Esquemes algorísmics. Recorregut i cerca d'una seqüència

PID_00268286

Autors que han participat col·lectivament en aquesta obra
Lluís Beltrà
Kenneth Capseta
Maria Jesús Marco Galindo
Antonio Ponce

Idea i direcció de l'obra
Maria Jesús Marco Galindo

Disseny i edició gráfica
Asunción Muñoz

Material docent de la UOC

Universitat Oberta de Catalunya

Tercera edició: febrer 2021

Av. Tibidabo, 39-43, 08035 Barcelona
Realització editorial: FUOC

Continguts

Esquemes algorísmics
Recorregut i cerca d'una seqüència

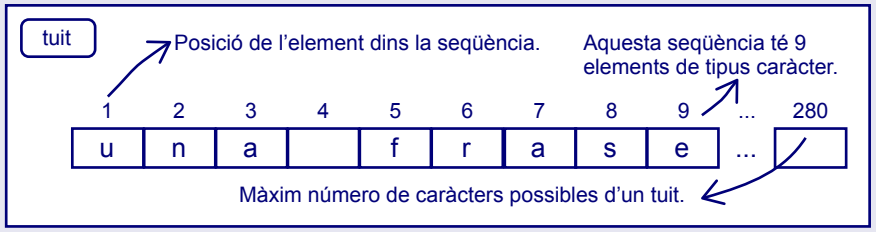
Per ser un bon programador, cal saber identificar patrons. Un patró vindria a ser el que es detecta quan una cosa es repeteix una vegada i un altra de la mateixa manera, com ara el patró cíclic en el que s'organitza el temps en una agenda; mesos, setmanes, dies i hores. El reconeixement de patrons aplicat a la resolució de problemes ens permet buscar semblances entre diversos problemes, per així poder aprofitar el que sabem sobre com solucionar un d'ells per aplicar-ho a la solució d'un altre que sigui similar. En el cas del codi, detectar patrons ens fa guanyar molta eficiència i eficàcia, és per això que són molt importants.

Recordeu que el **reconeixement de patrons** és un dels elements fonamentals del pensament computacional.

Ja coneixem l'algorísmica bàsica, així que, amb les tres estructures (seqüencial, condicional i iterativa), i els tipus bàsics i estructurats de dades (enter, caràcter, real, cadena, booleà, taula i tupla) podem planificar com trobar la solució a qualsevol problema que es pugui resoldre amb un ordinador (el que s'anomena un problema computable). No ens cal res més. Però podem fer-ho de forma més sistemàtica i eficient si utilitzem, per exemple, patrons. Els **esquemes algorísmics** són patrons de codi que ens ajuden a fer el tractament de manera més sistemàtica i eficient.

Exemple

Imaginem que estem escrivint un tuit i volem comprovar si hem arribat al màxim nombre de caràcters que permet Twitter que ara és de 280. El nostre tuit no és res més que una **seqüència de dades** de tipus caràcter que podem representar amb una taula:



Com representem el tuit en algorísmica?

El tuit el podem representar com una frase d'un màxim de 280 caràcters que és el que permet Twitter:

```
var
    tuit: taula[280] de caràcter;
fvar
```

Què és una SEQÜÈNCIA de dades?

Una SEQÜÈNCIA de dades és una successió d'elements ordenats d'una forma determinada. Una seqüència pot ser de diferents tipus de dades (caràcter, enter, etc.) i sempre té un final (és finita).

Entendre el concepte de seqüència de dades és important ja que gran part dels algorismes mínimament complexes consisteixen a repetir un conjunt de càlculs (un patró) en una seqüència de dades. Una de les estructures de dades que millor representa les seqüències és la taula.

Un cas molt habitual de patró que es repeteix quan tractem problemes que treballen amb seqüències de dades és el **recorregut**. Fer un recorregut vol dir passar per tots els elements de la seqüència un darrere l'altre. Un altre patró molt comú és la **cerca** que consisteix en buscar un element determinat en una seqüència d'elements.

