Exàmens de PRG - Problemes del Tema 2
Anàlisi d'algorismes. Eficiència. Ordenació

Curs 2018/19

P1 - Curs 18/19: 3 punts

Donada una matriu quadrada `a` de reals, el següent mètode comprova si la suma dels elements que estan per davall de la diagonal principal menys la dels que estan per damunt, coincideix amb la suma dels elements de la diagonal principal.

```
/** Precondició: a és una matriu quadrada. */
public static boolean sumBelowAbove(double[][] a) {
    double sumBelow = 0, sum = 0, sumAbove = 0;
    for (int i = 0; i < a.length; i++) {
        for (int j = 0; j < a.length; j++) {
            if (j < i) { sumBelow += a[i][j]; }
            else if (j == i) { sum += a[i][i]; }
            else { sumAbove += a[i][j]; }
        }
    }
    return sumBelow - sumAbove == sum;
}
```

Es demana:

a) (0.25 punts) Indica quina és la grandària o talla del problema, així com l'expressió que la representa.

b) (0.75 punts) Indica, i justifica, si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identifica-les si és el cas.

c) (1.50 punts) Tria una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella obtingues una expressió matemàtica, el més precisa possible, del cost temporal del mètode, distingint el cost de les instàncies més significatives en cas d'haver-les.

d) (0.50 punts) Expressa el resultat anterior utilitzant notació asimptòtica.

P1 - Curs 18/19: 3 punts

Una *matriu triangular superior* és una matriu quadrada els valors de la qual per davall de la diagonal principal són tots iguals a 0. El mètode recursiu `isUpperTriangular` següent comprova si les files de la matriu `m`, des de `0` fins a `nRow` compleixen aquesta propietat. Així, per a comprovar si certa matriu `mat` és triangular superior faríem la crida `isUpperTriangular(mat, mat.length - 1)`.

```
/** Precondició: m és una matriu quadrada d'enters, -1 <= nRow < m.length */
public static boolean isUpperTriangular(int[][] m, int nRow) {
    boolean res = true;
    if (nRow >= 0) {
        int j = 0;
        while (j < nRow && res) {
            if (m[nRow][j] != 0) { res = false; }
            else { j++; }
        }
        if (res) { res = isUpperTriangular(m, nRow - 1); }
    }
    return res;
}
```

Es demana:

a) (0.25 punts) Indica quina és la grandària o talla del problema, així com l'expressió que la representa.

b) (0.75 punts) Indica, i justifica, si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identifica-les si és el cas.

c) (1.50 punts) Escriu l'equació de recurrència del cost temporal en funció de la talla per a cadascun dels casos si hi haguera varis, o una única equació si només hi haguera un cas. Resol-la(es) per substitució.

d) (0.50 punts) Expressa la funció de cost utilitzant notació asimptòtica.

RecP1 - Curs 18/19: 3 punts

Una *matriu triangular superior* és una matriu quadrada, els valors de la qual per baix de la diagonal principal són 0. El mètode iteratiu `isUpperTriangular` comprova si la matriu `m` compleix aquesta propietat.

```
/** Precondició: m és una matriu quadrada d'enters */
public static boolean isUpperTriangular(int[][] m) {
    boolean res = true;
    int i = m.length - 1;
    while (i >= 0 && res) {
        int j = 0;
        while (j < i && m[i][j] == 0) { j++; }
        if (j < i) { res = false; }
        else { i--; }
    }
    return res;
}
```

Es demana:

a) (0.25 punts) Indica la talla del problema, i l'expressió que la representa.

b) (0.75 punts) Indica, i justifica, si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identifica-les si és el cas.

c) (1.50 punts) Tria una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella calcula una expressió matemàtica, el més precisa possible, del cost temporal del mètode, distingint el cost de les instàncies més significatives en cas d'haver-les.

d) (0.50 punts) Expressa el resultat anterior utilizant notació asimptòtica.

RecP1 - Curs 18/19: 3 punts

El següent mètode calcula recursivament la suma dels elements de la submatriu de grandària `n*n` amb origen en l'element `(0,0)`, d'una matriu quadrada `m`. Noteu que en el cas de voler sumar tots els elements de la matriu, la crida inicial seria `sumar(m, m.length)`.

```
/** Precondició: m és una matriu quadrada d'enters i 0 <= n <= m.length */
public static int sumar(int[][] m, int n) {
    if (n == 0) { return 0; }
    else {
        int s = m[n - 1][n - 1];
        for (int i = 0; i < n - 1; i++) {
            s = s + m[n - 1][i] + m[i][n - 1];
        }
        return s + sumar(m, n - 1);
    }
}
```

Es demana:

a) (0.25 punts) Indica quina és la talla del problema, i l'expressió que la representa.

b) (0.75 punts) Determina si existeixen instàncies significatives. Si n'hi ha, identifica les que representen els casos millor i pitjor de l'algorisme.

c) (1.50 punts) Escriu l'equació de recurrència del cost temporal en funció de la talla per a cadascun dels casos si n'hi hagueren, o una única equació si només hi haguera un cas. Ha de resoldre's per substitució.

d) (0.50 punts) Expressa el resultat anterior mitjançant notació asimptòtica.

