PAC3: Tercera Prova d'Avaluació Continuada

Format i data de lliurament

Cal lliurar la solució en un fitxer de tipus **pdf**.

La data límit per lliurar la solució és el **dilluns, 6 de Maig de 2019** (a les 23:59 hores).

Presentació

El propòsit d'aquesta tercera PAC és comprovar que has adquirit els conceptes explicats en els capítols '*Mètodes de Cerca'* i *'Complexitat'.*

Competències

Transversals

- Capacitat de comunicació en llengua estrangera.
- Coneixements de programació amb llenguatge algorísmic.

Específiques

- Capacitat de dissenyar i construir algorismes informàtics mitjançant tècniques de desenvolupament, integració i reutilització.

Objectius

Els objectius d'aquesta PAC són:

- Adquirir els conceptes teòrics explicats sobre els mètodes de cerca.
- Interpretar els algorismes de cerca, sent capaços de simular el seu funcionament donada una entrada.
- Implementar qualsevol mètode de cerca en un algorisme i avaluar-ne el seu rendiment.
- Adquirir els conceptes teòrics explicats sobre les tècniques d'anàlisi d'algorismes.
- Analitzar la complexitat d'una funció, calculant la seva funció de temps d'execució, i expressar aquesta complexitat amb notació asimptòtica.

Descripció de la PAC a realitzar

Raona i justifica totes les respostes.

Les respostes incorrectes **no** disminueixen la nota.

Tots els dissenys i implementacions han de realitzar-se en **llenguatge algorísmic**. Els noms dels tipus, dels atributs i de les operacions s'han d'escriure en anglès. No és obligatori escriure en anglès els comentaris i els missatges d'error, tot i que es valorarà positivament que es faci, ja que és l'estàndard.

Recursos

Per realitzar aquesta prova disposes dels següents recursos:

Bàsics

- Materials en format **web de l'assignatura.**
- **El llibre *"Manual d'Algorísmica".*** En els capítols *Mètodes de Cerca* i *Tècniques d'Anàlisi d'Algorismes* pots trobar el material necessari per realitzar la prova.
- **Fòrum de l'aula de teoria.** Disposes d'un espai associat en l'assignatura on pots plantejar els teus dubtes sobre l'enunciat.

Complementaris

- **Cercador web.** La forma més ràpida d'obtenir informació ampliada i extra sobre qualsevol aspecte de l'assignatura.
- La solució de la PAC d'un semestre anterior.

Criteris de valoració

Per a la valoració dels exercicis es tindrà en compte:

- L'adequació de la resposta a la pregunta formulada.
- Utilització correcta del llenguatge algorísmic.
- Claredat de la resposta.
- Completesa i nivell de detall de la resposta aportada.

Avís

Aprofitem per recordar que **està totalment prohibit copiar en les PACs** de l'assignatura. S'entén que hi pot haver un treball o comunicació entre els alumnes durant la realització de l'activitat, però el lliurament d'aquesta ha de ser individual i diferenciat de la resta.

Així doncs, els lliuraments que continguin alguna part idèntica respecte treballs d'altres estudiants seran considerats còpies i tots els implicats (sense que sigui rellevant el vincle existent entre ells) suspendran la prova lliurada.

Exercici 1: Conceptes bàsics sobre cerca (10%)

Tasca: Digues si són certes o no les sentències següents. En cas que no ho siguin, especifica la raó:

i) Sobre un vector ordenat, sempre és més eficient realitzar una cerca binària que una cerca lineal.

Fals. No sempre, si l'element a cercar es troba en una de les primeres posicions, és més eficient una cerca lineal que una cerca binària. També es podria donar el cas que es descarti que l'element a cercar no es troba al vector en una de les primeres comprovacions.

ii) La complexitat de l'algorisme de cerca lineal és O(n) mentre que la complexitat de la cerca binària és O(log n).

Cert.

iii) Emprant una taula d'hash, si dos ids diferents poden ser assignats a una mateixa posició de la taula, es produeix una col·lisió i és recomanable emprar una altra tècnica.

Fals. Quan es produeixen col·lisions es poden cercar sistemes com el hashing obert o el hashing tancat per solucionar-ho.

iv) És possible realitzar una cerca eficient en una taula de hash mitjançant la combinació d'un accés directe a l'índex que ens retorna la funció d'hash i una cerca lineal per trobar l'element buscat en aquesta entrada

Cert.

Exercici 2: Algorismes de cerca (20%)

Tasca: Donat el següent vector, aplica l'algorisme indicat per buscar l'element que es desitja. Escriu el resultat i la seqüència d'índexs consultats en el procés de cerca pel vector:

1	2	3	4	5	6	7	8	9	10	11	12
-15	-8	-3	0	2	5	9	14	20	27	35	44

i) Aplica l'algorisme de *cerca* **lineal** o **seqüencial** per trobar l'element 20.

