

Exámenes

Preliminares del Tema 2: Cuestiones sobre métodos recursivos (tipos numéricos)


[Volver a la Lista de Exámenes](#)

Parte 1 de 1 -

14.71/ 15.0 Puntos

Preguntas 1 de 4

2.0/ 2.0 Puntos

 [metodosRecursivosNumericos.txt](#) 1 KB

Cada uno de los métodos Java que figuran en el fichero adjunto (haz *click* para abrirlo) es un método recursivo. Para realizar esta afirmación basta observar dos líneas de su código; sabiendo que la primera de ellas es la cabecera del método, responde a las siguientes cuestiones *utilizando el nº de espacios en blanco imprescindibles*.

(a) ¿Cuál es la segunda línea del código del método en la que nos tenemos que fijar?

- Para `factorial`, `multiplicar` y `potencia`, la línea nº ☒ 3.
- Para `maximoCD`, bien la línea nº ☒ 5 o bien la nº ☒ 6.

(b) ¿Qué dos detalles te garantizan la elección correcta de dicha segunda línea de código? Completa el hueco que precede a cada una de las siguientes frases con una C (de Cierto) si describe uno de estos detalles y con una F (de Falso) en caso contrario.

- ☒ F En ella aparece, al contrario que en la cabecera, un `return`.
- ☒ F En ella no aparece, al contrario que en la cabecera, el nombre del método.
- ☒ C En ella aparece, al igual que en la cabecera, el nombre del método.
- ☒ F En ella decrecen los valores de todos los parámetros del método, con respecto a los que tenían en la cabecera.
- ☒ C En ella decrece el valor de uno de los parámetros del método, con respecto al que tenía en su cabecera.
- ☒ F En ella alcanza su valor mínimo alguno de los parámetros del método.

Respuesta correcta: 3, 5, 6, F, F, C|T|V, F, C|T|V, F

Preguntas 2 de 4

5.0/ 5.0 Puntos

Utilizando el nº de espacios en blanco y mayúsculas imprescindibles, ...

(a) Indica qué variable o relación de variables expresan la talla x del problema en los métodos recursivos de la pregunta anterior:

- Para `factorial`, $x = \checkmark \underline{n}$.
- Para `multiplicar`, $x = \checkmark \underline{a}$.
- Para `potencia`, $x = \checkmark \underline{k}$.
- Para `maximoCD`, $x = \text{máximo}(\checkmark \underline{n}, \checkmark \underline{m})$.

(b) ¿En qué líneas del código de estos métodos te has fijado, como mínimo, para definir la talla del problema que resuelven?

- Para `factorial`, `multiplicar` y `potencia`, líneas nº $\checkmark \underline{1}$ y $\checkmark \underline{3}$.
- Para `maximoCD`, bien en las líneas nº $\checkmark \underline{1}$ y $\checkmark \underline{5}$ o bien en las líneas nº $\checkmark \underline{1}$ y $\checkmark \underline{6}$.

(c) ¿En qué líneas del código de estos métodos te debes fijar, sí o sí, para comprobar que terminan en un tiempo finito, i.e. tras un nº de llamadas finito?

- Para `factorial`, `multiplicar` y `potencia`, líneas nº $\checkmark \underline{2}$ y $\checkmark \underline{3}$.
- Para `maximoCD`, bien en las líneas nº $\checkmark \underline{3}$ y $\checkmark \underline{5}$ o bien en las líneas nº $\checkmark \underline{3}$ y $\checkmark \underline{6}$.

(d) Todos estos métodos presentan el mismo tipo de recursión:

- ¿Cuál? $\checkmark \underline{\text{Lineal}}$.
- ¿Por qué?
 - Porque la ejecución de su llamada más alta, su invocación desde un método de otra clase, origina una $\checkmark \underline{\text{secuencia}}$ de llamadas recursivas, y no un $\checkmark \underline{\text{árbol}}$. Ello se puede observar trazando tal ejecución para un argumento de la talla sencillo, pero no trivial.
 - Porque en el caso general de su código el nº de llamadas recursivas en secuencia que aparecen es $\checkmark \underline{1}$.
- Contesta SÍ o NO: ¿Está relacionada con su coste Temporal? $\checkmark \underline{\text{SÍ}}$.
- Contesta SÍ o NO: Sus métodos iterativos equivalentes...
 - ¿Tienen un mejor coste Temporal?: $\checkmark \underline{\text{NO}}$.
 - ¿Tienen un mejor coste Espacial?: $\checkmark \underline{\text{SÍ}}$.

Respuesta correcta:n, a, k, m|n, n|m, 1, 3, 1, 5, 1, 6, 2, 3, 3, 5, 3, 6, Lineal|lineal,

secuencia|Secuencia|Lista|lista|sucesión|Sucesión|sucesion|Sucesion, Árbol|árbol|Arbol|arbol, 1|Uno|uno, SÍ|SI|Sí|Si|sí|si, NO|No|no, SÍ|SI|Sí|Si|sí|si

Preguntas 3 de 4

4.71/ 5.0 Puntos

Para los métodos recursivos de la pregunta anterior, utilizando el nº de espacios en blanco y mayúsculas imprescindibles, ...

(a) Indica el valor de su talla en la última llamada recursiva:

- Para `factorial`, $x = \checkmark \underline{0}$.
- Para `multiplicar`, $x = \checkmark \underline{0}$.
- Para `potencia`, $x = \checkmark \underline{0}$.
- Para `maximoCD`, x tal que $\checkmark \underline{n=m}$.

