Tecnologia de la programació

Enginyeria de Sistemes TIC – *i*TIC

Sebastià Vila-Marta

Escola Politècnica Superior d'Enginyeria de Manresa
Universitat Politècnica de Catalunya

16 d'abril de 2012

Estructures de dades lineals

- En el tema anterior...
- Estructures de dades
- Estructures de dades lineals
- Piles
- Cues
- Per a la setmana vinent ...

En el tema anterior...

- Hem estudiat com es comporten els algoritmes respecte el temps de càlcul i quins són els factors que influeixen.
- Hem descobert que la component més rellevant en el temps de càlcul és l'estructura del propi algoritme.
- Hem Definit el concepte de cost en temps assimptòtic en cas pitjor com una forma de classificar el comportament dels algoritmes i poder-los comparar.

Estructures de dades

- Anomenem *estructura de dades* a una forma concreta d'organitzar i emmagatzemar dades en un computador de manera que l'accés sigui eficient.
- Les estructures de dades serveixen per representar les entitats del problema a resoldre.
- Les estructures de dades són un àmbit d'estudi fonamental de la programació:
- Llistes, tuples i diccionaris són estructures de dades.
- Les estructures de dades:
  - Es classifiquen d'acord amb la funcionalitat.
  - Tenen especificació (QUÈ) + implementació (COM).
  - La implementació determina l'eficiència en temps de les seves operacions.

Equació de Niklaus Wirth

$$\text{Programes} = \text{Algoritmes} + \text{Estructures de dades}$$

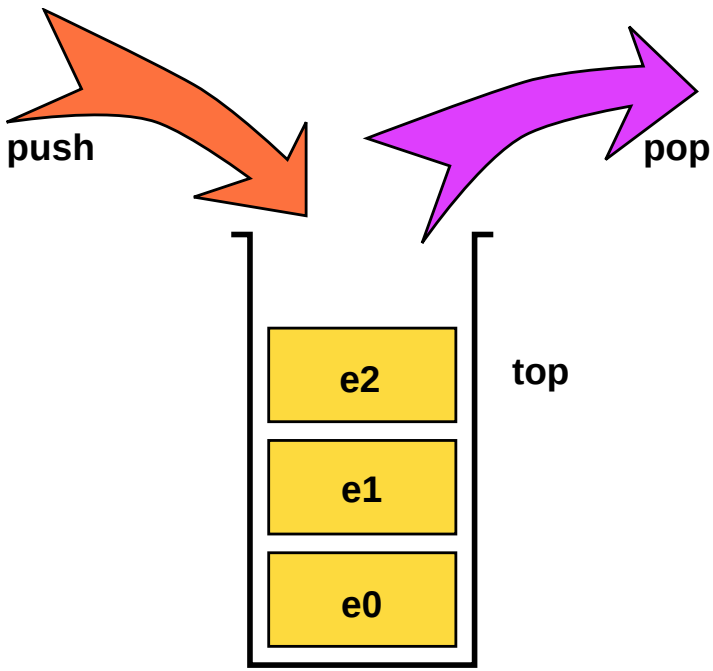
Estructures de dades lineals

- Les ED lineals són aquelles en què l'organització de les dades és seqüèncial, i.e.:
  - Hi ha un primer (esquerra) i un darrer (dreta) element.
  - El concepte de següent (anterior) està ben definit.
- Les ED lineals es classifiquen segons les formes d'accedir-hi en:
  - Piles (stacks).
  - Cues (queues), cues de doble entrada (dequeues) i cues amb prioritat (priority queues).
  - Llistes.

Piles

- Una pila és una estructura lineal en la que el darrer element emmagatzemat és sempre el primer en ser esborrat (LIFO). La seva estructura recorda una pila de plats:

- push(e) Empila un element damunt la pila.
- pop() Desempila l'element del cim de la pila.
- top() Retorna l'element de cim de la pìla.
- empty() Retorna cert si la pila és buida.



Una pila implementada sobre llistes

Implementació:

```
class Stack(object):
    def __init__(self):
        self._p = list()

    def push(self,e):
        self._p.append(e)

    def pop(self):
        del self._p[-1]

    def top(self):
        return self._p[-1]

    def __len__(self):
        # servira com a operacio empty()
        return len(self._p)
```

Exemple d'ús:

```
>>> s = Stack()
>>> s.push(23)
>>> s.push(12)
>>> s.top()
12
>>> s.top() + 10
22
>>> len(s)
2
>>> s.pop()
>>> len(s)
1
>>> s.top()
23
```

Exemple d'ús: avaluacio postfixa (RPN) I

- Les calculadores HP usen tradicionament una forma diferent d'expressar els càlculs anomenada RPN (*Reverse Polish Notation* també notació postfixa. Aquesta forma no requereix parèntesis i és notablement més àgil d'escriure que la notació convencional.
- Una expressió com aquesta $(2+3)*4$ es pot escriure de tres formes diferents segons on es posin els operadors:
  - Davant dels operands (prefixa): $* \ 4 \ + \ 2 \ 3$
  - En mig dels operands (infixa): $(2 \ + \ 3) \ * \ 4$
  - Darrera dels operands (postfixa): $2 \ 3 \ + \ 2 \ *$
- La forma postfixa és molt senzilla d'avaluar usant una pila. Simplement cal llegir d'esquerra a dreta l'expressió i:
  1. Si l'element és un número, l'empilem.
  2. Si l'element és una operació, desempilem els operands, operem i empilem el resultat.