L' element 20 està en la posició 9 del vector:

índex 1: -15 ≠ 20
índex 2: -8 ≠ 20
índex 3: -3 ≠ 20
índex 4: 0 ≠ 20
índex 5: 2 ≠ 20
índex 6: 5 ≠ 20
índex 7: 9 ≠ 20
índex 8: 14 ≠ 20
índex 9: 20 = 20 Trobat!

ii) Aplica l'algorisme de *cerca* **lineal** per trobar l'element 3.

L'element 3 no es troba en el vector:

índex 1: -15 ≠ 3
índex 2: -8 ≠ 3
índex 3: -3 ≠ 3
índex 4: 0 ≠ 3
índex 5: 2 ≠ 3
índex 6: 5 ≠ 3
índex 7: 9 ≠ 3
índex 8: 14 ≠ 3
índex 9: 20 ≠ 3
índex 10: 27 ≠ 3
índex 11: 35 ≠ 3
índex 12: 44 ≠ 3 No trobat

Si es coneix a priori que el vector està ordenat, llavors també és correcta la solució que atura la cerca en l'índex 6, ja que a partir d'aquest índex tots els elements seran majors que 5, i per tant el valor 3 no estarà entre ells.

iii) Aplica l'algorisme de **cerca binària** per trobar l'element 20.

L'element 20 es troba en la posició 9 del vector:

E=1 D=12	(1+12) /2 = 13/2 = 6	índex 6: 5 < 20
E=7 D=12	(7+12) /2 = 19/2 = 9	índex 9: 20 = 20. Trobat

iv) Aplica l'algorisme de **cerca binària** per trobar l'element 3.

L'element 3 no es troba en el vector:

E=1 D=12	(1+12) /2 = 13/2 = 6	índex 6: 5 > 3
E=1 D=5	(1+5) /2 = 6/2 = 3	índex 3: -3 < 3
E=4 D=5	(4+5) /2 = 9/2 = 4	índex 4: 0 < 3
E=5 D=5	(5+5) /2 = 10/2 = 5	índex 5: 2 < 3.
E=6 D=5		No trobat

Exercici 3: Ús dels algorismes de cerca (20%)

Tasca: Tornant al sistema de gestió de vins del celler, en aquest exercici s'utilitzaran els tipus de dades que es mostren a continuació i que estan basats en la solució de l'exercici de la PAC1.

```
const
        MAXWINES : integer := 80;
end const
type
        tWine = record
                        id : integer;
                        brand : string;
                        areaDO: string;
                        alcoholDegree: real;
                        holdingTime: integer;
                        price: real;
        end record

        tCellar = record
                        wineSet: vector [MAXWINES] of tWine;
                        numWines: integer;
        end record
end type
```

Se'ns demana implementar en llenguatge algorítmic la funció que a partir del celler (c) i d'un codi de vi (id) ens retorna el nombre de graus d'alcohol que té. En cas de que el vi no existeix, llavors retornarà el valor -1.

```
function getAlcoholDegree(c: tCellar, id: integer) : real
```

El conjunt de vins *wineSet* està implementat en una taula ordenada en ordre creixent pel camp *id* dels vins. Així doncs, per implementar la cerca s'haurà d'utilitzar el **mètode de cerca binària.**

```
function getAlcoholDegree(c: tCellar, id: integer) : real
        var
                left, right, middle: integer;
                found: boolean;
                res: real;
        end var
        left := 1;
        right := c.numWines;
        found := false;
        res := -1.0;
        while (left ≤ right) and not found do
                middle := (left + right) div 2;
                if c.wineSet[middle].id = id then
                        found := true;
                        res := c.wineSet[middle].alcoholDegree;
                else
                        if c.wineSet[middle].id < id then
                                left := middle +1;
                        else
                                right := middle -1;
                        end if
                end if
        end while
        return res;
end function
```

Exercici 4: Conceptes bàsics sobre complexitat (20%)

Tasca: Respon les preguntes següents:

i) Què defineix T(n)?

T(n) defineix la funció de temps d'execució d'un algorisme l'entrada del qual té una grandària n. S'expressa sense unitats i representa el nombre d'operacions elementals que realitza l'algorisme, necessàries per obtenir la solució. Sempre es fa referència al temps d'execució en el pitjor cas.

ii) Calcula la complexitat computacional de les següents funcions de temps d'execució usant la notació asimptòtica:

a.	$T(n) = 500n + 300000$	$O(n)$	lineal
b.	$T(n) = (2+3n)\cdot\log(50n)$	$O(n.\log(n))$	quasi-lineal
c.	$T(n) = 1^n$	$O(1)$	constant
d.	$T(n) = 8000 + \log(n+1)$	$O(\log(n))$	logarítmica
e.	$T(n) = 5^n + 300n$	$O(5^n)$	exponencial
f.	$T(n) = 4 + 80n + 100\cdot n^2$	$O(n^2)$	quadràtica

iii) Ordena les complexitats computacionals de l'apartat anterior de més eficient a menys eficient.

c, d, a, b, f, e

iv) Quan parlem de l'eficiència, per quina raó parlem del cas pitjor? Raona la teva resposta.