Esquema de recorregut d'una seqüència

Pensem com dissenyar un patró, un esquema algorísmic, que ens permeti recórrer tots els elements d'una seqüència i anar-los tractant un a un de forma ordenada, des del primer fins a l'últim…

Per recórrer tota la seqüència ens haurem de situar en el primer element, és a dir, a la primera posició de la seqüència. Després, haurem d'accedir al element i així successivament anirem passant per tots els elements de la seqüència fins que arribem a l'últim element. Ens caldrà doncs una estructura iterativa.

Exemple

Imaginem ara que volem comptar el nombre de frases completes que conté un tuit de 280 caràcters. Sabem que una frase acaba perquè trobem un punt, així que haurem de recórrer tots els elements (caràcters) de la taula (que representa el tuit) i anar comptant els punts que hi trobem:

```
algorisme comptarFrasesTuit
var
    i, comptador: enter;
    tuit : taula[280] de caràcter;
fvar

    i := 1;
    comptador := 0;
    llegeixTuit(tuit);
    mentre i ≤ 280 fer

        si tuit[i] = '.' aleshores
            comptador := comptador + 1;
        fsi
        i := i + 1;
    fmentre
    escriurePantalla(comptador);
falgorisme
```

En aquest cas la seqüència de caràcters d'un tuit està representada per una taula.

Inicialitzem l'índex de la taula a 1 per situar-lo a la primera posició de la taula.

Inicialitzem el comptador de frases acabades.

Separem el problema de llegir el tuit en una altra funció de la que ja ens ocuparem més endavant. Estem aplicant la **descomposició de problemes**.

Iterem per recorrer les 280 posicions que té la seqüència.

Sabem que s'ha acabat una frase perquè trobem un punt.

Incrementem el comptador de frases completes.

Avançar al següent element de la taula: consisteix únicament a incrementar l'índex, la variable *i*.

Podem llegir el tuit del teclat amb la següent funció que rep una variable de tipus taula buida com a paràmetre d'entrada i la modifica omplint-la amb el text llegit per teclat:

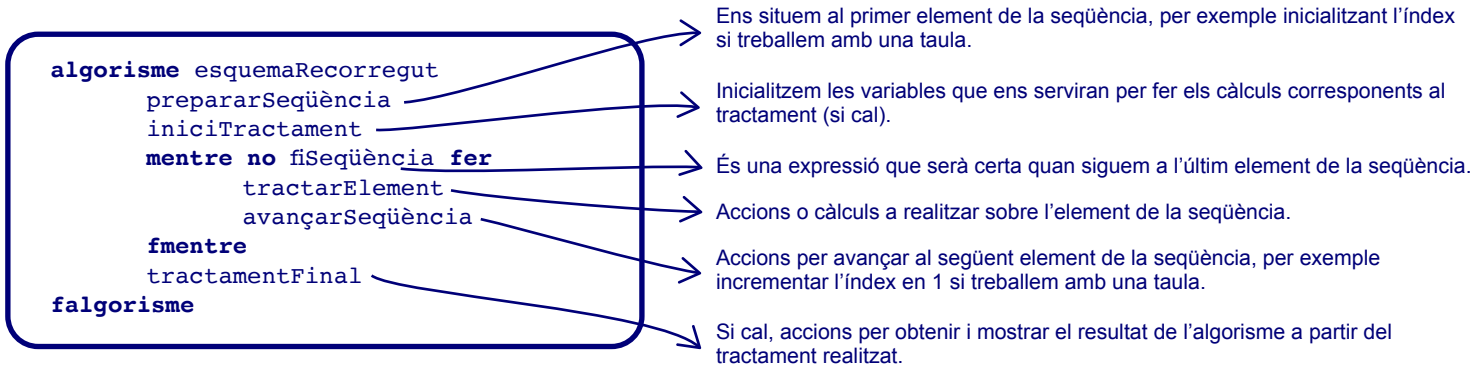
```
funció llegeixTuit (t: taula[280] de caràcter)
var
    i: enter;
fvar
    i := 1;
    mentre i ≤ 280 fer
    t[i] := llegirTeclat();
    i := i + 1;
    fmentre
ffunció
```

Aquesta funció espera que s'introdueixin per teclat 280 caràcters, ni més ni menys. No serviria per llegir tuits de menys caràcters.

