Donat un array **`a`** d'**`int`** i un enter **`m`**, escriu un mètode **recursiu** que comprove si existeix una parella **`a[i]`** i **`a[f]`**, **`0 ≤ i ≤ f < a.length`**, de components *simètriques* (la distància d'**`i`** a **`0`** és la mateixa que la distància de **`f`** a **`a.length – 1`**) que sumen **`m`**. Si existeix la parella, ha de tornar l'índex en el que es troba (l'índex **`i`**, el més baix de la parella), **`-1`** en cas contrari.
Per exemple, per a **`m = 4`** i **`a = {1, 4, 5, 9, 6, 0, -8}`** el mètode en la crida inicial que s'extenga sobre tot l'array ha de tornar **`1`**, per a **`m = 18`** i **`a = {1, 4, 5, 9, 6, 0, 2}`** ha de tornar **`3`**, per a **`m = 25`** i **`a = {1, 3, 2, 5, 4, 6}`** ha de tornar **`-1`**.

Es demana:

a) Perfil del mètode, amb els paràmetres adequats per tal de resoldre recursivament el problema, i precondició relativa als paràmetres.
b) Cas base i cas general de la recursió.
c) Implementació en Java del mètode recursiu.
d) Crida inicial al mètode recursiu per a comprovar si existeix una parella de components simètriques en un array **`dades`** que sumen **`s`**.

a) Una possible solució consisteix a definir un mètode com:

```
/** Precondició: 0 <= ini, fi < a.length. */
public static int parellaSim(int[] a, int ini, int fi, int m)
```

de manera que, éssent **0 ≤ ini**, **fi < a.length**, cerca una parella d'elements simètrics que sumen **m** al subarray **a[ini..fi]**.

b)

- Cas base, **ini > fi**: No es troba la parella buscada, s'ha de retornar **-1**.
- Cas general, **ini ≤ fi**: Si **a[ini] + a[fi]** val **m**, s'ha de retornar **ini**, sinó la cerca es redueix a **a[ini+1..fi-1]**.

c)

```
/** Busca l'índex de la primera parella d'elements simètrics en a[ini..fi],
 *  que sumen m. Si no existeix, retorna -1.
 *  Precondicio: 0 <= ini, fi < a.length. */
public static int parellaSim(int[] a, int ini, int fi, int m) {
    if (ini > fi) { return -1; }
    else if (a[ini] + a[fi] == m) { return ini; }
    else { return parellaSim(a, ini + 1, fi - 1, m); }
}
```

d) Per a un array **dades** i un enter **s**, la crida **parellaSim(dades, 0, dades.length - 1, s)** resol el problema de l'enunciat.

El següent mètode **iteratiu** transposa una matriu quadrada:

```
/** Precondició: m és 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) Indica quina és la talla o grandària del problema, així com l'expressió que la representa.
b) Indica, i justifica, si existeixen diferents instàncies significatives per al cost temporal de l'algorisme i identifica-les si és el cas.
c) Escull 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 mètode, distingint el cost de les instàncies més significatives en cas d'haver-les.
d) Expressa el resultat anterior utilitzant notació asimptòtica.

```java
/** Precondició: m és 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;
        }
    }
}
```

Talla: Dimensió de la matriu `m`, és a dir, `m.length` = n

Instàncies: No existeixen instàncies diferents.

Si considerem com instrucció crítica la guarda del bucle més intern **`j < i`**, de cost unitari, i es compta com a mesura del cost el nombre de vegades que s'executa aquesta guarda, s'arriba a l'expressió:

$$\mathrm{T}(\mathrm{n}) = \sum_{i=0}^{n-1} \sum_{j=0}^{i} 1 = \sum_{i=0}^{n-1} (i+1) = \sum_{i=0}^{n-1} i + \sum_{i=0}^{n-1} 1 = \sum_{i=0}^{n-1} i + n = \frac{n(n+1)}{2} \text{ i.c.}$$

En notació asimptòtica: $\mathrm{T}(\mathrm{n}) \in \Theta(\mathrm{n}^2)$

El següent mètode **recursiu** ordena per *inserció directa* un array **`v[0..pos]`**, **`0 ≤ pos < v.length`**. Si es vol ordenar tot un array **dades**, la crida inicial al mètode seria: **`insDirectaRec(dades, dades.length - 1);`**