Curs 2017/18

P1 - Curs 17/18: 3 punts

Donada una matriu quadrada `m` d'enters, els components de la qual són tots positius, el següent mètode comprova per a quines files la suma de tots els seus components és menor que `limit`. Per a cadascuna d'aquestes files, escriu el valor de la suma dels seus components.

```
/** Precondició: m és una matriu quadrada, i els seus elements
 *  són tots positius. El valor de limit és > 0. */
public static void sumes(int[][] m, int limit) {
    int n = m.length;
    for (int i = 0; i < n; i++) {
        int suma = m[i][0];
        int j = 1;
        while (j < n && suma < limit) {
            suma += m[i][j];
            j++;
        }
        if (suma < limit) {
            System.out.println("Fila " + i + ": " + suma);
        }
    }
}
```

Es demana:

a) (0.25 punts) Indica quina és la grandària o talla del problema, així com l'expressió que la representa.
b) (0.75 punts) Indica, i justifica, si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identifica-les si és el cas.
c) (1.50 punts) Tria una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella obtín una expressió matemàtica, el més precisa possible, del cost temporal del mètode, distingint el cost de les instàncies més significatives en cas d'haver-les.
d) (0.50 punts) Expressa el resultat anterior utilitzant notació asimptòtica.

P1 - Curs 17/18: 3 punts

Es desitja calcular el cost del següent mètode recursiu:

```
public static double testMethod(double[] v, int left, int right) {
    if (left > right) { return 1.0; }
    else {
        int middle = (left + right) / 2;
        return v[middle]
            * testMethod(v, left, middle - 1)
            * testMethod(v, middle + 1, right);
    }
}
```

Es demana:

a) (0.25 punts) Indica quina és la talla o grandària del problema, així com l'expressió que la representa.
b) (0.75 punts) Indica, i justifica, si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identifica-les si és el cas.
c) (1.50 punts) Escriu l'equació de recurrència del cost temporal en funció de la talla per a cadascun dels casos si hi ha més d'un, o una única equació si únicament hi haguera un cas. Ha de resoldre's per substitució.
d) (0.50 punts) Expressa el resultat anterior utilitzant notació asimptòtica.

RecP1 - Curs 17/18: 3 punts

Donada una matriu quadrada `m` de caràcters i un caràcter `c`, el següent mètode escriu les paraules o seqüències de caràcters que apareixen en cada fila, eliminant d'elles cada aparició de `c`.

```
/** Precondició: m és una matriu quadrada. */
public static void escriuSense(char[][] m, char c) {
    int dim = m.length;
    for (int i = 0; i < dim; i++) {
        for (int j = 0; j < dim; j++) {
            if (m[i][j] != c) { System.out.print(m[i][j]); }
        }
        System.out.println();
    }
}
```

Per exemple, si `m` = {{'e','e','l','e'}, {'m','e','m','e'}, {'n','u','l','l'}, {'c','a','s','e'}}, i `c` = 'e', aleshores el mètode escriu:

```
l
mm
null
cas
```

Es demana:

a) (0.25 punts) Indiqueu quina és la grandària o talla del problema, així com l'expressió que la representa.

b) (0.75 punts) Indiqueu, i justifiqueu, si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identifiqueu-les si és el cas.

c) (1.50 punts) Trieu una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella obteniu una expressió matemàtica, el més precisa possible, del cost temporal del mètode, distingint el cost de les instàncies més significatives en cas d'haver-les.

d) (0.50 punts) Expresseu el resultat anterior utilitzant notació asimptòtica.

RecP1 - Curs 17/18: 3 punts

Es desitja calcular el cost del següent mètode recursiu, que donats `x` > 1 i `d` $\leq$ `x`, comprova si `x` no té cap divisor propi en el rang [`d`, `x`[:

```
/** Precondició: x > 1 && 1 < d <= x */
public static boolean senseDivisors(int x, int d) {
    if (x == d) { return true; }
    else {
        if (x % d == 0) { return false; }
        else { return senseDivisors(x, d + 1); }
    }
}
```

Es demana:

a) (0.25 punts) Indiqueu quina és la talla o grandària del problema, així com l'expressió que la representa.

b) (0.75 punts) Indiqueu, i justifiqueu, si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identifiqueu-les si és el cas.

c) (1.50 punts) Escriviu l'equació de recurrència del cost temporal en funció de la talla per a cadascun dels casos si hi ha més d'un, o una única equació si solament hi haguera un cas. Ha de resoldre's per substitució.

d) (0.25 punts) Expresseu el resultat anterior utilitzant notació asimptòtica.

e) (0.25 punts) Quin seria el cost asimptòtic en funció del valor de `x` de la crida `senseDivisors(x, 2)`, és a dir, d'averiguar si `x` és primer?

Curs 2016/17

P1 - Curs 16/17: 3 punts

Donat un array de caràcters `a` i un caràcter `c` qualsevol, el següent mètode escriu en l'eixida estàndard, línia a línia, tots els prefixes de la seqüència de caràcters en `a`, de longitud `1` en endavant, que no acaben en el caràcter `c`.

```
public static void prefixes(char[] a, char c) {
    for (int i = 0; i < a.length; i++) {
        if (a[i] != c) {
            for (int j = 0; j <= i; j++) {
                System.out.print(a[j]);
            }
            System.out.println();
        }
    }
}
```

Per exemple, si `a = {'g', 't', 'a', 't', 'c'}`, els prefixes de longituds succesives són `g`, `gt`, `gta`, `gtat` i `gtatc`. Per a `a` i `c = 't'`, el mètode escriu:

```
g
gta
gtatc
```

Es demana:

a) (0.25 punts) Indicar quina és la grandària o talla del problema, així com l'expressió que la representa.

b) (0.75 punts) Indicar si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identificar-les si és el cas.

c) (1.50 punts) Triar una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella obtenir una expressió matemàtica, el més precisa possible, del cost temporal del mètode, distingint el cost de les instàncies més significatives en cas d'haver-les.

d) (0.50 punts) Expressar el resultat anterior utilitzant notació asimptòtica.