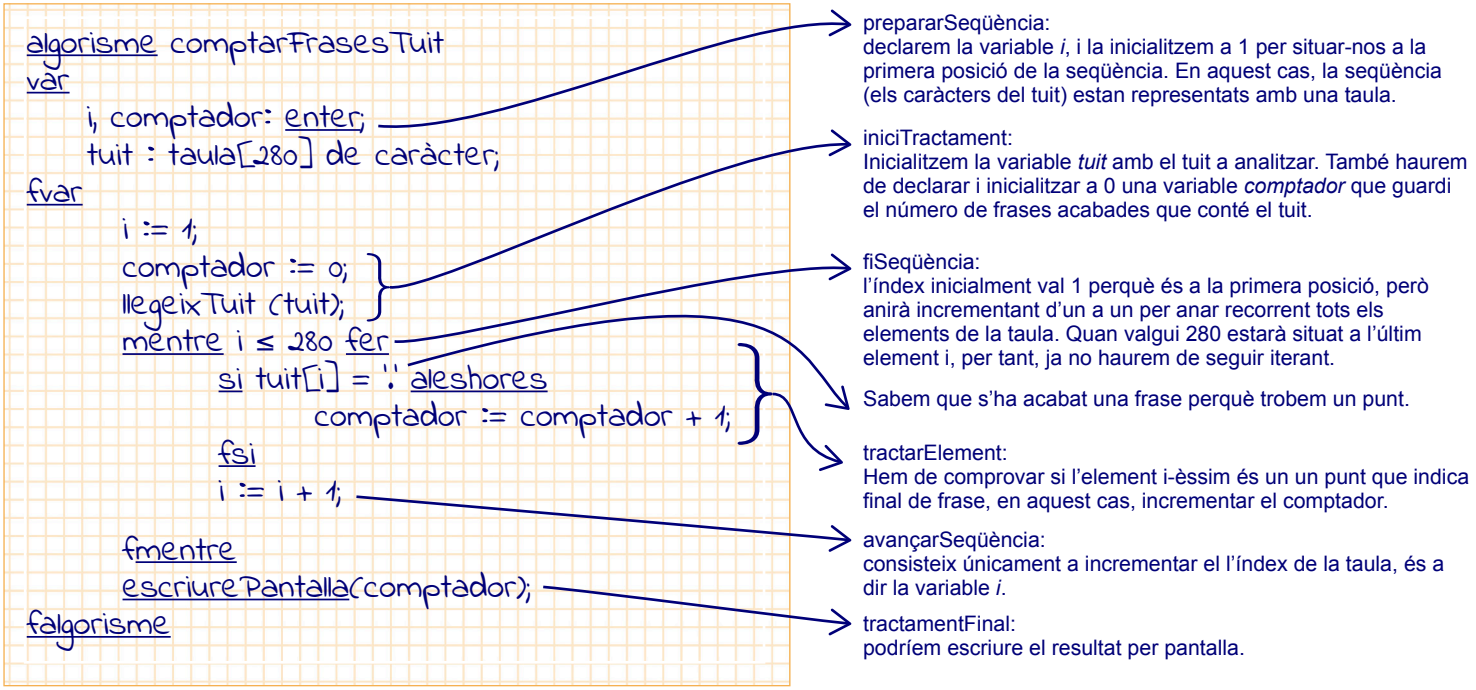
Fixeu-vos que les funcions poden modificar els paràmetres que reben sempre que aquests siguin de tipus estructurats. En aquest cas se'ls anomena **paràmetres d'entrada/sortida**.

