print (lst);
    system ("pause");
}
class Node:
    def __init__(self, cargo=None, next=None):
        self.cargo = cargo
        self.next = next

    def __str__(self):
        return str(self.cargo)

def print_list(node):
    while node is not None:
        print(node, end=" ")
        node = node.next
    print()

node1 = Node(1)
node2 = Node(2)
node3 = Node(3)
node1.next = node2
node2.next = node3
print_list(node1)

>>> 1 2 3
FIFOs: Distance in Networks

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

```
0 1 2 3 4 5
---
d 2 3 1 0 1 6
```
FIFOs: Distance in Networks

```
#define TAM 6

int A[TAM][TAM] = {{0, 1, 0, 0, 0, 0},
                     {0, 0, 1, 0, 0, 0},
                     {0, 0, 0, 0, 1, 0},
                     {0, 0, 1, 0, 1, 0},
                     {1, 0, 0, 0, 0, 0},
                     {0, 1, 0, 0, 0, 0}};

int *Distancias (int n, int o) {
    int *d, x, y;
    int *f, s, t;

    d = malloc (n * sizeof (int));
    for (x = 0; x < n; x++) d[x] = -1;
    d[o] = 0;
    f = malloc (n * sizeof (int));
    s = 0; t = 1; f[s] = o;

    while (s < t) {
        x = f[s++];
        for (y = 0; y < n; y++)
            if (A[x][y] == 1 && d[y] == -1) {
                d[y] = d[x] + 1;
                f[t++] = y;
            }
    }

    free (f);
    return d;
}
```
A = 
[[0, 1, 0, 0, 0, 0],
 [0, 0, 1, 0, 0, 0],
 [0, 0, 0, 0, 1, 0],
 [0, 0, 1, 0, 1, 0],
 [1, 0, 0, 0, 0, 0],
 [0, 1, 0, 0, 0, 0]]

def Distancias(n, origem):
    d = [-1] * n
    d[origem] = 0
    f = []
    f.append(0)
    f.append(0)
    while len(f) > 0:
        x = f[0]
        del f[0]
        for y in range(n):
            if A[x][y] == 1 and d[y] == -1:
                d[y] = d[x] + 1
                f.append(y)
    return d
Stacks: well-formed expression

```c
int BemFormada (char s[]) {
    char *p; int t;
    int n, i;
    n = strlen (s);
    p = malloc (n * sizeof (char));
    t = 0;
    for (i = 0; s[i] != '\0'; i++) {
        /* p[0..t-1] é uma pilha */
        switch (s[i]) {
            case '): if (t != 0 && p[t-1] == '(') --t;
                      else return 0;
                      break;
            case '}' : if (t != 0 && p[t-1] == '{') --t;
                      else return 0;
                      break;
            default: p[t++] = s[i];
        }
    }
    free (p);
    return t == 0;
}

int main (void) {
    printf ("%s\n", BemFormada ("((){()})") ? "Bem formada" : "Mal formada");
    printf ("%s\n", BemFormada ("({})") ? "Bem formada" : "Mal formada");
    system ("pause");
}```
```python
def BemFormada(s):
    p = []
    for c in s:
        if c == ')':
            if p[-1] == '(':  
                p.pop()
            else:
                return False
        elif c == '}':
            if p[-1] == '{':  
                p.pop()
            else:
                return False
        else:
            p.append(c)
    return True

print (BemFormada('((()){}())'))
print (BemFormada('(){}'))
```

Stacks: well-formed expression
Selection Sort

```c
void Selecao (int n, int v[]) {
    int i, j, k, min, x;
    for (i = i < n-1; i++) {
        min = i;
        for (j = i+1; j < n; j++)
            if (v[j] < v[min])
                min = j;
        x = v[i];
        v[i] = v[min];
        v[min] = x;
    }
}

int main(void){
    int i;
    int v[10]={7, 4, 3, 9, 0, 8, 5, 2, 6, 1};
    Selecao (10, v);
    for (i = 0; i < 10; i++)
        printf ("%d", v[i]);
    putchar(';');
}```
def selection(v):
    resp = []
    while v:
        m = min(v)
        resp.append(m)
        v.remove(m)
    return resp