P1 - Curs 16/17: 3 punts

El següent mètode determina si, donat un nombre enter no negatiu `num`, el seu literal pot estar expressat en una base determinada `b` ($2 \leq$ `b` ≤ 10), comprovant que tots els dígits del nombre tenen un valor estrictament menor que la base `b`. Per exemple, el nombre 453123 pot representar un valor en base 6, 7, 8, 9 i 10 ja que tots els seus dígits són estrictament inferiors als valors d'aquestes possibles bases.

```
/** Precondició: 2 <= b <= 10 i num >= 0 */
public static boolean basePossible(int num, int b) {
    if (num == 0) { return true; }
    else {
        int ultDig = num % 10;
        if (ultDig < b) { return basePossible(num / 10, b); }
        else { return false; }
    }
}
```

Es demana:

a) (0.25 punts) Indicar quina és la grandària o talla del problema, així com l'expressió que la representa.

b) (0.75 punts) Indicar si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identificar-les si és el cas.

c) (1.50 punts) Escriure l'equació de recurrència del cost temporal en funció de la talla per a cada un dels casos si n'hi ha diversos, o una única equació si només hi hagués un cas. Cal resoldre-la per substitució.

d) (0.50 punts) Expressar el resultat anterior utilitzant notació asimptòtica.

RecP1 - Curs 16/17: 4 punts

El següent mètode, donat un array `a` ordenat ascendentment, torna el número de vegades que apareix a l'array el primer número repetit (el de valor més menut) o 0 si no hi ha repetits a l'array. Per exemple, si `a = {2, 5, 5, 5, 8, 8, 9}` torna `3` (el primer número que apareix repetit és el `5` i es repeteix `3` vegades); si `a = {2, 5, 8, 9}` torna `0`.

```
/** Precondició: a ordenat ascendentment */
public static int comptarPrimerRepetit(int[] a) {
    int compt = 0, i = 0;
    boolean repe = false;
    while (i < a.length - 1 && !repe) {
        int j = i + 1;
        repe = (a[j] == a[i]);
        while (j < a.length && a[j] == a[i]) { j++; }
        if (repe) { compt = j - i; }
        i++;
    }
    return compt;
}
```

Es demana:

a) (0.25 punts) Indicar quina és la grandària o talla del problema, així com l'expressió que la representa.

b) (0.75 punts) Indicar si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identificar-les si és el cas.

c) (1.50 punts) Triar una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella obtenir una expressió matemàtica, el més precisa possible, del cost temporal del mètode, distingint el cost de les instàncies més significatives en cas d'haver-les.

d) (0.50 punts) Expressar el resultat anterior utilitzant notació asimptòtica.

e) (1 punt) Tenint en compte que l'algorisme d'ordenació per *inserció directa* vist en classe té un cost lineal en el millor cas i quadràtic en el pitjor, quin és el cost d'ordenar `a` per inserció directa i després aplicar el mètode `comptarPrimerRepetit(int[])`?

RecP1 - Curs 16/17: 3 punts

El següent mètode calcula la suma dels bits de la representació en binari d'un enter no negatiu `num`.

```
/** Precondició: num >= 0 */
public static int sumaBits(int num) {
    if (num <= 1) { return num; }
    else {
        int ultBit = num % 2;
        return sumaBits(num / 2) + ultBit;
    }
}
```

Es demana:

a) (0.25 punts) Indicar quina és la grandària o talla del problema, així com l'expressió que la representa.

b) (0.5 punts) Indicar si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identificar-les si és el cas.

c) (1.5 punts) Escriure l'equació de recurrència del cost temporal en funció de la talla per a cada un dels casos si n'hi ha diversos, o una única equació si només hi hagués un cas. Cal resoldre-la per substitució.

d) (0.75 punts) Expressar el resultat anterior utilitzant notació asimptòtica.

Curs 2015/16

P1 - Curs 15/16: 3 punts

El següent mètode comprova si un array d'enters es troba ordenat de manera ascendent entre les posicions `ini` i `fi` inclusivament:

```
/** Torna true si a[ini..fi] està ordenat ascendentment,
 *  false en cas contrari. */
public static boolean ordenat(int[] a, int ini, int fi) {
    if (ini >= fi) { return true; }
    else {
        if (a[ini] > a[ini + 1] || a[fi] < a[fi - 1]) { return false; }
        else { return ordenat(a, ini + 1, fi - 1); }
    }
}
```

Es demana:

a) (0.25 punts) Indiqueu quina és la grandària o talla del problema, així com l'expressió que la representa.

b) (0.5 punts) Indiqueu si existeixen diferents instàncies significatives per al cost temporal de l'algorisme, identificant-les si és el cas.

c) (1.5 punts) Doneu la relació de recurrència per al cost, resolent-la per substitució, distingint el cost de les instàncies més significatives en cas d'haver-les.

d) (0.75 punts) Expresseu el resultat anterior utilitzant notació asimptòtica.