Pot ser que la darrera frase no l'hàgim pogut acabar perquè hem gastat els 280 caràcters i no ens ha cabut. Així que, el nostre algorisme no comptarà aquesta darrera frase del tuit com acabada.

Sintaxi de l'esquema de recorregut



Busquem l'esquema de recorregut en l'exemple anterior on comptàvem les frases acabades en un tuit:



Com veieu, amb aquest algorisme hem pogut fer un recorregut per tots els elements de la taula. El mateix algorisme podria servir per llegir qualsevol seqüència, encara que tingués milers d'elements. Per exemple, si s'ampliés de nou la longitud màxima d'un tuit només caldria canviar la longitud màxima de la taula que ara és de 280, però l'esquema seria el mateix. O, si volem comptar el nombre de lletres 'a' del tuit, només hem de canviar '.' per 'a' i l'esquema segueix sent vàlid perquè en tots els casos fem el mateix: un recorregut per tots els elements de la taula.

Un altre esquema de recorregut d'una seqüència

Sabem que les seqüències sempre són finites, però hi haurà ocasions en les quals no coneixem el seu número d'elements. En aquests casos, el que sí que necessitem saber sempre és com detectar que hem arribat al final de la seqüència. Una manera alternativa de saber-ho és conèixer quin serà el darrer valor de la sequència, l'últim element.

Exemple

En l'exemple anterior hem treballat amb una seqüència de la qual coneixíem el número d'elements, la longitud màxima d'un tuit (280 caràcters). Imaginem ara que tenim una seqüència de números que volem sumar. No sabem quants, però sí que sabem que la seqüència acabarà amb el número 0.

L'algorisme que crearem s'encarregarà d'anar sumant els números que anirem llegint del teclat (entrada estàndard). Quan vulguem finalitzar, simplement haurem de teclejar un zero i així indiquem que hem arribat al final de la seqüència:

```
algorisme sumaNúmeros
var
    num, suma: enter;
fvar

    num := llegirTeclat();
    suma := 0;

    mentre num ≠ 0 fer

        suma := suma + num;

        num := llegirTeclat();

    fmentre

        escriurePantalla(suma);

falgorisme
```

preparar Seqüència:
declarem la variable *num*, on hi emmagatzemarem cada número que anirem introduint a través del teclat.

preparar Seqüència:
en aquest cas, a més, abans d'entrar a la composició iterativa, hem de llegir el primer element de la seqüència.

iniciTractament:
inicialitzem la variable *suma* on anirem guardant el resultat d'anar sumant els valors que anirem introduint.

fiSeqüència:
comprovar si ja hem arribat al final, és a dir, si el número introduït per teclat és un 0.

tractarElement:
accions a realitzar sobre cada element de la seqüència. En aquest cas sumem l'element actual a la variable suma.

avançarSeqüència:
consisteix a llegir el següent element de la seqüència, en aquest cas, el següent número que s'introduirà pel teclat.

tractamentFinal:
mostrem per pantalla el valor de la variable *suma* que conté la suma de tots els elements de la seqüència que hem anat introduint.

Hem pogut fer un recorregut per tots els elements de la seqüència fins a arribar a l'últim element (el número 0).

En aquest cas no ha calgut representar la seqüència de números amb una taula, no sempre les seqüències de dades es representen amb taules.

Totes les seqüències que podem tractar han de ser finites i sempre haurem de tenir alguna manera de saber quan hem arribat al final, ja sigui perquè sabem quants elements conté la seqüència o perquè coneixem quin és l'últim element (que evidentment ha de ser un element distintiu i diferent de tots els anteriors).

Per a què serveix...?

