PAC3 – DANIEL GALEANO SANCHO

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, 11 de Maig de 2020 (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) En una cerca lineal si la llista està ordenada, no sempre és necessari recórrer tota la llista

Cert.

ii) Sobre un vector ordenat, si l'element a cercar no es troba en el vector, qualsevol algorisme de cerca tindrà una complexitat de $O(n)$.

Fals. Utilitzant la cerca binària s'obté una complexitat $O(n)$ i utilitzant taules hash, $O(1)$.

iii) A les taules de hash, després de cada nova inserció, si es produeix una col·lisió, s'ha de reordenar tota la taula.

Fals. Al hashing obert, es forma una llista encadenada i, al hashing tancat, es busca la següent posició lliure dins de la pròpia taula, no es fa cap ordenació.

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 de 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
"Al"	"Au"	"B"	"C"	"Cl"	"Fe"	"H"	"Mg"	"N"	"O"	"Pt"	"S"

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

Índex	**Comparació**
1	"Al" ≠ "H"
2	"Au" ≠ "H"
3	"B" ≠ "H"
4	"C" ≠ "H"
5	"Cl" ≠ "H"
6	"Fe" ≠ "H"
7	"H" = "H"

Es troba que $index = 7$, després de set iteracions.

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

Índex	**Comparació**
1	"Al" ≠ "Ar"
2	"Au" ≠ "Ar"
3	"B" ≠ "Ar"
4	"C" ≠ "Ar"
5	"Cl" ≠ "Ar"
6	"Fe" ≠ "Ar"
7	"H" ≠ "Ar"
8	"Mg" ≠ "Ar"
9	"N" ≠ "Ar"
10	"O" ≠ "Ar"
11	"Pt" ≠ "Ar"
12	"S" ≠ "Ar"

No es troba l'índex després de dotze iteracions, per tant, l'element "Ar" no es troba al vector.

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

First	Last	Middle	Comparació
1	12	6	"Fe" < "H"
7	12	9	"N" > "H"
7	8	7	"H" = "H"

Es troba que $index = 7$, després de tres iteracions.

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

First	Last	Middle	Comparació
1	12	6	"Fe" > "Ar"
1	5	3	"B" > "Ar"
1	2	1	"Al" < "Ar"
2	2	2	"Au" > "Ar"
2	1	-	-1

No es troba l'índex després de cinc iteracions, per tant, l'element "Ar" no es troba al vector..

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

Tasca: Tornant al sistema de gestió del torneig d'escacs, en aquest exercici s'utilitzaran els tipus de dades que es mostren a continuació i que estan basats en la solució de l'exercici 1 de la PAC1..

```
type
    tPlayer = record
        id : integer;
        name : string;
        age : integer;
        nacionality : string;
        elo : integer;
    end record
    tChessTournament = record
        players : pointer to tPlayer;
        numPlayers : integer;
```

```
        end record
end type
```

Assumint que els jugadors estan ordenats en forma ascendent pel seu id, se'ns demana implementar en **llenguatge algorísmic** la funció **getPlayerElo** que a partir del torneig (**c**) i un identificador de jugador (**id**) retorna la seva puntuació (**elo**). Si el jugador no existeix retornarà el valor -1. Per implementar la cerca s'haurà d'utilitzar el **mètode de cerca binària**.

```
function getPlayerElo(c: tChessTournament, id: integer): integer
```

```
function getPlayerElo (c: tChessTournament, id: integer): integer
    var
        first, last, middle: integer;
    end var

    first := 1;
    last := c.numPlayers;

    while first <= last do
        middle = (first + last) div 2;
        if c.players[middle].id = id then
                return c.players[middle].elo;
        else
                if c.players[middle].id < id then
                        first := middle + 1;
                else
                        last = middle - 1;
                end if
        end if
    end while

    return -1;
end function
```

Exercici 4: Conceptes bàsics sobre complexitat (20%)

Tasca: Respon les preguntes següents:

i) Què representa $T(n)$?.

Representa la funció de temps d'execució d'un algorisme l'entrada del qual té mida n.

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

a. $T(n) = 10^n + 2\,n2$

$O(10^n)$ exponencial

b. $T(n) = \frac{1}{2} n \cdot \log_2(n)$

$O(n \log(n))$ quasi-lineal

c. $T(n) = \frac{n}{25}$

$O(n)$ lineal

d. $T(n) = 5 \cdot \log_2(n)$

$O(\log(n))$ logarítmica

e. $T(n) = 2n^2$

$O(n^2)$ quadràtica

f. $T(n) = 12059$

$O(1)$ constant

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

f, d, c, b, e, a.

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

Perquè el cas pitjor ens dóna una cota màxima per al cost d'un algorisme i calcular-lo no és tan complex com calcular el cas mitjà, que implicaria calcular la probabilitat de cadascuna de les entrades. Per tant, el cas pitjor ens dóna una aproximació a l'eficiència prou bona com perquè no haguem de fer servir el cas mitjà.

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:

$$T(n) = \begin{cases} 1, & si\ n = 1 \\ T(n/2) + n, & si\ n > 1 \end{cases}$$

Apliquem la definició de la funció fins que veiem quin comportament té per a $n = i$.

$$\begin{aligned} T(n) \quad & = \boldsymbol{T\left(\frac{n}{2}\right) + n} = T\left(\left(\frac{n}{2}\right)/2\right) + \frac{n}{2} + n = \\ & = \boldsymbol{T\left(\frac{n}{2^2}\right) + \frac{n}{2} + n} = T\left(\left(\frac{n}{2^2}\right)/2\right) + \frac{n}{2^2} + \frac{n}{2} + n = \\ & = \boldsymbol{T\left(\frac{n}{2^3}\right) + \frac{n}{2^2} + \frac{n}{2} + n} = \\ & = \cdots = \\ & = \boldsymbol{T\left(\frac{n}{2^i}\right) + \sum_{k=0}^{i-1}\left(\frac{n}{2^k}\right)} = \\ & = \left\{ \begin{matrix} aplicant\ l'expressió\ de\ sèrie\ geomètrica \\ \sum_{k=0}^{i} ar^k = a\left(\frac{1-r^{k+1}}{1-r}\right) \end{matrix} \right\} = \\ & = \boldsymbol{T\left(\frac{n}{2^i}\right) + 2n\left(1 - \frac{1}{2^i}\right)} \end{aligned}$$

Hem de trobar quin valor de i ens permet aplicar el cas base de $T(n)$:

$$\frac{n}{2^i}$$
$$n = 2^i$$
$$i = \log_2 n$$

Apliquem aquest cas al resultat que havíem trobat i apliquem el cas base de la definició de $T(n)$, $T(1) = 1$:

$$\begin{aligned} T(n) \quad & = T\left(\frac{n}{2^i}\right) + 2n\left(1 - \frac{1}{2^i}\right) = T\left(\frac{n}{2^{\log_2 n}}\right) + 2n\left(1 - \frac{1}{2^{\log_2 n}}\right) = \\ & = \left\{\begin{matrix} tenint\ en\ compte\ que \\ 2^{\log_2 n} = n \end{matrix}\right\} = T\left(\frac{n}{n}\right) + 2n\left(1 - \frac{1}{n}\right) = \\ & = T(1) + 2n - 2 = 1 + 2n - 2 = \mathbf{2n - 1} \end{aligned}$$

ii) Calcula la complexitat computacional de la funció **mystery** i explica el procés que has seguit per obtenir el resultat.
La funció de temps de *binarySearch(x: vector [N] of integer ,y: integer): integer* ve determinada per la fórmula $T(n) = \log_2 n$ essent n la mida del vector x.

function mystery (a: vector [N] of integer, b: vector [N] of integer): **integer**

```
var
        count: integer;
        i: integer;
end var
count := 0;                                              (1)
for i := 1 to N do                                       (2)
        if a[i] > 0 then                                 (3)
                count := count binarySearch (b, a[i]);   (4)
        end if                                           (5)
end for                                                  (6)
return count;                                            (7)
end function                                             (8)
```

Una assignació a (1) és una operació elemental amb un temps constant, k_1.

El bucle que hi ha entre (2) i (6) s'executa N vegades. Tant la inicialització i increment del bucle de (2) com la condició i accés a l'element del vector de (3), són operacions elementals amb un temps constant, k_2.

La línia (4) té el cost de la crida a la funció **binarySearch** que, tal i com indica l'enunciat, és $\log_2 N$.

Per tant, el cost d'una iteració del bucle serà $k_2 + \log_2 N$. Com que el bucle s'executa N vegades, el temps total del bucle serà $(k_2 + \log_2 N) \cdot N$

La funció del temps d'execució de **mystery** és:

$$T(n) = k_1 + k_2(1 + n) + n \cdot log_2 n$$

Finalment, **la complexitat de l'algorisme és quasi-lineal, $O(n \log n)$**.