P1 - Curs 15/16: 3 punts

El següent mètode transposa una matriu quadrada:

```
/** Canvia la matriu m a la seua transposada.
 *  Precondició: m es una matriu quadrada. */
public static void transposada(int[][] m) {
    for (int i = 0; i < m.length; i++) {
        for (int j = 0; j < i; j++) {
            int aux = m[i][j];
            m[i][j] = m[j][i];
            m[j][i] = aux;
        }
    }
}
```

Es demana:

a) (0.25 punts) Indiqueu quina és la grandària o talla del problema, així com l'expressió que la representa.

b) (0.5 punts) Indiqueu si existeixen diferents instàncies significatives per al cost temporal de l'algorisme, identificant-les si és el cas.

c) (1.5 punts) Escolliu una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella obtenir una expressió matemàtica, el més precisa possible, del cost temporal del mètode, distingint el cost de les instàncies més significatives en cas d'haver-les .

d) (0.75 punts) Expresseu el resultat anterior utilitzant notació asimptòtica.

RecP1 - Curs 15/16: 3 punts

Mitjançant el següent mètode es comprova si totes les files de la matriu `m`, quadrada, de `double`, sumen sempre un mateix valor. Se sap que `m` és almenys d'ordre 2 (té un nombre de files i columnes major o igual que 2).

```
/** Precondició: m és quadrada d'ordre major o igual que 2. */
public static boolean sumenIgual(double[][] m) {
    // Es calcula la suma de la primera fila:
    int sum0 = 0;
    for (int i = 0; i < m.length; i++) { sum0 += m[0][i]; }

    boolean sumIgual = true;
    // Es calcula la suma de cada fila posterior:
    for (int i = 1; i < m.length && sumIgual; i++) {
        int sumFil = 0;
        for (int j = 0; j < m.length; j++) { sumFil += m[i][j]; }
        sumIgual = (sum0 == sumFil);
    }
    return sumIgual;
}
```

Es demana:

a) (0.25 punts) Indicar quina és la grandària o talla del problema, així com l'expressió que la representa.

b) (0.75 punts) Indicar si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identificar-les si és el cas.

c) (1.50 punts) Triar una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella obtenir una expressió matemàtica, el més precisa possible, del cost temporal del mètode, distingint el cost de les instàncies més significatives en cas d'haver-les.

d) (0.50 punts) Expressar el resultat anterior utilitzant notació asimptòtica.

RecP1 - Curs 15/16: 3 punts

Donada la següent implementació d'un algorisme recursiu que resol la multiplicació *a la russa* de dos nombres naturals `a` i `b`:

```
/** Precondició: a >=0 i b >= 0. */
public static int producteRus(int a, int b) {
    if (b == 0) { return 0; }
    else {
        if (b % 2 == 0) { return producteRus(a * 2, b / 2); }
        else { return a + producteRus(a * 2, b / 2); }
    }
}
```

Es demana: analitzar el cost temporal d'aquest algorisme. En concret:

a) (0.25 punts) Indicar quina és la talla del problema, i quina expressió la defineix.

b) (0.5 punts) Identificar, cas que les hi haguera, les instàncies del problema que representen el cas millor i pitjor de l'algorisme.

c) (1.5 punts) Triar una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella obtenir una expressió matemàtica, el més precisa possible, de l'equació de recurrència del cost temporal en funció de la talla (una única equació si no hi haguera instàncies significatives; dues equacions, corresponents als casos millor i pitjor, en el cas que el problema tinguera instàncies significatives). Resoldre-la(les) per substitució.

d) (0.75 punts) Expressar el resultat anterior utilitzant notació asimptòtica.

Curs 2014/15

P1 - Curs 14/15: 7 punts

Per tal de determinar quants elements d'un array `a` són menors que un valor donat `x`, es proposen les dues solucions següents en Java, on la primera d'elles suposa que l'array està ordenat ascendentment.

- Solució 1

```
/** Torna el nombre d'elements de l'array a menors que x
 *  Precondició: a està ordenat ascendentment.  */
public static int comptarMenorsX1(int[] a, int x) {
    int i = a.length - 1;
    while (i >= 0 && a[i] >= x) { i--; }
    return i + 1;
}
```

- Solució 2

```
/** Torna el nombre d'elements de l'array a menors que x
 *  Precondició: 0 <= pos <= a.length.
 *  Crida inicial: int res = comptarMenorsX2(a, x, 0); */
public static int comptarMenorsX2(int[] a, int x, int pos) {
    if (pos == a.length) { return 0; }
    else if (a[pos] < x) { return 1 + comptarMenorsX2(a, x, pos + 1); }
    else { return comptarMenorsX2(a, x, pos + 1); }
}
```

Es demana:

i. (3 punts/proposta) Per a cada solució proposada:
   a) (*0.25 punts*) Indicar quina és la grandària o talla del problema, així com l'expressió que la representa.
   b) (*0.5 punts*) Indicar si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identificar-les si és el cas.
   c) (*1.5 punts*) En el cas del mètode iteratiu, triar una unitat de mesura per a l'estimació del cost (pas de programa, instrucció crítica) i d'acord amb ella, obtindre una expressió matemàtica, el més precisa possible, del cost temporal, a nivell global o en les instàncies més significatives si n'hi ha.
   En el cas del mètode recursiu, escriure l'equació de recurrència del cost temporal en funció de la talla per a cada un dels casos si n'hi ha diversos, o una única equació si només hi hagués un cas. Cal resoldre-la per substitució.
   d) (*0.75 punts*) Expressar el resultat anterior utilitzant notació asimptòtica.

ii. (1 punt) Tenint en compte que l'algorisme d'ordenació per *inserció directa* vist en classe té un cost lineal en el millor cas i quadràtic en el pitjor, i que per a la primera solució caldria ordenar l'array prèviament, comparar el cost temporal total de comptar els valors menors que `x` en un array `a` no necessariament ordenat, quan es resol el problema mijançant els següents algorismes:
   - Algorisme 1: Primer ordenar `a` per inserció directa i després aplicar el mètode `comptarMenorsX1(int[], int)`.
   - Algorisme 2: Aplicar directament a l'array `a` el mètode `comptarMenorsX2(int[], int, int)`.