(b) Indica para qué valor de su talla se establece su resultado definitivo:

- Para `factorial(5)`, cuando $x = \checkmark \underline{5}$.
- Para `multiplicar(4, 3)`, cuando $x = \checkmark \underline{4}$.
- Para `potencia(2, 4)`, cuando $x = \checkmark \underline{4}$.
- Para `maximoCD(2345, 1)`, cuando $x = \checkmark \underline{1}$.

(c) Indica (con un SÍ o un NO) si el resultado del método es igual al de la llamada recursiva, junto con el número de línea de su código que lo explicita.

- Para `factorial` $\checkmark \underline{NO}$, como se observa en su línea nº $\checkmark \underline{3}$.
- Para `multiplicar` $\checkmark \underline{NO}$, como se observa en su línea nº $\checkmark \underline{3}$.
- Para `potencia` $\checkmark \underline{NO}$, como se observa en su línea nº $\checkmark \underline{3}$.
- Para `maximoCD` $\checkmark \underline{SÍ}$, como se observa en su línea nº $\times \underline{5}$.

(d) ¿Cuál de todos estos métodos recursivos es final? Indica su nombre, o identificador: $\checkmark \underline{\text{maximoCD}}$

Respuesta correcta:0, 0, 0, $n=m|n==m|n = m|n == m|m=n|m==n|m = n|m == n$, 5, 4, 4, 1, no|No|NO, 3, no|No|NO, 3, no|No|NO, 3, sí|si|Sí|Si|SÍ|SI, 7, maximoCD

Preguntas 4 de 4

3.0/ 3.0 Puntos

 [metodosRecursivosPotencia.txt](#) 0 KB

[AnimacionMetodosRecursivosPotencia.ppsx](#) 115 KB

Los dos métodos Java que figuran en el fichero adjunto *metodosRecursivosPotencia.txt* (haz *click* para abrirlo) calculan recursivamente el mismo resultado: a elevado a k (a^k). Del primero de ellos ya conoces sus características (talla, tipo de recursión que presenta, etc.), pues es equivalente al método `potencia` de las

preguntas anteriores; el segundo, como su nombre indica, es una versión de `potencia` que se ha diseñado con la intención de mejorar su eficiencia. Para que analices -informalmente- esta versión y sepas si realmente es más eficiente que el método `potencia`, utilizando el nº de espacios en blanco y mayúsculas imprescindibles, ...

(a) ¿Qué primera línea del código de `potenciaV1` "delata" la intención de mejorar la eficiencia de `potencia`? ¿Por qué? La línea nº 4. En ella aparece el parámetro k como parte de la expresión k/2, mientras que el mismo parámetro en la misma línea de `potencia` forma parte de la expresión k-1. Y esto indica la intención de pasar de un coste lineal con la talla a uno logarítmico.

(b) ¿Qué primera línea del código de `potenciaV1` "delata" que la intención de mejorar la eficiencia de `potencia` puede ser fallida? ¿Por qué? La línea nº 5, porque de ella se deduce que el tipo de recursión de `potenciaV1` NO es lineal como el de `potencia` sino múltiple, lo que no es buena señal cuando implica repetir cálculos ya realizados -como sucede en el cálculo recursivo de un término de la sucesión de Fibonacci-

(c) Abre el fichero adjunto *AnimacionMetodosRecursivosPotencia.ppsx* y comprueba que, en efecto, durante la ejecución del método `potenciaV1(2, 3)` se realizan cálculos innecesarios que provocan su coste temporal sea el mismo que el de `potencia` y, además, que su coste espacial sea ... ¡mayor!

Sin embargo, no todo está perdido: fíjate también que haciendo una modificación mínima en una sola línea del código de `potenciaV1` se eliminarían todos los cálculos innecesarios (provocados por las llamadas con flecha en rojo en la animación) y, con ello, se conseguiría la pretendida mejora del coste de `potencia`.

- ¿De qué línea de `potenciaV1` estamos hablando? De la nº 5.
- Escribe la instrucción que aparece en dicha línea PERO con la modificación requerida:
`int resMetodo=resLlamada*resLlamada;`
- Traza de nuevo `potenciaV1(2, 3)` para comprobar que, con la modificación realizada, el tipo de recursión de `potenciaV1` ya es lineal no final, el mismo que el de `potencia`, lo que permite la mejora del coste temporal que se pretendía.

MORALEJA: lleva mucho cuidado al implementar en Java cualquier buena idea para reducir el coste de un método.

(d) Contesta SÍ o NO: El método iterativo equivalente a `potenciaV1`...

- ¿Tienen un mejor coste Temporal?: NO.

- ¿Tienen un mejor coste Espacial?: SÍ.

Respuesta correcta: 4, k, k / 2|k/2|k / 2|k / 2, k - 1|k-1|k - 1|k - 1, lineal|Lineal,

logarítmico|Logarítmico|logaritmico|Logaritmico, 5, lineal|Lineal, múltiple|Múltiple|multiple|Multiple, 5, int
 resMetodo = resLlamada * resLlamada;|int resMetodo=resLlamada*resLlamada;|int resMetodo =
 resLlamada*resLlamada;|int resMetodo=resLlamada * resLlamada;, lineal|Lineal|LINEAL, no final|no Final|NO
 FINAL|No Final|No final, NO|No|no, SÍ|SI|Sí|Si|sí|si

- UPV
- Powered by Sakai
- Copyright 2003-2020 The Sakai Foundation. All rights reserved. Portions of Sakai are copyrighted by other parties as described in the Acknowledgments screen.