Concepte clau

Un **esquema algorísmic** és un patró que podem seguir per resoldre un problema prototípic habitual. Així no hem de pensar-lo de nou i reinventar cada vegada la solució. Amb això, guanyem eficiència i eficàcia ja que si apliquem bé l'esquema és segur que la solució funcionarà correctament.

L'**esquema de recorregut** es pot fer servir per resoldre problemes en els quals, per trobar la solució calgui, recórrer (revisar, tractar) totes i cadascuna de les dades d'una seqüència. Aquesta seqüència de dades ha de ser finita i sempre haurem de tenir manera de saber quan hem arribat al final, ja sigui perquè sabem quants elements conté la seqüència o perquè coneixem quin és l'últim element que evidentment ha de ser un element distintiu i diferent de tots els anteriors.

L'esquema segueix sempre una mateix estructura:

1. Obtenir el primer element.
2. Inicialitzar les variables.
3. Iterar fins a arribar al final i dins la iteració:
   - **3.1** Fer el tractament que calgui a cada element per resoldre el problema.
   - **3.2** Passar al següent element.
4. Si cal, calcular i mostrar el resultat final un cop tractades totes les dades de la seqüència.

```
algorisme esquemaRecorregut
        prepararSeqüència
        iniciTractament
        mentre no fiSeqüència fer
                tractarElement
                avançarSeqüència
        fmentre
        tractamentFinal
falgorisme
```

Una **seqüència** és una successió ordenada de dades. A vegades la representarem algorísmicament amb una taula, però no sempre ha de ser així. El que sí que és imprescindible perquè puguem aplicar aquest esquema és que sigui finita i que tinguem alguna manera de saber quan hem arribat al final, bé perquè sabem la seva longitud, bé perquè tenim manera d'identificar el darrer element que és distinitiu i diferent de la resta (a aquest element se l'anomena sentinella).

Esquema de cerca d'una seqüència

I si el problema que tenim implica buscar un element dins d'una seqüència de dades?

Imaginem que som uns apassionats de les olimpíades i volem saber si l'any que fem 40 anys podrem celebrar-ho anant a veure les olimpíades. Coneixem la seqüència prevista dels anys olímpics de les dues properes dècades: 2021, 2024, 2028, 2032, 2036, 2040. Hi haurà l'any del nostre 40è aniversari en aquesta seqüència?

Podem pensar a fer un recorregut per la seqüència, passant per, i tractant tots els seus elements. Però fixem-nos que potser no caldrà recórrer tots els elements; per exemple, si fem els 40 anys el 2028, quan arribem a trobar l'any 2028 ja sabem que el nostre 40è aniversari el podrem celebrar en unes olimpíades i no caldrà seguir revisant la resta d'elements de la seqüència fins al final.

El que estem fent és **cercar** un element en una seqüència, així que quan el trobem ja no caldrà que continuem fins al final de la seqüència. Una cerca segueix també, un esquema, un patró.

Exemple

El següent algorisme mira si l'any 2008 va ser un any olímpic o no. Partim del supòsit que tenim una seqüència de números corresponents als darrers anys en què hi ha hagut olimpíades:

	1	2	3	4	5	6
Anys	1996	2000	2004	2008	2012	2016

Ho podem representar com una taula d'enters:

```
var
    anysolimpics : taula [6] d'enter;
fvar
```

```
algorisme buscaAnyolímpic
var
    i: enter;
    anysolímpics: taula [6] d'enter;
    trobat: booleà;
fvar
    i := 1;
    trobat := fals;
    anysolímpics[1]:1996;
    anysolímpics[2]:2000;
    anysolímpics[3]:2004;
    anysolímpics[4]:2008;
    anysolímpics[5]:2012;
    anysolímpics[6]:2016;

    mentre i ≤ 6 i no trobat fer
        trobat := (anysolímpics[i] = 2008)
        si trobat aleshores
            escriurePantalla("L'any 2008 es van disputar
            unes olimpíades");
        fsi

        i := i + 1;
    fmentre

    si no trobat aleshores
        escriurePantalla("L'any 2008 no van haver-hi
        olimpíades");
    fsi
falgorisme
```

Ja hem trobat el valor que estàvem buscant i podem aplicar les accions que ens interessin. En aquest cas, simplement escriurem per pantalla que el 2008 es van celebrar unes olimpíades.