import random
v = list(range(10))
random.shuffle(v)
v = selection(v)
print(v)
```c
1. int Divide (int p, int r, int v[]) {
2.     int c, j, k, t;
3.     c = v[r]; j = p;
4.     for (k = p; k < r; k++)
5.         if (v[k] <= c) {
6.             t = v[j], v[j] = v[k], v[k] = t;
7.             j++;
8.         }
9.     v[r] = v[j], v[j] = c;
10.    return j;
11. }

12. void Quicksort (int p, int r, int v[]) {
13.     int j;
14.     if (p < r) {
15.         j = Divide (p, r, v);
16.         Quicksort (p, j - 1, v);
17.         Quicksort (j + 1, r, v);
18.     }
19. }
```
def quicksort(v):
    if len(v) <= 1:
        return v

    pivot = v[0]
    equals = [x for x in v if x == pivot]
    smaller = [x for x in v if x < pivot]
    higher = [x for x in v if x > pivot]
    return quicksort(smaller) + equals + quicksort(higher)

print(quicksort([5, 7, 9, 3, 4, 0, 2, 1, 6, 8]))
with open('alice.txt') as arq:
    texto = arq.read()
    texto = texto.lower()
import string
for c in string.punctuation:
    texto = texto.replace(c, ' ')
    texto = texto.split()

dic = {}
for p in texto:
    if p not in dic:
        dic[p] = 1
    else:
        dic[p] += 1

Word Count, please download http://www.gutenberg.org/cache/epub/11/pg11.txt
## Projects: Sorting Algorithm comparison

<table>
<thead>
<tr>
<th></th>
<th>Mergesort</th>
<th>Quicksort</th>
<th>Selection</th>
<th>Native</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>0.05</td>
<td>0.00</td>
<td>0.40</td>
<td>0.00</td>
</tr>
<tr>
<td>4000</td>
<td>0.05</td>
<td>0.02</td>
<td>1.65</td>
<td>0.00</td>
</tr>
<tr>
<td>6000</td>
<td>0.05</td>
<td>0.03</td>
<td>3.64</td>
<td>0.00</td>
</tr>
<tr>
<td>8000</td>
<td>0.07</td>
<td>0.02</td>
<td>6.49</td>
<td>0.00</td>
</tr>
<tr>
<td>10000</td>
<td>0.10</td>
<td>0.03</td>
<td>10.16</td>
<td>0.01</td>
</tr>
<tr>
<td>12000</td>
<td>0.10</td>
<td>0.05</td>
<td>14.69</td>
<td>0.00</td>
</tr>
<tr>
<td>14000</td>
<td>0.13</td>
<td>0.06</td>
<td>19.90</td>
<td>0.02</td>
</tr>
<tr>
<td>16000</td>
<td>0.14</td>
<td>0.06</td>
<td>26.55</td>
<td>0.01</td>
</tr>
<tr>
<td>18000</td>
<td>0.16</td>
<td>0.08</td>
<td>32.94</td>
<td>0.02</td>
</tr>
<tr>
<td>20000</td>
<td>0.20</td>
<td>0.08</td>
<td>42.20</td>
<td>0.02</td>
</tr>
<tr>
<td>22000</td>
<td>0.23</td>
<td>0.08</td>
<td>50.67</td>
<td>0.02</td>
</tr>
</tbody>
</table>
Projects: Arthur Merlin Games simplification
Projects: Binaries regions

teste3 =

```
0011001010 0011002020
0110001010 0110002020
0011001110 0011002220
0000000000 0000000000
0010001010 0030004040
0010011111 0030044444
1111100000 3333300000
0010001110 0030005550
0010001110 0030005550
```
```
Projects: Minimum Degree Greedy Heuristic for MIS

- S={5}
- S={5, 1}
- S={5, 1, 2}
- S={5, 1, 2, 4}
Conclusions

- C is good for optimization (details, low level)
- Python is good to show the essence of the algorithms (readability, high level)
- If the algorithm is the same (complexity) “premature optimization is evil” also in teaching Data Structures.
Questions?
gist.github.com/fmasanori
http://about.me/fmasanori
fmasanori@gmail.com

Slides: bit.ly/python-DS