Curs 2013/14

P1 - Curs 13/14: 3 punts

Considerar el següent mètode recursiu en Java que comprova si tots els elements del subarray `a[pos..a.length-1]` apareixen formant una progressió aritmètica de diferència `d`:

```
/** Retorna true si per a tota parella a[i], a[i + 1] en a[pos..a.length - 1]
 *  es compleix que a[i + 1] = a[i] + d, i false en cas contrari.
 *  Precondició: a.length >= 1 && 0 <= pos <= a.length - 1. */
public static boolean progAritmetica(int[] a, int d, int pos) {
    if (pos == a.length - 1) { return true; }
    else { return a[pos + 1] == a[pos] + d  && progAritmetica(a, d, pos + 1); }
}
```

Es demana:

a) (0.25 punts) Indicar quina és la talla del problema i quina expressió la defineix.

b) (0.5 punts) Identificar, cas de que les haguera, les instàncies del problema que representen el cas millor i pitjor de l'algorisme.

c) (1.5 punts) Escriure l'equació de recurrència del cost temporal en funció de la talla per a cadascun dels casos si n'hi ha diversos, o una única equació si només hi hagués un cas. Resoldre-la(les) per substitució.

d) (0.75 punts) Expressar el resultat anterior utilitzant notació asimptòtica.

P1 - Curs 13/14: 4 punts

El següent mètode, `triangle(int)`, determina, escrivint-los, el número de triangles rectangles de costats enters i hipotenusa `h`:

```
/** El mètode compta, escrivint-los, tots els triangles
 *  rectangles de costats enters i hipotenusa h. */
public static int triangle(int h) {
    int cont = 0;
    for (int c1 = 4; c1 < h; c1++) {
        for (int c2 = 3; c2 < c1; c2++) {
            if (c1 * c1 + c2 * c2 == h * h) {
                cont++;
                System.out.println("c1= " + c1 + ", c2= " + c2 + ", h= " + h);
           }
        }
    }
    return cont;
}
```

Es demana:

a) (0.5 punts) Indicar quina és la grandària o talla del problema, així com l'expressió que la representa.

b) (0.5 punts) Indicar si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identificar-les si és el cas.

c) (2 punts) Triar una unitat de mesura per al cost (passos de programa, instrucció crítica) i d'acord amb ella, obtenir una expressió el més precisa possible del cost temporal del programa, a nivell global o en les instàncies més significatives si és el cas.

d) (1 punt) Expressar el resultat anterior en notació asimptòtica.

RecP1 - Curs 13/14: 3 punts

El següent mètode recursiu, `inversio(String)`, obté una `String` amb la inversió dels caràcters de la que rep com a argument. Per exemple, la inversió de la cadena `hola` és `aloh`.

```
public static String inversio(String s) {
    if (s.length() <= 1) { return s; }
    else { return inversio(s.substring(1)) + s.charAt(0); }
}
```

Es vol estudiar el seu cost temporal en les dues situacions següents:

1. Suposar que tant l'operació `substring(int)` com l'operació de concatenació (operador `+`) tenen un **cost constant** amb la llargària de la String `s`.
2. Suposar que l'operació `substring(int)` té un **cost lineal** amb la llargària de la String `s`, mentre que l'operació de concatenació (operador `+`) té un **cost constant** amb el nombre total de caràcters que es concatenen.

Per a cadascuna de les dues situacions **es demana**:

(a) (0.25 punts) Indicar quina és la mida o talla del problema, així com l'expressió que la representa.

(b) (0.5 punts) Indicar si existeixen diferents instàncies significatives per al cost temporal de l'algorisme, i identificar-les si és el cas.

(c) (1.5 punts) Escriure les equacions de recurrència del cost temporal en funció de la talla, resolent-les per substitució.

(d) (0.5 punts) Expressar el resultat anterior en notació asimptòtica.

(e) (0.25 punts) Quines de les dues situacions creus que és la més favorable des d'un punt de vista de cost temporal? Justifica la teva resposta.

RecP1 - Curs 13/14: 4 punts

El següent algorisme iteratiu retorna un enter corresponent a la suma màxima dels valors emmagatzemats en posicions consecutives d'un array donat `a`.

```
/** Precondició: a.length >= 1. */
public static int metode(int[] a) {
    int n = a.length, max = a[0];
    for (int i = 0; i <= n - 1; i++) {
        int suma = 0;
        for (int j = i; j <= n - 1; j++) {
            suma = suma + a[j];
            if (suma > max) { max = suma; }
        }
    }
    return max;
}
```

Es demana:

a) (0.5 punts) Indicar quina és la mida o talla del problema, així com l'expressió que la representa.

b) (0.5 punts) Indicar si existeixen diferents instàncies significatives per al cost temporal de l'algorisme, i identificar-les si és el cas.

c) (2 punts) Escollir una unitat de mesura per al cost (passos de programa, instrucció crítica) i d'acord amb ella, obtenir una expressió el més precisa possible del cost temporal del programa (per al cas millor i el cas pitjor si és el cas).

d) (1 punt) Expressar el resultat anterior en notació asimptòtica.

Curs 2012/13

P1 - Curs 12/13: 3 punts

Donada certa matriu `m` quadrada i un array `a`, d'enters, els dos amb la mateixa dimensió (això és, `m.length == a.length`), el següent mètode **iteratiu** torna la posició (número de fila) on es troba l'array `a` en `m`, cas de que es trobe, o torna -1 si no fora així:

```
/** Torna la posició en que l'array a es troba com a fila
 *  dins de la matriu quadrada m, o -1 si no es troba.
 *  Precondició: m es quadrada i m.length == a.length. */
public static int cercaPosFila(int[][] m, int[] a) {
    boolean trobat = false;
    int i = 0;
    while (i < m.length && !trobat) {
        trobat = true;
        for (int j = 0; j < m.length && trobat; j++) {
            trobat = (m[i][j] == a[j]);
        }
        if (!trobat) { i++; }
    }
    if (trobat) { return i; }
    else { return -1; }
}
```

Es demana l'estudi del seu cost temporal:

a) Indica quina és la grandària o talla del problema, així com l'expressió que la representa.

b) Identifica, en cas que n'hi hagués, les instàncies del problema que representen el cas millor i pitjor de l'algorisme.

c) Tria una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella, obté una expressió matemàtica, el més precisa possible, del cost temporal del programa, a nivell global o en les instàncies més significatives si les hi ha.

d) Expressa el resultat anterior utilitzant notació asimptòtica.

P1 - Curs 12/13: 2 punts

El següent mètode comprova si cert subarray `a[i..f]` d'`int` està o no ordenat ascendentment subdividint-lo prèviament i comprobant que cadascuna de les seues dues parts ho estiga, així com la relació entre elles:

```
/** Torna cert sii a[i..f] està ordenat ascendentment.
 *  Precondició: i <= f. */
public static boolean esOrdenat(int[] a, int i, int f) {
    if (i == f) { return true; }
    else {
        int m = (i + f) / 2;
        return esOrdenat(a, i, m) && esOrdenat(a, m + 1, f) && a[m] <= a[m + 1];
   }
}
```

Es demana l'estudi del seu cost temporal:

a) Indica quina és la talla del problema i quina expressió la defineix.

b) Determina si existeixen instàncies significatives. Si n'hi ha, identifica les que representen els casos millor i pitjor de l'algorisme.

c) Escriu l'equació de recurrència del cost temporal en funció de la talla per a cada un dels casos si n'hi ha diversos, o una única equació si només hi hagués un cas. Cal resoldre-la per substitució.

d) Expressa el resultat anterior fent servir notació asimptòtica.

P1 - Curs 12/13: 1 punt

És possible modificar una única instrucció de l'algorisme anterior per a que el seu cost temporal siga $\Omega(1)$. Quina instrucció s'hauria de modificar i de quina forma?

RecP1 - Curs 12/13: 3.0 punts

Donada certa matriu `m` quadrada de valors reals, el següent mètode determina si aquesta és triangular inferior (és a dir, si tots els elements superiors a la diagonal principal són iguals a 0):

```
/** Determina si una matriu quadrada és triangular inferior, això és:
  * si tots els elements superiors a la diagonal principal valen 0. * /
public static boolean esInferior(double[][] m) {
    boolean esInf = true;
    for (int i = 0; i < m.length && esInf; i++) {
        for (int j = i + 1; j < m.length && esInf; j++) {
            esInf = m[i][j] == 0;
        }
    }
    return esInf;
}
```

Es demana: estudiar el cost temporal del mètode, per el que has de:

a) Indicar quina és la mida o talla del problema, així com l'expressió que la representa.

b) Identificar, cas que n'hi hagués, les instàncies del problema que representen el cas millor i pitjor de l'algorisme.

c) Triar una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i acord amb ella, obtenir una expressió matemàtica, el més precisa possible, del cost temporal del programa, a nivell global o en les instàncies més significatives si n'hi ha.

d) Expressar el resultat anterior utilitzant notació asimptòtica.

RecP1 - Curs 12/13: 2.5 punts

El següent mètode recursiu comprova si dos `String` que tenen la mateixa longitud són simètriques. Dues `String` són simètriques quan el primer element de la primera és igual a l'últim de la segona i així successivament.

Per exemple, les `String`: `"HOLA"` i `"ALOH"` són simètriques, mentre que `"HOLA"` i `" ALHA"` no ho són.

```
/** Determina si a i b són o no simètriques
 *  Precondició: a.length() == b.length(). */
public static boolean simetriques(String a, String b) {
    if (a.length() == 0) { return true; }
    else {
        int ult = b.length() - 1;
        return a.charAt(0) == b.charAt(ult)
            && simetriques(a.substring(1), b.substring(0, ult));
    }
}
```

Es demana: estudiar el cost temporal del mètode anterior, sabent que **totes les operacions de la classe** `String` que s'apliquen a alguna `String` en l'algorisme **tenen un cost constant** (això és, el seu cost no depèn de la llargària de la `String` a la qual s'aplique) per a això:

a) Indica quina és la talla del problema i quina expressió la defineix.

b) Determina si hi ha instàncies significatives. Si n'hi ha, identifica les que representen els casos millor i pitjor de l'algorisme.

c) Escriu l'equació de recurrència del cost temporal en funció de la talla per a cada un dels casos si n'hi ha diversos, o una única equació si només hi hagués un cas. Resol-la per substitució.

d) Expressa el resultat anterior usant notació asimptòtica.

Curs 2011/12

P1 - Curs 11/12: 2 punts

Donat el següent mètode:

```
/** Precondició: n >= 0. */
public static void binari(int n) {
    if (n > 0) { binari(n / 2); }
    System.out.print(n % 2);
}
```

Es demana:

a) Indica quina és la talla del problema i quina expressió la defineix.

b) Determina si existeixen instàncies significatives. Si n'hi ha, identifica les que representen els casos millor i pitjor de l'algorisme.

c) Escriu l'equació de recurrència del cost temporal en funció de la talla per a cada un dels casos si n'hi ha diversos, o una única equació si només hi hagués un cas. Cal resoldre-la per substitució.

d) Expressa el resultat anterior fent servir notació asimptòtica.

P1 - Curs 11/12: 3 punts

Siga una matriu quadrada d'enters, `m`, amb tots els seus valors no negatius. Per tal de comprovar si els elements de la seua diagonal principal sumen més que cert valor `val`, no negatiu, es consideren els dos mètodes següents:

```
public class Problema4 {
    /** Per a tot i, j: 0 <= i < m.length, 0 <= j < m.length, m[i][j] >= 0 i val >= 0 */
    public static boolean metode1(int[][] m, int val) {
        int s = 0;
        for (int i = 0; i < m.length; i++) { s += m[i][i]; }
        return s > val;
    }

    /** Per a tot i, j: 0 <= i < m.length, 0 <= j < m.length, m[i][j] >= 0 i val >= 0 */
    public static boolean metode2(int[][] m, int val) {
        int s = 0;
        for (int i = 0; i < m.length && s <= val; i++) {
            for (int j = 0; j <= i && s <= val; j++) {
                if (i == j) { s += m[i][j]; }
            }
        }
        return s > val;
    }
}
```

Es demana:

a) **Per a cada mètode:**

1. Indica quina és la grandària o talla del problema, així com l'expressió que la representa.
2. Identifica, en cas que n'hi hagués, les instàncies del problema que representen el cas millor i pitjor de l'algorisme.
3. Tria una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella, obté una expressió matemàtica, el més precisa possible, del cost temporal del programa, a nivell global o en les instàncies més significatives si les hi ha.
4. Expressa el resultat anterior utilitzant notació asimptòtica.

b) Descriu breument les diferències entre ambdós mètodes.

c) Com modificaries el primer per tal de millorar-lo?

P1 - Curs 11/12: 1 punt

La taula següent mostra els temps d'execució, en mil·lisegons, de cert algorisme per als valors de la talla del problema que es mostren en la primera columna.

```
#    Talla    Temps (ms)
#---------------------------------------
      1000       49.78
      2000      202.33
      3000      454.42
      4000      804.03
      5000     1270.28
      6000     1841.47
      7000     2506.30
      8000     3253.62
      9000     4141.05
     10000     5277.99
```

a) Des d'un punt de vista asimptòtic, quina creus que és la funció de cost temporal que millor aproxima els valors de la columna `Temps`?

b) De forma aproximada, quant de temps creus que tardaria l'algorisme anterior en executar-se per a una talla de 20000 elements?

RecP1 - Curs 11/12: 2.5 punts

Donat el següent mètode:

```
/** Precondició: n >= 0, 1 <= x <= 9. */
public static boolean cercarX(int n, int x){
    if (n > 0) {
        if (n % 10 == x) { return true; }
        else { return cercarX(n / 10, x); }
    }
    else { return false; }
}
```

Es demana:

a) Indicar quina és la talla del problema i quina expressió la defineix.

b) Determinar si existeixen instàncies significatives. Si n'hi ha, identificar les que representen els casos millor i pitjor de l'algorisme.

c) Escriure l'equació de recurrència del cost temporal en funció de la talla per a cada un dels casos si n'hi ha diversos, o una única equació si només hi hagués un cas. Cal resoldre-la per substitució.

d) Expressar el resultat anterior fent servir notació asimptòtica.

RecP1 - Curs 11/12: 3.5 punts

Siga `a` un array $\{a_0, a_1, a_2, \ldots, a_{n-1}\}$ de `double`, que representa els coeficients d'un polinomi $a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$. Per tal de calcular el valor del polinomi per a un x donat, es proposen els següents mètodes:

Mètode 1:

```
/** Precondició: a.length >= 1. */
public static double polinomi1(double[] a, double x) {
    double result = a[0];
    for (int i = 1; i < a.length; i++) {
        double pot = 1;
        for (int k = 1; k <= i; k++) { pot = pot * x; }
        result += a[i] * pot;
    }
    return result;
}
```

Mètode 2:

```
/** Precondició: a.length >= 1. */
public static double polinomi2(double[] a, double x) {
    double result = a[a.length - 1];
    for (int i = a.length - 2; i >= 0; i--) {
        result = result * x + a[i];
    }
    return result;
}
```

Es demana:

a) **Per a cada mètode:**

1. Indicar quina és la grandària o talla del problema, així com l'expressió que la representa.
2. Identificar, en cas que n'hi hagués, les instàncies del problema que representen el cas millor i pitjor de l'algorisme.
3. Triar una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella, obtindre una expressió matemàtica, el més precisa possible, del cost temporal del programa, a nivell global o en les instàncies més significatives si les hi ha.
4. Expressar el resultat anterior utilitzant notació asimptòtica.

b) Indicar quin dels dos mètodes és més eficient i perquè.

Curs 2010/11

P1 - Curs 10/11: 3 punts

El següent mètode **iteratiu** calcula el producte d'un vector fila `x` per una matriu quadrada `a`, obtenint com a resultat un altre vector fila. Per tractar-se d'una matriu quadrada tant el vector `x` com el vector resultat tindran la mateixa dimensió, és a dir, el nombre de files de la matriu.

```
/** Precondició: x.length = a.length i a és una matriu quadrada. */
public static double[] vectorPerMatriu(double[] x, double[][] a) {
    double[] r = new double[a.length];
    for (int i = 0; i < r.length; i++) {
        r[i] = 0.0;
        for (int j = 0; j < x.length; j++) {
            r[i] += x[j] * a[j][i];
        }
    }
    return r;
}
```

Per exemple, per `x` $= (2, 1, 3)$, un vector fila de dimensió 1×3, i `a` $= \begin{pmatrix} 4 & 3 & 2 \\ 2 & 5 & 1 \\ 1 & 0 & 3 \end{pmatrix}$, una matriu de dimensió 3×3, el producte `x` $\times$ `a` és `r` $= (13, 11, 14)$, un vector fila de dimensió 1×3.

Es demana:

a) Indicar quina és la grandària o talla del problema, així com l'expressió que la representa.

b) Identificar, en cas que n'hi hagués, les instàncies del problema que representen el cas millor i pitjor de l'algorisme.

c) Triar una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella, obtenir una expressió matemàtica, el més precisa possible, del cost temporal del programa, a nivell global o en les instàncies més significatives si les hi ha.

d) Expressar el resultat anterior utilitzant notació asimptòtica.

P1 - Curs 10/11: 3.5 punts

Per obtenir la posició de l'últim element senar d'un array `v` d'enters, es proposa el següent mètode **iteratiu**:

```
public static int posUltimSenar(int[] v) {
    int i = v.length - 1;
    while (i >= 0 && v[i] % 2 == 0) { i--; }
    return i;
}
```

Es demana:

a) Indicar quina és la grandària o talla del problema, així com l'expressió que la representa.

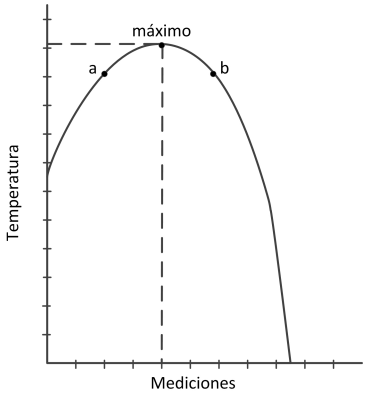
b) Identificar, en cas que n'hi hagués, les instàncies del problema que representen el cas millor i pitjor de l'algorisme.

c) Triar una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella, obtenir una expressió matemàtica, el més precisa possible, del cost temporal del programa, a nivell global o en les instàncies més significatives si les hi ha.

d) Expressar el resultat anterior utilitzant notació asimptòtica.

P1 - Curs 10/11: 3.5 punts

Es disposa d'una sèrie de mesures de temperatura que formen part d'una corba parabòlica (positiva) en un array `v` de reals. El següent mètode **recursiu** obté el màxim valor de la corba representada per l'array `v`.



Per a aquest tipus de corbes, donat un valor se sap que si l'anterior i el posterior són menors que ell, ens trobem en el punt màxim mesurat. Si per contra l'anterior és més gran i el posterior és menor, estem a la dreta del màxim, i si l'anterior és menor i el posterior gran, estem a l'esquerra del màxim. Un altra situació és impossible en aquest tipus de corba. A la corba de la figura, es pot observar que els punts `a` i `b` (anterior i posterior al punt màxim, respectivament) són menors que el punt `máximo`.

```
public static double maximaTemp(double[] v, int ini, int fi) {
    if (ini == fi) { return v[ini]; }
    else if (ini == fi - 1) { return Math.max(v[ini], v[fi]); }
    else {
        int meitat = (fi + ini) / 2;
        if (v[meitat] > v[meitat - 1] && v[meitat] > v[meitat + 1]) {
            return v[meitat];
        }
        else if (v[meitat] < v[meitat - 1] && v[meitat] > v[meitat + 1]) {
            return maximaTemp(v, ini, meitat);
        }
        else { return maximaTemp(v, meitat, fi); }
    }
}
```

La crida inicial serà: `double maxim = maximaTemp(v, 0, v.length - 1);`

Es demana:

a) Indicar quina és la talla del problema i quina expressió la defineix.

b) Determinar si hi ha instàncies significatives. Si n'hi ha, identificar les que representen els casos millor i pitjor de l'algorisme.

c) Escriure l'equació de recurrència del cost temporal en funció de la talla per a cada un dels casos si n'hi ha diversos, o una única equació si només hi hagués un cas. Resoldre per substitució.

d) Expressar el resultat anterior fent servir notació asimptòtica.

RecP1 - Curs 10/11: 5 punts

Donat el següent mètode **recursiu** on la seua invocació inicial deu ser: `palindrom(s, 0, s.length() - 1)`:

```
public static boolean palindrom(String s, int ini, int fi) {
    if (ini >= fi) { return true; }
    if (s.charAt(ini) != s.chartAt(fi)) { return false; }
    return palindrom(s, ini + 1, fi - 1);
}
```

Es demana:

a) Indicar quin és el tamany o talla del problema, així com l'expressió que el representa.

b) Identificar, en cas de que les hi haja, les instàncies del problema que representen els casos millor i pitjor de l'algorisme.

c) Triar una unitat de mesura per estimar el cost (pas de programa o instrucció crítica), i d'acord amb la mesura escollida obtenir l'expressió matemàtica, el més precisa possible, del cost temporal del programa. A nivell general si no existeixen instàncies significatives, o per als casos millor i pitjor en cas de que sí en existeixen.

d) Expressar el resultat anterior fet ús de la notació asimptòtica.