Potser les accions de l'estructura iterativa es veuen més clares així:

```
mentre i ≤ 6 i no trobat fer
    si (anysolímpics[i] = 2008) aleshores
        escriurePantalla("L'any 2008 es van disputar unes
        olimpíades");
        trobat := cert;
    si no
        trobat := fals;
    fsi
        i := i + 1;
fmentre
```

Penseu si aquest algorisme es podria optimitzar. Per exemple, si busquem l'any 2007, quan arribem a llegir l'any 2008, cal continuar llegint?. No, en aquest cas, si sabem del cert que la seqüència està ordenada, ja podem acabar encara que no hàgim recorregut tots els elements ni trobat el que cerquem perquè ja sabem que no el trobarem. Ara bé, si la seqüència no està ordenada, sí que cal seguir buscant fins el final mentre no trobem el 2007 perquè podria estar a la darrera posició.

Sintaxi de l'esquema de cerca

```
algorisme esquemaCerca
        var
          trobat: booleà;
        fvar
        prepararSeqüència
        trobat := fals;
        iniciTractament
        mentre no fiSeqüència i no trobat fer
               actualitzarTrobat
               si no trobat aleshores
                  tractarElement
                  avançarSeqüència
               fsi
        fmentre
        tractamentFinal
falgorisme
```

Obtenim el primer element de la seqüència, ens situem al primer element de la seqüència.

Iniciem les variables que ens serviran per fer els càlculs corresponents al tractament (si cal).

És una expressió que serà certa quan siguem al darrer element de la seqüència.

Accions per determinar si l'element actual és el que estem buscant. En cas afirmatiu, la variable *trobat* pren el valor de cert.

Accions a realitzar sobre un element de la seqüència.

Accions que avançaran cap al següent element.

Accions finals per obtenir el resultat de l'algorisme a partir de la cerca realitzada (si cal).

A diferència de l'esquema de recorregut, on recorrem i tractem tots els elements de la seqüència, amb aquest nou esquema recorrem la seqüència únicament fins que trobem un element que compleixi una determinada condició o que ja tinguem clar que no el trobarem, com passava a l'exemple dels anys olímpics si la seqüència estava ordenada.

Cal tenir present que aquesta condició no té perquè referir-se a un element amb un valor concret (com per exemple, quan volem trobar la lletra 'a' en una frase), sinó que pot ser una condició més general (trobar el primer número parell d'una seqüència d'enters, la primera paraula capicua d'un text, etc.).

Busquem l'esquema de cerca en l'exemple anterior, on cercàvem si l'any 2008 era olímpic:

```
algorisme buscaAnyOlímpic
var
    i: enter;
    anysolímpics: taula [6] d'enter;
    trobat: booleà;
fvar
    i := 1;
    trobat := fals;
    anysolímpics[1] := 1996; anysolímpics[2] := 2000; anysolímpics[3] := 2004;
    anysolímpics[4] := 2008; anysolímpics[5] := 2012; anysolímpics[6] := 2016;

    mentre i ≤ 6 i no trobat fer
       trobat := (anysolímpics[i] = 2008)
       si trobat aleshores
          escriurePantalla("L'any 2008 es van disputar unes olimpíades");
       fsi
       i := i + 1;
    fmentre

       si no trobat aleshores
          escriurePantalla("L'any 2008 no van haver-hi olimpíades");
       fsi
falgorisme
```

prepararSeqüència:
inicialitzem la variable *i* a 1 per situar-nos a la primera posició de la taula.