  En acabar el cim de la pila té el resultat final.

Exemple d'ús: avaluacio postfixa (RPN) II

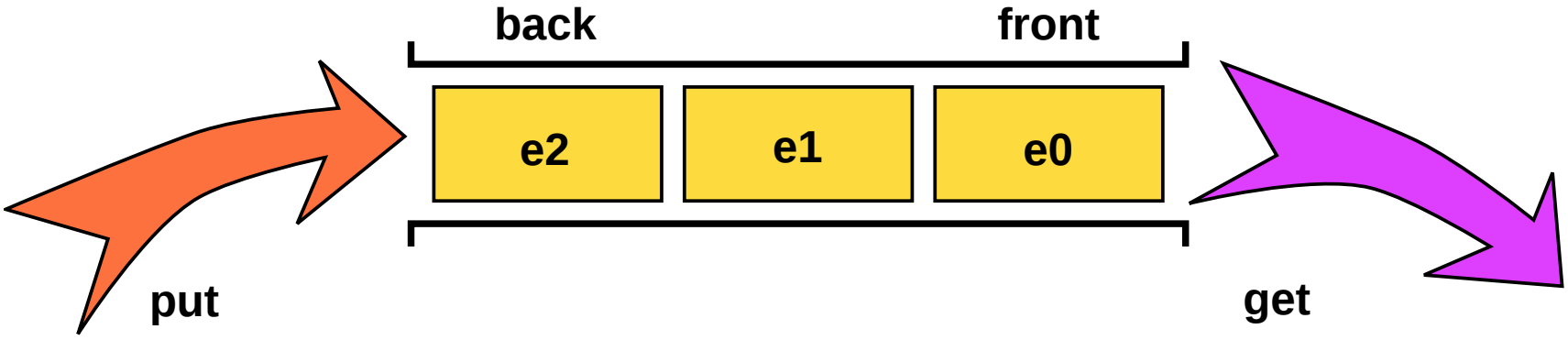
```
from stack import Stack

def pop_two(s):
    op1 = s.top()
    s.pop()
    op2 = s.top()
    s.pop()
    return (op1, op2)
```

```
s = Stack()
t = raw_input()
if t.isdigit():
    s.push(int(t))
else:
    if t == '+':
        op1, op2 = pop_two(s)
        s.push(op1 + op2)
    elif t == '-':
        op1, op2 = pop_two(s)
        s.push(op1 - op2)
    elif t == '*':
        op1, op2 = pop_two(s)
        s.push(op1 * op2)
    elif t == '/':
        op1, op2 = pop_two(s)
        s.push(op1 / op2)
    else:
        print "Unknown operand"
print 'R: ', s.top()
```

Cues

- Una cua és una estructura lineal en la que el primer element emmagatzemant és sempre el primer en ser esborrat (FIFO). La seva estructura recorda una cua de persones en una botiga:



- get() Retorna l'element del front de la cua.
- put(e) Encua l'element e.
- remove() Esborra l'element del front de la cua.
- empty() Retorna cert si la cua és buida.

Implementació sobre llistes

Una implementació sobre llistes:

```
class Queue(object):
    def __init__(self):
        self._q = list()

    def get(self):
        return self._q[0]

    def put(self,e):
        self._q.append(e)

    def remove(self):
        del self._q[0]

    def __len__(self):
        return len(self._q)
```

Exemple d'ús:

```
>>> q = Queue()
>>> q.put(100)
>>> q.put(200)
>>> q.put(300)
>>> q.get()
100
>>> len(q)
3
>>> q.remove()
>>> q.get()
200
>>> len(q)
2
```

Exemple d'ús: sumar els elements d'una cua

Assumiu que q és una cua d'enters i volem una funció que sumi tots els seus elements. Podem anar extraient tots elements, sumar-los i encuar-los de nou:

```
def sumaq(q):
    l = len(q)
    s = 0
    for i in range(l):
        e = q.get()
        s += e
        q.remove()
        q.put(e)
```

La implementació de la llibreria de Python

- Parlar del cost en temps d'una estructura de dades significa avaluar els cost da cadascuna de les seves operacions.
- La implementació que hem fet de les cues té el següent cost assimptòtic:

Operació	**Cost**
put	$O(1)$
get	$O(1)$
len	$O(1)$
remove	$O(n)$

- La causa rau en el fet que afegir o treure elements d'una llista Python pel davant implica moure tots els elements de la llista.
- El mòdul `collections` de la llibreria de Python conté una implementació d'una llista de doble entrada (dequeue). La implementació és eficient, $O(1)$, treballant pels dos extrems de la cua indistintament.

Feina per aquesta setmana

1. Estudi de la teoria. A partir de les transparències i les notes que heu pres.
2. Incrementar el xuletari i el glossari del tema.
3. Solució dels problemes del tema.