```java
/** 0 <= pos < v.length */
public static void insDirectaRec(int[] v, int pos) {
    if (pos > 0) {
        insDirectaRec(v, pos - 1);
        int x = v[pos];
        int j = pos - 1;
        while (j >= 0 && v[j] > x) {
            v[j + 1] = v[j];
            j--;
        }
        v[j + 1] = x;
    }
}
```

Es demana:

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

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

c) 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) Expressa el resultat anterior utilitzant notació asimptòtica.

```java
/** 0 <= pos < v.length */
public static void insDirectaRec(int[] v, int pos) {
    if (pos > 0) {
        insDirectaRec(v, pos - 1);
        int x = v[pos];
        int j = pos - 1;
        while (j >= 0 && v[j] > x) {
            v[j + 1] = v[j];
            j--;
        }
        v[j + 1] = x;
    }
}
```

Crida inicial per a ordenar tot un array dades:

```java
insDirectaRec(dades, dades.length - 1);
```

Talla: Nombre d'elements de v en consideració, donat per pos + 1 = n

Instàncies: Sí

- **Cas millor**: L'array està ordenat ascendentment. Es fa la crida recursiva i el bucle no fa cap desplaçament, ja que la guarda és `false` perquè v[pos-1]<=x. És a dir, la guarda s'avalua 1 vegada i, per tant, el cost del bucle és constant. Així, el cost del mètode en el cas millor:

$$T_{insDirectaRec}^{m}(n) = \begin{cases} T_{insDirectaRec}^{m}(n-1) + 1 & \text{si } n > 1 \\ 1 & \text{si } n = 1 \end{cases}$$

- **Cas pitjor**: L'array està ordenat descendentment. Es fa la crida recursiva i el bucle s'executa sempre el major número de vegades; triant la guarda com instrucció crítica, el cost del bucle és lineal amb la talla del problema ($\sum_{j=0}^{pos} 1 = pos + 1 = n$). Així, el cost del mètode en el cas pitjor:

$$T_{insDirectaRec}^{p}(n) = \begin{cases} T_{insDirectaRec}^{p}(n-1) + n & \text{si } n > 1 \\ 1 & \text{si } n = 1 \end{cases}$$

Resolent per substitució:

$$T_{insDirectaRec}^{m}(n) = \begin{cases} T_{insDirectaRec}^{m}(n-1) + 1 & \text{si } n > 1 \\ 1 & \text{si } n = 1 \end{cases}$$

$$T^m(n) = T^m(n-1) + 1 = T^m(n-2) + 2 = T^m(n-3) + 3 = \ldots = T^m(n-i) + i = \ldots = 1 + (n-1) = n$$

Apleguem al cas base quan:

$n-i = 1 \rightarrow i = n-1$

$T_{insDirectaRec}^{m}(n) \in \Theta(n)$

☞ $T_{insDirectaRec}(n) \in \Omega(n)$

$$T_{insDirectaRec}^{p}(n) = \begin{cases} T_{insDirectaRec}^{p}(n-1) + n & \text{si } n > 1 \\ 1 & \text{si } n = 1 \end{cases}$$

$$T^p(n) = T^p(n-1) + n = T^p(n-2) + (n-1) + n = T^p(n-3) + (n-2) + (n-1) + n =$$

$$= \ldots = T^p(n-i) + (n-(i-1)) + \ldots + (n-2) + (n-1) + n = \ldots =$$

$$= \boxed{1 + 2 + \ldots + (n-2) + (n-1) + n}$$

$$= \sum_{j=1..n} j = n(n+1)/2$$

Apleguem al cas base quan:

$n-i = 1 \rightarrow i = n-1$

$T_{insDirectaRec}^{p}(n) \in \Theta(n^2)$

☞ $T_{insDirectaRec}(n) \in O(n^2)$