Examen d'Anàlisi i Disseny d'Algorismes
primer parcial, 12 de novembre de 2014

Nom:
Cognoms:
NIU:

Nota: Cada pregunta te el seu valor entre (). L'examen es respon a continuació de les preguntes.

1.-(2 punts) La classe CDQueue implementa una llista doblement encadenada. Quins asserts s'han d'afegir per verificar les precondicions i quina és la invariant de la classe CDQueue? Considereu que els elements d'aquesta llista no poden ser null i que CNode no comprova res. Les precondicions i la invariant de la classe s'han d'expressar en Java.

```
public class CDQueue {
	class CNode {
		public Object m_Element;
		public CNode m_Next;
		public CNode m_Previous;
		CNode(Object e) {
			m_Element=e;
			m_Next=null;
			m_Back=null;
		}
	}
	CNode m_Front;
	CNode m_Back;
	public CDQueue() {
		m_Front=null;
		m_Back=null;
	}
	public void PushFront(Object e) {
		CNode n=new CNode(e);
		n.m_Next=m_Front;
		m_Front=n;
		if (n.m_Next==null) m_Back=n;
	}
	public void PushBack(Object e) {
		CNode n=new CNode(e);
		n.m_Previous=m_Back;
		m_Back=n;
		if (n.m_Previous==null)
			m_Front=n;
	}
	public Object PopFront() {
		Object e=m_Front.m_Element;
		m_Front=m_Front.m_Next;
		if (m_Front==null)
			m_Back=null;
		return e;
	}
	public Object PopBack() {
		Object e=m_Back.m_Element;
		m_Back=m_Back.m_Previous;
		if (m_Back==null)
			m_Front=null;
		return e;
	}
}
```

Solución:

En PushFront(Object e) assert e!=null;
En PushBack(Object e) assert e!=null;

En PopFront() assert m_Front!=null;
En PopBack() assert m_Back!=null;

Invariante:

```
	public boolean Invariant() {
		CNode n=m_Front;
		CNode ant=null;
		while (n!=null) {
			if (n.m_Element==null) return false;
			if (n.m_Previous!=ant) return false;
			ant=n;
			n=n.m_Next;
		}
		return ant==m_Back;
	}
```

2.-(2 punts) Calcula la complexitat d'un algorisme recursiu que tingui la seva complexitat definida d'aquesta manera. Fes els càlculs segons la tècnica del desplegat. Justifica la resposta.

$$T(n)=\begin{cases}1, \text{ si } n=1\\ n+4T(n/4) \text{ , si } n>1\end{cases}$$

$T(n) = n+4T(n/4) =$
$= n+4(n/4+4T(n/4^2))=2n+4^2T(n/4^2) =$
$= 2n+4^2 (n/4^2+4T(n/4^3)=3n+4^3T(n/4^3)=...$
$= jn+4^jT(n/4^j)=n \log 4n+n$

$(n/4^j)=1$ ➔ $j=\log_4 n$
$T(n/4^j)=1$
$4^j = n$

La complexitat és O(nlogn)

3.-(0,5 punts) Què és un problema NP? Si el problema de calcular la suma dels n primers nombres naturals el resolguéssim utilitzant un algorisme que te un cost n! , podríem dir que el problema de calcular la suma dels n primers nombres naturals és NP? Justifica la resposta.

Un problema pel qual actualment no es coneix una solució determinista en temps polinòmic, però si es pot verificar una solució en temps polinòmic.
El problema de calcular la suma dels n primers nombres naturals no és NP donat que tenim un algorisme que el calcula en temps constant.

4.- (0,5 punts) Quina diferència hi ha entre backtraking, branch and bound i greedy a l'hora de tractar els nodes vius?
En Backtraking els nodes vius els tenim a la branca que estem analitzant en cada moment, i podem trobar per sobre d'on ens trobem perquè encara no haguem obert tots els seus fills.
En greedy nomes tenim un node viu que es el que estem analitzant, en el moment que obrim els seus fills tots ho seran i ell morirà, però de seguida matarem tots els fills menys un, per això es diu que nomes tenim un node viu.
En Branch and Bound en canvi tenim una llista de nodes vius de la qual anem seleccionant el millor i generant els seus fills, ell es mor i els fills que siguin completables seran introduïts a la llista de nodes vius ordenats segons una heurística.

5.- (5 punts) Suposeu que heu de matricular-vos de **L crèdits** d'assignatures optatives i teniu 2 anys per fer-ho, però aquest any com a mínim us voldríeu matricular de **L/2 crèdits**, i com a màxim de **L** (no voleu matricular més crèdits dels que necessiteu). L'escola us ofereix **N assignatures** i per cada una d'elles sabeu quants crèdits té: **[C1,C2,...CN].** Per altre banda els vostres companys d'anys anteriors han dissenyat un llistat intern (i secret) de **l'esforç real** que requereix cada assignatura **[E1,E2,...EN].** Vosaltres voleu trobar una configuració de matrícula que **minimitzi l'esforç/crèdits** de les assignatures matriculades, sigui **com a mínim de L/2 crèdits** i **no passi de L crèdits**.

Volem resoldre el problema segons la tècnica de branch and bound. Per fer-ho volem que dissenyeu:

a) L'espai de cerca. Què indicaria cada nivell de l'arbre? Què calcularíeu per cada node?
b) Faríeu algun preprocés? Quin?
c) La funció de cota inferior per l'estimació mínima per cada node viu de Esforç/Crèdits.
d) La representació de cada estat.
e) Com faríeu la ramificació a cada pas de l'algorisme?

Suposeu el següent cas concret:
30 crèdits com a **màxim**, com a **mínim 15 crèdits**.
9 assignatures, numerades de l'1 al 9 que tenen els següents crèdits i esforços.
Esforç = [13, 2,10, 2, 6, 3, 4, 5, 1]
Crèdits= [5, 3, 5, 5, 5, 5, 5, 5, 2]
Voldríem fer com a poc 15 crèdits i com a molt 30.
La matrícula que minimitzaria esforç/crèdit seria: les assignatures 2,4,6 i 9.

a) Cada nivell indicaria si escollim o no una assignatura. Per cada node calcularíem la funció cota inferior.
b) Ordenaríem les assignatures segons Esforç/crèdit .
c) La idea de la cota inferior és relaxar el fet d'haver d'escollir assignatures senceres. La calculem com el promig de E/C de les assignatures escollides en una branca, + les que encara no hem escollit per ordre de E/C fins arribar a L/2 , o si la ultima assignatura escollida no arriba a L/2 crèdits i la següent es passa, agafaríem la part proporcional de la següent per no passar-nos mai de L/2 crèdits, donat que agafar més seria augmentar el E/C promig i ja no podríem assegurar que fos una cota inferior.
d) Per cada estat tindríem una llista de 0´s i 1`s indicant quines assignatures hem agafat i quines no, la suma E/C que portem fins ara, més el numero de crèdits que portem i el nivell en que ens trobem.
e) La ramificació es faria així: Escolliríem el millor node de la llista de nodes vius ordenada segons la funció de cota inferior. Generaríem els seus fills (agafar o no la següent assignatura) i el mataríem, miraríem si els fills són completables i si ho són els afegiríem a la llista de nodes vius ordenats segons la funció de cota inferior que es calcularia prèviament.