El càlcul més proper a la realitat seria el del cas mitjà, però la complexitat els càlculs el fan impracticable ja que hauríem de conèixer la probabilitat de cadascuna de les entrades.

Per això, el cas pitjor és un càlcul abordable que ens permet deduir el cost màxim de l'algorisme i ens dona una imatge propera a la realitat.

Exercici 5: Anàlisi d'algorismes (30%)

Tasca: Respon a les preguntes següents.

i) Donada la següent definició del temps d'execució, calcula la funció *T(n)* sense que aparegui cap definició recursiva i explica el procés que has seguit per obtenir el resultat:

$$T(n) = \begin{cases} 1, & \text{si } n = 0 \\ T(n-1) + 3n, & \text{si } n > 0 \end{cases}$$

El mètode per resoldre aquest tipus d'equacions consisteix en aplicar la definició recursiva de forma repetida un total de *i* vegades, fins que veiem quina forma pren l'expressió resultant:

$$\begin{aligned} T(n) &= \mathbf{T(n-1) + 3{\cdot}n} = T(n-1-1) + 3{\cdot}(n-1) + 3n = \\ &= T(n-2) + 3{\cdot}n - 3 + 3n = \\ &= \mathbf{T(n-2) + 6{\cdot}n - 3} = (T(n-1-2) + 3{\cdot}(n-2)) + 6{\cdot}n - 3 = \\ &= T(n-3) + 3n - 6 + 6n - 3 = \\ &= \mathbf{T(n-3) + 9{\cdot}n - 9} = (T(n-1-3) + 3{\cdot}(n-3)) + 9n - 9 = \\ &= T(n-4) + 3n - 9 + 9n - 9 = \\ &= \mathbf{T(n-4) + 12{\cdot}n - 18} = \ldots \\ &= \boldsymbol{T(n-i) + 3 \cdot n \cdot i - 3 \cdot \sum_{j=0}^{i-1} j} \end{aligned}$$

Esbrinem quin valor ha de prendre *i* per poder aplicar el cas base de la definició de *T(n)*:

$$\begin{aligned} n - i &= 0 \\ n &= i \end{aligned}$$

A continuació, completem el càlcul aprofitant que l'expressió *T(n - i)* quan *i = n* es transforma en *T(0)*, que segons la definició de l'enunciat és 1:

$$\boldsymbol{T(n) = T(n-n) + 3 \cdot n \cdot n - 3 \cdot \sum_{j=0}^{n-1} j} =$$

$$= \boldsymbol{T(0) + 3 \cdot n^2 - 3 \cdot \frac{n \cdot (n-1)}{2}} =$$

$$= 1 + 3 \cdot n^2 - 3 \cdot \frac{n^2 - n}{2} = 1 + \frac{6n^2}{2} - \frac{3n^2 - 3n}{2} =$$

$$= \frac{3n^2}{2} + \frac{3n}{2} + 1$$

ii) Calcula la complexitat computacional de la funció **perfect_number** i explica el procés que has seguit per obtenir el resultat.

```
function perfect_number (num: integer) : boolean
  var
        res: integer;
  end var
        res:=0;                                  (1)
        for i:=1 to num-1 do                     (2)
              if (num mod i )=0 then             (3)
                    res : = res + i;             (4)
              end if                             (5)
        end for                                  (6)
        return res=num;                          (7)
end function                                     (8)
```

La funció *perfect_number*, comença amb una assignació (línia 1) que és una operació elemental amb un temps constant k_1.

Després hi ha un bucle (línies 2-6) que s'executa *num-1* vegades, depenent del nombre a analitzar. Per simplificar-ho, ho representarem amb emprant *n*.

Dintre del bucle, a cada iteració incrementam el valor de la variable i (assignació i suma) i comparam aquest valor amb num-1, que és la condició de finalització. A més, el pitjor cas de tots és que sempre es compleixi la condició de la sentència condicional que hi ha dintre d'ell (línies 3-5). Les operacions que hi ha tenen cost constant: l'operació mòdul (línia 3), la comparació (línia 3) i la suma i l'assignació de la variable *res* (línia 4). El resultat d'aquesta suma, és el cost d'una iteració i ho agruparem amb el valor k_3.

El temps total del bucle és la suma del cost d'inicialització del comptador de bucle *i* (línia 2) k_2 i la multiplicació del cost d'una iteració pel nombre d'iteracions $k_2 + k_3 \cdot N$.

Per acabar, ens falta sumar la comparació entre el paràmetre *num* i la variable *res*, que té temps d'execució constant, k_4.

Finalment, la funció del temps d'execució de la funció *perfect_number* és:

$$\mathbf{T(N) = k_1 + k_2 + k_3 \cdot N + k_4}$$

Per tant, la funció té una complexitat lineal **O(n),** on *n* depèn del valor del paràmetre *num*.