iniciTractament:
inicialitzem la variable *anysOlímpics* amb els valors indicats anteriorment. També haurem de declarar i inicialitzar a **fals** una variable *trobat* que ens indicarà si ja s'ha trobat el valor buscat o no.

fiSeqüència:
L'index inicialment val 1 perquè és a la primera posició, però s'anirà incrementant d'un a un per anar recorrent tots els elements de la taula fins a arribar al final o bé fins a trobar evalor 2008 que estem buscant. Si es compleix una d'aquestes dues condicions, sortirem de la composició iterativa i deixarem de recórrer la taula.

actualitzarTrobat:
comprovem si el valor actual és el que estem buscant. En cas afirmatiu, la variable *trobat* passa a tenir valor **cert**, de tal manera que en la següent iteració ja no es compliran les condicions per seguir iterant i deixarem de recórrer la taula.

avançarSeqüència:
consisteix únicament a incrementar la variable *i*.

tractamentFinal:
si en sortir de la composició iterativa el valor de la variable *trobat* segueix essent **fals**, significa que no hem trobat el valor que buscàvem dins la taula.

Per a què serveix...?

Concepte clau

L'**esquema de cerca** es pot fer servir per resoldre problemes en els quals la solució impliqui buscar i tractar un element (o conjunt d'elements) concret en una seqüència. És, de fet, una variació de l'esquema de recorregut però, en aquest cas, és possible que no calgui tractar tota la seqüència de dades.

L'esquema segueix sempre una mateixa estructura:

1 Obtenir el primer element.

2 Inicialitzar les variables.

3 Iterar fins a trobar l'element que busquem o fins a arribar al final.

- **3.1** Fer el tractament que calgui per resoldre el problema.
- **3.2** Passar al següent element si encara no hem trobat el que busquem.

4 Si cal, calcular i mostrar el resultat final un cop hàgim trobat el que busquem dins de la seqüència o haguem arribat al final.

```
algorisme EsquemaCerca
        var
           trobat: booleà;
        fvar
        prepararSeqüència
        trobat := fals;
        iniciTractament
        mentre no fiSeqüència i no trobat fer
                actualitzarTrobat
                si no trobat aleshores
                   tractarElement
                   avançarSeqüència
                fsi
        fmentre
        tractamentFinal
falgorisme
```

Fixeu-vos que si l'element que estem buscant resulta que és l'últim de la seqüència, estarem fent, de fet, un recorregut per tota la seqüència. per això diem que l'esquema de cerca és un cas particular de l'esquema de recorregut.

Recorregut i cerca d'una seqüència en JS

Com serien aquests algorismes en JavaScript?

Exemple

Comptar frases acabades d'un tuit.

```
var i = 0;
var comptador = 0;
//Frase cèlebre de Grouxo Marx
var tuit = "Aquests són els meus principis. Si no us agraden, en tinc uns altres.";
// En JavaScript les cadenes de text són accessibles com si fossin un
// array i tenen la propietat length per saber la seva extensió.

while (i < tuit.length){
    console.log(i, tuit[i]);
    if(tuit[i]=="."){
        comptador++;
    }
    i++;
}
console.log(comptador);
```

Sumar una seqüència de números.

```
var suma = 0;
console.log(suma);
var i = 0;
var numerosSuma = [3, 1, 2, 3, 0, 5, 8];
while (numerosSuma[i]!=0) {
  suma = suma + numerosSuma[i];
  i++;
}
console.log(suma);
```

Esbrinar si un any és olímpic.

```
var i = 0;
var anys = [1996, 2000, 2004, 2008, 2012, 2016];
var trobat = false;
while (i < 6 && !trobat){
    trobat = anys[i] == 2008;
    i++;
    if (trobat){
        console.log("l'any 2008 es van disputar unes olimpiades!");
    }
}
if (!trobat){
    console.log("l'any 2008 no hi van haver olimpiades");
}
```

Podeu provar aquests codis a la web de PythonTutor: