

TEMA 2. ÍNDICES INVERTIDOS. TÉRMINOS Y CONSULTAS

Contenidos basados en los materiales de otros cursos como los de Manning y Baeza

Contenidos

1. Introducción
2. Obtener el vocabulario de términos
 - 2.1. Documentos
 - 2.2. Tokenización
 - 2.3. Vocabulario de términos
3. Postings
 - 3.1. Skip list
 - 3.2. Postings posicionales y consultas de Sintagmas

classification

search

precision

recall

links

query

clustering

xml

index

web

ranking

language model

Contents

Christopher D. Manning
Prabhakar Raghavan
Hinrich Schütze

Introduction to Information Retrieval

Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze.
Cambridge University Press, 2009.

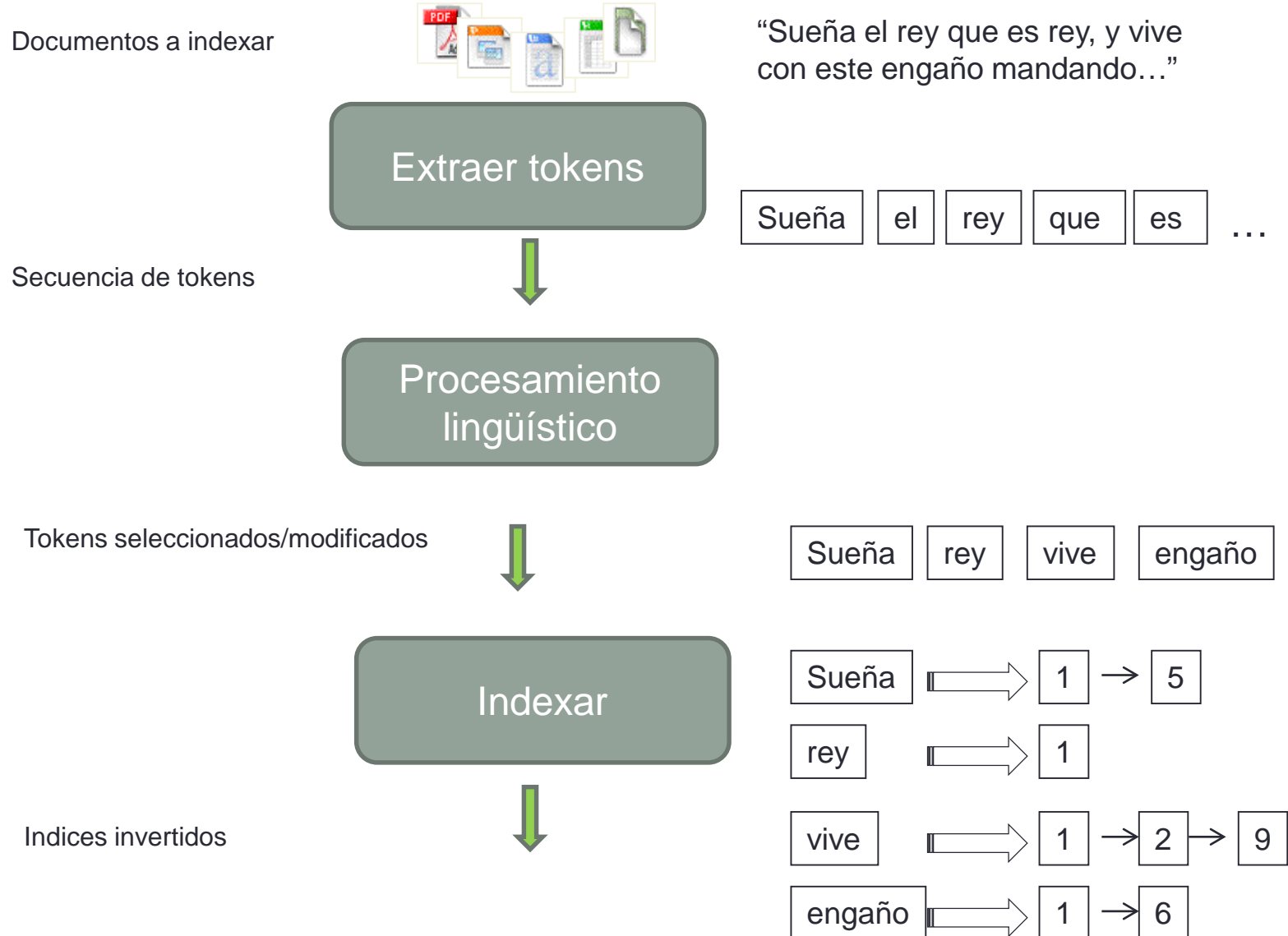
Ricardo Baeza-Yates and Berthier Ribeiro-Neto
Addison Wesley, First printed **1999**

[illegible]

1. INTRODUCCIÓN

Recordemos ...

Etapas de construcción de índices invertidos



Objetivo del Tema: conocer una forma básica de construir un índice,

1) Preproceso para obtener los términos:

- Documentos
- Tokenización
- ¿Qué términos poner en el índice?

2) Postings Lists:

- Mejora de la operación “Intersección”: Skip lists
- Posting posicionales y consultas de Sintagmas

2.OBTENER EL VOCABULARIO DE TÉRMINOS

2.1. Documentos

2.2. Tokenización

2.3. Vocabulario de términos

2.1. Documentos

Decisiones respecto al análisis de los documentos

- ¿De qué tipo de documento se trata (pdf, txt, html, avi..)?
- ¿En qué lengua(-s) está escrito?
- ¿Qué conjunto de caracteres usa?

Son tareas que pueden abordarse como un problema de clasificación, pero se suelen resolver de forma heurística.

2.1. Documentos

Decisiones respecto al documento como unidad

- La colección de documentos puede incluir documentos en diferentes lenguas
➡ Un índice simple debería incluir términos en diferentes lenguas.
- Hay documentos que contienen componentes en diferentes lenguas o formatos.
➡ Un email en francés con un pdf adjunto en alemán.

¿Cuál es la unidad “Documento” (fichero o grupo de ficheros; e-mail; o e-mail y sus ficheros adjuntos,...)?

Formatos de documentos

Texto

- Representación de los códigos de caracteres en formato binario a través de los distintos “esquemas de codificación”:

EBCDIC (7 bits), ASCII (8 bits), UNICODE (16 bits)

- Recuperación desde diversos formatos de documentos de texto (doc, pdf, html, txt, rtf, ps,..)

Otros formatos de intercambio se usan para codificar correos electrónicos y comprimir textos.

- **Multipurpose Internet Mail Exchange (MIME):** para codificar email
- **ZIP:** *gzip* en Unix/linux, *Winzip* en Windows para comprimir texto.

=> Los sistemas de RI tienen que disponer de filtros y conversores de formato para acceder a los tipos de documentos más habituales.

Lenguajes de marcado

Sirven para incorporar información al texto, tal como estructura de la información, acciones de formato, semántica, atributos...

Algunos lenguajes conocidos son:

- ☐ SGML: Standard Generalized Markup Language
- ☐ XML: eXtensible Markup Language
- ☐ HTML: Hyper Text Markup Language

Ejemplo de lenguajes de marcado. HTML

```
<html><head>
<title>HTML Example</title>
<meta name=rby content="Just an example">
</head>
<body>
<h1>HTML Example</h1>
<p><hr><p>HTML has many <i>tags</i>, among them:
<ul>
<li> links to other <a href=example.html>pages</a> (a from anchor),
<li> paragraphs (p), headings (h1, h2, etc), font types (b, i),
<li> horizontal rules (hr), indented lists and items (ul, li),
<li> images (img), tables, forms, etc.
</ul>
<p><hr><p>
This page is <b>always</b> under construction.
</body></html>
```

HTML Example

HTML has many *tags*, among them:

- links to other pages (a from anchor),
- paragraphs (p), headings (h1, h2, etc), font types (b, i),
- horizontal rules (hr), indented lists and items (ul, li),
- images (img), tables, forms, etc.



This page is **always** under construction.

2.2. Tokenización

Dada una secuencia de caracteres y una unidad de documento definida, la tokenización consiste en trocear la secuencia en piezas (**tokens**).

- Un token es una secuencia de caracteres
- Cada token es un candidato para ser una entrada del diccionario.
- ¿Son válidos todos los tokens?
- Ejemplo.

Input: “***Friends, Romans, Countrymen***”

Output: Tokens

- ***Friends***
- ***Romans***
- ***Countrymen***

2.2. Tokenización

Decisiones respecto a tokens

- ***Finland's capital*** →
Finland? Finlands? Finland's?
- ***Hewlett-Packard*** → ***Hewlett*** y ***Packard*** como dos tokens?
 - ***state-of-the-art***
 - ***co-education***
 - ***lowercase, lower-case, lower case*** ?
- ***San Francisco***: como dos tokens?
- **Números, fechas:** *3/12/91, Mar. 12, 1991, 12/3/91, 55 B.C., B-52, (800) 234-2333.*

2.2. Tokenización

Algunos problemas con las distintas lenguas

Francés, Catalán:

- la expresión ***L'ensemble: L ? L' ? Le ?***

Idiomas aglutinantes: Palabras largas

- **Alemán:** *Donaudampfschiffahrtselektrizitätenhauptbetriebswerkbauunterbeamten gesellschaft* («Sociedad de funcionarios subordinados de la construcción de la fábrica principal de la electricidad para la navegación de barcos de vapor en el Danubio»)
- **Turco:** *muvaaffakiyetsizleştiricileştiriveremeyebileceklerimizdenmişsinizcesine*
- **Chino y japonés:** no tienen espacio entre palabras

莎拉波娃现在居住在美国东南部的佛罗里达。

2.2. Tokenización

Algunos problemas con las distintas lenguas....cont

Árabe: El árabe se escribe de derecha a izquierda, pero ciertos elementos como los números se escriben de izquierda a derecha.

استقلت الجزائر في سنة 1962 بعد 132 عاما من الاحتلال الفرنسي.

Lenguas que utilizan múltiples alfabetos

フォーチュン500社は情報不足のため時間あた\$500K(約6,000万円)

2.2. Tokenización

Stopwords (palabras “muy” frecuentes)

Conjunto de palabras con poco significado.

Métodos:

- Crear listas “fijas” de palabras para cada idioma:
Inglés: a, cannot, into, our, thus, about, co, is, ours, to, above, could, it, ourselves, together, ...
Castellano: él, éstos, última, últimas, aún, actualmente, adelante, además, ahí, ahora, de, ...
- Crear listas de categorías léxicas: (preposiciones, artículos, conjunciones, pronombres, adverbios).
- Crear listas a partir de la frecuencia de las palabras en la colección de documentos.

2.2. Tokenización

Problemas al eliminar las **Stopwords**:

- Pueden ser necesarias:

*“Rey **de** Dinamarca” ,*

*“Vitamina **A**”,*

*“¿Qué significa **ESA**?”,*

*“**to be or not to be**”,*

*“**En un lugar de La Mancha**”,....*

- Las buenas técnicas de compresión hacen que el espacio necesario para incluir stopwords no sea muy grande.

- **Lista de stopwords(castellano):**

él ésta éstas éste éstos última últimas último últimos a añadió aún actualmente adelante además afirmó agregó ahí ahora al algún algo alguna algunas alguno algunos alrededor ambos ante anterior antes apenas aproximadamente aquí así aseguró aunque ayer bajo bien buen buena buenas bueno buenos cómo cada casi cerca cierto cinco comentó como con conocer consideró considera contra cosas creo cual cuales cualquier cuando cuanto cuatro cuenta da dado dan dar de debe deben debido decir dejó del demás dentro desde después dice dicen dicho dieron diferente diferentes dijeron dijo dio donde dos durante e ejemplo el ella ellas ello ellos embargo en encuentra entonces entre era eran es esa esas ese eso esos está están esta estaba estaban estamos estar estará estas este esto estos estoy estuvo ex existe existen explicó expresó fin fue fuera fueron grandes ha había habían haber habrá hace hacen hacer hacerlo hacia haciendo han hasta hay haya he hecho hemos hicieron hizo hoy hubo igual incluso indicó informó junto la lado las le les llegó lleva llevar lo los luego lugar más manera manifestó mayor me mediante mejor mencionó menos mi mientras misma mismas mismo mismos momento mucha muchas mucho muchos muy nada nadie ni ningún ninguna ningunas ninguno ningunos no nos nosotras nosotros nuestra nuestras nuestro nuestros nueva nuevas nuevo nuevos nunca o ocho otra otras otro otros para parece parte partir pasada pasado pero pesar poca pocas poco pocos podemos podrá podrán podría podrían poner por porque posible próximo próximos primer primera primero primeros principalmente propia propias propio propios pudo pueda puede pueden pues qué que quedó queremos quién quien quienes quiere realizó realizado realizar respecto sí sólo se señaló sea sean según segunda segundo seis ser será serán sería si sido siempre siendo siete sigue siguiente sin sino sobre sola solamente solas solo solos son su sus tal también tampoco tan tanto tenía tendrá tendrán tenemos tener tenga tengo tenido tercera tiene tienen toda todas todavía todo todos total tras trata través tres tuvo un una unas uno unos usted va vamos van varias varios veces ver vez y ya yo

- Lista de stopwords (inglés):

a about above across actually after again against ago all almost along
already also although always among an and another any anyone around as
at b back bad because before behind best better between big biggest both
but by c cent complete d day down during e each early eight enough entire ep
etc even ever every everything f face fact far fell few finally first five for found
four from g good got h he held her here him himself his hour hours how
however i idea if in including instead into it its itself j k l lack last later least led
less little long longer lot m man many matter may me men miles million
moment month months more morning most much my n near nearly necessary
never night no nor not note nothing now o of off often on once one only or
other others our out outside over own p page part past per perhaps place
point proved q qm question r really recent recently reported round s same sec
second section sense seven she short should showed since single six small
so some soon still such t ten text than that the their them themselves then
there these they thing things third this those though thought thousands three
through thus time tiny to today together too took toward two u under until up
upon us v very w warning way we week weeks well went were what when
where whether which while who whom whose why will with without word
words would x y year years yet you your z

2.3. Vocabulario de términos

Normalización de términos

Las palabras se deben normalizar:

- al procesar los documentos (off-line),
- al procesar las consultas (on-line).

Por ejemplo, deben considerarse iguales U.S.A. y USA?

Término: Un término es una “palabra” (normalizada) que será una entrada en el diccionario del sistema de RI.

2.3. Vocabulario de términos

Decisiones sobre normalización de términos

(siempre las mismas en la indexación y en la consulta)

Ejemplo.-

- Acentos: *résumé, Tübingen, acción.*
- Eliminar puntos: U.S.A. como USA
- Eliminar guiones (hyphens):
anti-discriminatorio como antidiscriminatorio
- Eliminar mayúsculas:
Excepciones (nombres? Instituciones? Países? ...)
Ejemplo: *General Motors*

2.3. Vocabulario de términos

Lematización

Eliminar las variantes/inflexiones de las palabras, almacenando sólo el lema o forma base.

Ejemplo:

come, comía, comerá → comer

am, are, is → be

car, cars, car's, cars' → car

amarillo, amarilla, amarillos → amarillo

Stemming

Reducir los términos a sus “raíces”, eliminando los sufijos.

Ejemplo:

automate(s), automatic, automation → automat

2.3. Vocabulario de términos

Stemmer para el inglés: *Algoritmo de Porter.*

Aplica reglas para eliminar los sufijos de las palabras.

- Grupos de Reglas aplicadas al sufijo más largo:

<i>sses</i> → <i>ss</i>	(<i>caresses</i> → <i>caress</i>)
<i>ies</i> → <i>i</i>	(<i>ponies</i> → <i>poni</i>)
<i>ss</i> → <i>ss</i>	(<i>caress</i> → <i>caress</i>)
<i>s</i> →	(<i>cats</i> → <i>cat</i>)

- Reglas dependientes de la longitud de las palabras:

(*m* > 1) *EMENT*:
replacement → *replac*
cement → *cement*

2.3. Vocabulario de términos

Stemmer para el inglés: *Algoritmo de Porter.*

The project is a final examination exercise in the Informatics Engineering and in both Diplomas. Consequently, the student is required to show that he or she is able to apply the knowledge acquired during his/her studies to typical informatics engineering situations, allowing for the different specific course objectives, which are defined in the curriculum. For the Informatics Engineering these are to provide the student with a broad base and professional skills in software engineering and information systems for organisations, informatics engineering, or industrial information technology...

2.3. Vocabulario de términos

Stemmer para el inglés: *Algoritmo de Porter.*

the project is a final exam in the informatics engine and in both diploma. consequently, the student is required to show that he or she is able to apply the knowledge acquired in his/her studies to typical informatics situations, allow for the different specific course objects, which are defined in the curriculum. for the informatics engine these are to provide the student with a broad base and professional skill in software engineering and information systems for organisations, informatics engine, or industrial information technology...

2.3. Vocabulario de términos

Stemming: pros and cons

- Pros:
 - Mejora la efectividad de la recuperación, en cierta medida
 - Casi tan efectivo como la lematización
 - Necesita menos conocimiento de la lengua que la lematización
 - Más rápido, fácil de describir y de implementar
- Cons:
 - La salida no es legible
 - Puede reducir a la misma raíz palabras diferentes
 - Relative (family)
 - Relativity (physics)
 - Relativism (philosophy)

2.3. Vocabulario de términos

Enriquecimiento (consulta)

- Sinónimos
gun, arm, weapon -> weapon
- Palabras relacionadas
laptop -> portable computer, light PC
- Categorías (semantic web)
Ski, soccer, marathon -> sports
- Información adicional (part of speech)
- Herramientas de Procesamiento del Lenguaje Natural

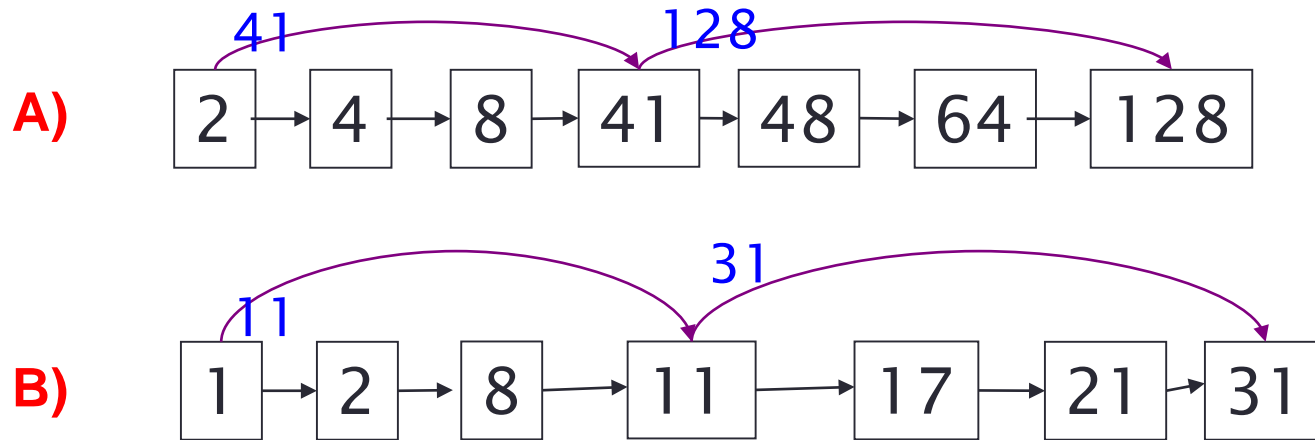
3. POSTINGS

3.1. Skip list

3.2. Posting posicionales y consultas de Sintagmas

3.1 Skip list

Modificar las posting list incluyendo “*skip pointers*”.



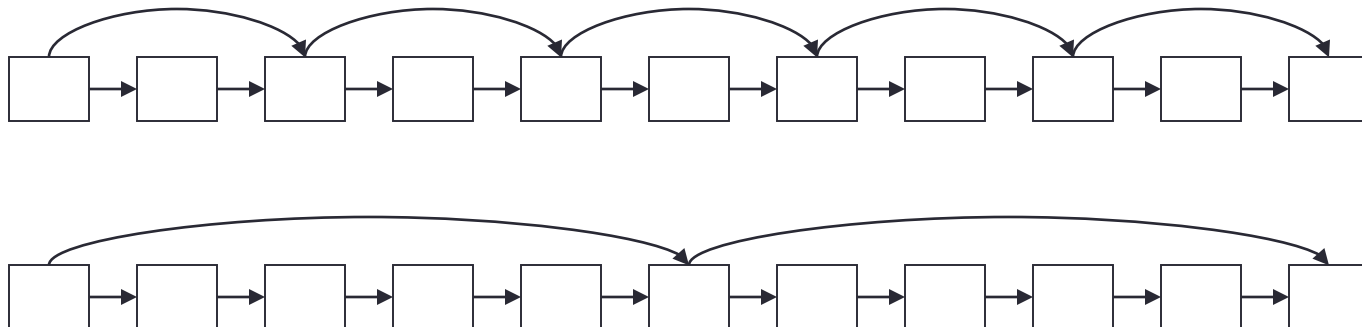
- Supongamos que hemos explorado ambas listas hasta llegar a **8** en cada lista.
- Los siguientes elementos que tenemos son **41** y **11**.
- Como **11** es menor, tenemos que seguir en esa lista, pero podemos comparar con su *skip pointer* que vale 31 (y sigue siendo menor que 41) por lo que podemos saltar todo ese tramo de la lista y seguir por el 31.

3.1 Skip list

¿Dónde y cuántos skip pointers?

Equilibrio:

- Si hay muchos skip-pointers entonces es más fácil que se produzcan saltos, aunque serán cortos. Además hay un coste añadido en las comparaciones con los skip-pointers.
- Si hay pocos entonces hay menos comparaciones con los skip pointers y los saltos son mayores, pero es menos probable que se produzcan.



3.1 Skip list

Algoritmo rápido: operación “Intersección con skips” de posting lists

ALGORITMO INTERSECCION_CON_SKIPS (p1, p2)

respuesta $\leftarrow \{\}$

mientras No_FINAL(p1) AND No_FINAL(p2)

hacer si docID (p1) = docID (p2)

entonces Añadir (respuesta, docID (p1))

 p1 \leftarrow Avanzar_Siguiente(p1)

 p2 \leftarrow Avanzar_Siguiente(p2)

sino si docID (p1) < docID (p2)

entonces si Tiene_Skip(p1) AND (docID(skip(p1)) \leq docID (p2))

entonces mientras Tiene_Skip(p1) AND (docID(skip(p1)) \leq docID (p2))

hacer p1 \leftarrow skip(p1)

sino p1 \leftarrow Avanzar_Siguiente(p1)

sino si Tiene_Skip(p2) AND (docID(skip(p2)) \leq docID (p1))

entonces mientras Tiene_Skip(p2) AND (docID(skip(p2)) \leq docID (p1))

hacer p2 \leftarrow skip(p2)

sino p2 \leftarrow Avanzar_Siguiente(p2)

3.1 Skip list

Ejercicio#1

Suponer una consulta de dos palabras (AND) que proporcionan las siguientes posting lists.

[4 6 10 12 14 16 18 20 22 32 47 81 120 122 157 180]

[47]

Cuántas comparaciones se realizan suponiendo:

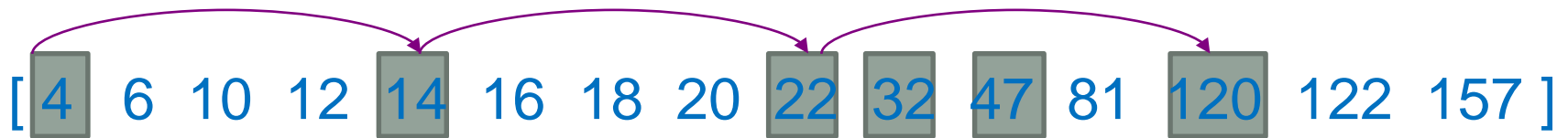
- ☐ que no hay skip pointers
- ☐ que hay skip pointers de longitud 4

3.1 Skip list

Ejercicio#1

Suponer una consulta de dos palabras (AND) que proporcionan las siguientes posting lists.

[4 6 10 12 14 16 18 20 22 32 47 81 120 122 157]



[47]

Cuántas comparaciones se realizan suponiendo:

- ☐ que no hay skip pointers: 11
- ☐ que hay skip pointers de longitud 4: 6

Ejercicio#2.

Los posting lists implementados con skip pointers, ¿son útiles para las operaciones?:

- a) t1 and t2
- b) t1 or t2
- c) (t1) and (not t2)

siendo t1 y t2 los términos que se buscan.

Ejercicio#2.

Los posting lists implementados con skip pointers, ¿son útiles para las operaciones?:

- a) t1 and t2
- b) t1 or t2
- c) (t1) and (not t2)

siendo t1 y t2 los términos que se buscan.

Solución:

- a) Es útil para la operación **AND** porque al buscar sólo coincidencias evita comparaciones que se saltan con los skip pointers.
- b) No es útil para la operación OR porque deben incluirse todos los documentos de ambas listas (evitando repeticiones).
- c) En este caso podrían ser útiles los skip pointers de t2 ya que permitirían acceder rápidamente al documento de t1 que queremos saber si existe en t2.

3.2. Postings posicionales y consultas de Sintagmas

Supongamos que queremos acceder a un sintagma compuesto por varias palabras

- Sintagma Preposicional: “*En un lugar de la Mancha*”
- Sintagma Nominal: “*Universidad de Barcelona*” o “*Leo Messi*”.

La aproximación de indexar cada palabra por separado y hacer operaciones AND entre las postings lists correspondientes puede devolver resultados no deseados:

- “Estas jornadas, celebradas en Barcelona recientemente, contaron con la presencia de investigadores de la universidad.”

3.2. Posting posicionales y consultas de Sintagmas

Primera solución simple: **Indexar pares de palabras consecutivas (biword = término)**

“Universitat Politècnica València”

generaría las siguientes entradas en el diccionario:

Universidad Politècnica

Politècnica València

→ La consulta de Sintagmas de dos palabras se realiza de forma trivial.

→ La consulta de Sintagmas de más de dos palabras se realiza obteniendo secuencias de longitud dos.

3.2. Posting posicionales y consultas de Sintagmas

Problemas con Biwords:

- La talla del diccionario se hace muy grande.
- La búsqueda de palabras sueltas requiere otro diccionario o realizar la unión de los resultados del diccionario de biwords.

Buscar sintagmas de más de dos palabras no es trivial.
Calcular la intersección del resultado de la búsqueda de

“Universitat Politècnica” y “Politécnica València”

no garantiza que los documentos encontrados tengan

“Universitat Politècnica València”

3.2. Posting posicionales y consultas de Sintagmas

Indices posicionales (o full inverted index)

Para cada término se almacenan postings de la forma:
identificador de documento y posiciones en que aparece
el término:

```
<term, number of docs containing term;  
doc1: position1, position2 ... ;  
doc2: position1, position2 ... ;  
etc.>
```

3.2. Posting posicionales y consultas de Sintagmas

Ejercicio#3:

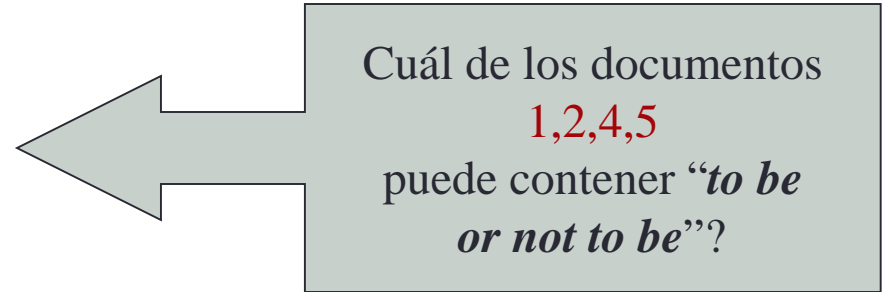
<*be*: 993427;

1: 7, 18, 33, 72, 86, 231;

2: 3, 149;

4: 17, 191, 291, 430, 434;

5: 363, 367, ...>



3.2. Posting posicionales y consultas de Sintagmas

Búsqueda de Sintagmas en índices posicionales

- Extraer las posting lists de cada término: ***to, be, or, not.***
- Hacer “Intersección” de las listas [*docID,tf:positions*] *para encontrar todas las posiciones en que aparece “to be or not to be”.*

to, 993427: [2,5:[1,17,74,222,551]; 4,5:[8,16,190,**429,433**];
7,3:[13,23,191]; ...]

be, 178239: [1,2:[17,19]; 4,5:[17,191,291,**430,434**];
5,3:[14,19,101]; ...]

Requiere trabajar con los offsets entre las palabras y comprobar la compatibilidad con la consulta

3.2. Posting posicionales y consultas de Sintagmas

La talla del índice posicional

- Incluir índices posicionales aumenta la talla de la *postings list* considerablemente
- Sin embargo lo normal es utilizarlos porque son muy útiles para las búsquedas de sintagmas.
- Un posting tendrá una entrada por cada ocurrencia de un término.
- La talla del índice depende de la talla promedio de los documentos.

Por ejemplo, en promedio, una página web tiene una longitud <1000 mientras que un libro puede tener una longitud de 100.000.

3.2. Posting posicionales y consultas de Sintagmas

Si consideramos un término con frecuencia del 1 por 1000 (0.1%) de promedio.

Talla de los documentos	Postings	Positional postings
1 000	1	1
1 00,000	1	100

En general, un índice posicional es de 2 a 4 veces la talla de uno sin posiciones y corresponde aproximadamente de 35–50% del volumen del texto original.

Ejercicio#4

Considerar los siguientes fragmentos de un índice invertido en el cual se ha almacenado la frecuencia de cada término en cada documento indicando la posición de cada ocurrencia. El primer número de cada línea antes de ':' identifica el documento.

- a) ¿ Qué documentos satisfacen la consulta: la AND perla AND negra?
- b) ¿ Cuántas veces podemos encontrar el sintagma “la perla negra” en cada documento?

la 3:34,38,55; 5:12,16,25,44; 7:67,87,90,101; 10:33,39,45,62;	perla 3:12,15,19; 5:3,5,17,41,45,96; 6:21,25,55,62; 7:4,68,70,85,110; 10:15,34,40,65,81;	negra 3:22,26; 5:18,46,52,65; 7:5,69,91,105; 8:32,42,65,93; 10:32,44,75,83;
---	---	--

Ejercicio#4

Considerar los siguientes fragmentos de un índice invertido en el cual se ha almacenado la frecuencia de cada término en cada documento indicando la posición de cada ocurrencia. El primer número de cada línea antes de ':' identifica el documento.

- a) ¿ Qué documentos satisfacen la consulta: la AND perla AND negra?
- b) ¿ Cuántas veces podemos encontrar el sintagma “la perla negra” en cada documento?

la 3:34,38,55; 5:12,16,25,44; 7:67,87,90,101; 10:33,39,45,62;	perla 3:12,15,19; 5:3,5,17,41,45,96; 6:21,25,55,62; 7:4,68,70,85,110; 10:15,34,40,65,81;	negra 3:22,26; 5:18,46,52,65; 7:5,69,91,105; 8:32,42,65,93; 10:32,44,75,83;
---	---	--

Solución:

a.

Los documentos que satisfacen la intersección de los términos son 3, 5, 7 y 10.

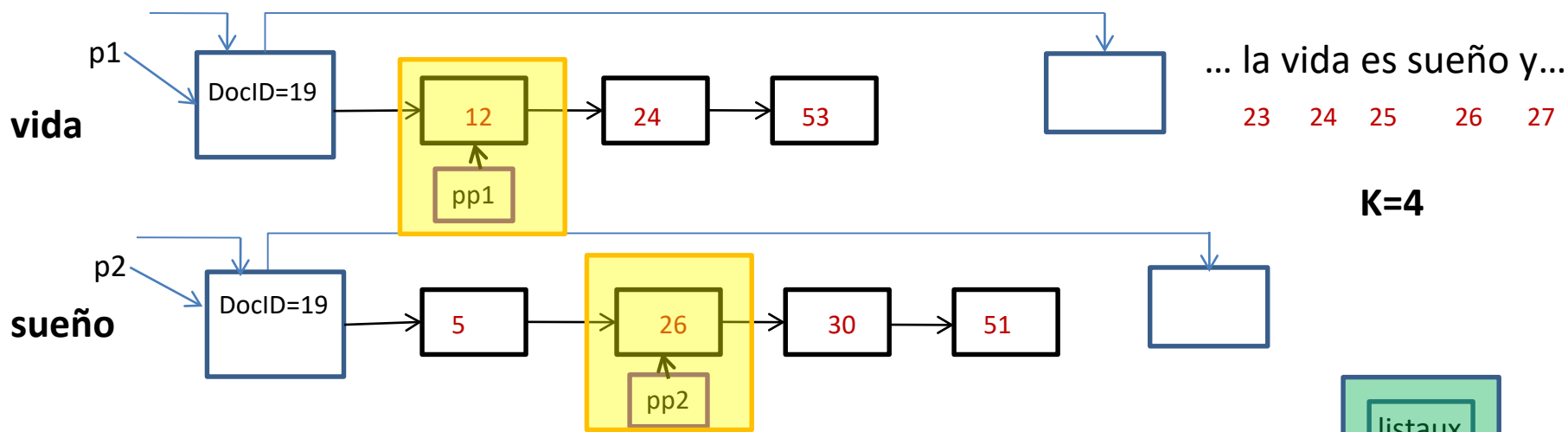
b.

En el documento 5 se encuentra 2 veces (posiciones 16-18 y 44-46), en el documento 7 se encuentra 1 vez (posición 67-69), en el resto de documentos 0 veces.

ALGORITMO INTERSECCION_POSICIONAL (p1, p2,k)

```
respuesta ← [ ]
mentras No_final(p1) and No_final(p2)
hacer si docID (p1) = docID (p2)
    entonces listaux = [ ]
        pp1= posición (p1)
        pp2= posición (p2)
        mentras No_final (pp1)
        hacer mentras No_final (pp2)
            hacer si |pos(pp1)-pos(pp2)| ≤ k
                entonces añadir (listaux,pos(pp2))
                sino si pos(pp2)>pos(pp1)
                    then break
            pp2 = siguiente (pp2)
        mentras (listaux ≠ [ ]) and (|listaux[0]- pos(pp1)|>k)
        hacer eliminar (listaux[0])
        paratodo ps∈listaux
            hacer añadir(respuesta, [docID(p1), pos(pp1),ps])
        pp1= siguiente(pp1)
    p1 = siguiente (p1)
    p2 = siguiente (p2)
sino si docID (p1) < docID (p2)
    entonces p1 ← siguiente(p1)
    sino p2 ← siguiente(p2)

return respuesta
```

ALGORITMO INTERSECCION_POSICIONAL ($p1, p2, k$)

respuesta \leftarrow []

mientras No_final($p1$) and No_final($p2$)

hacer si docID ($p1$) = docID ($p2$)

entonces listaux = []

pp1= posición ($p1$)

pp2= posición ($p2$)

mientras No_final ($pp1$)

hacer mientras No_final ($pp2$)

\rightarrow **hacer** si $|\text{pos}(pp1) - \text{pos}(pp2)| \leq k$

entonces añadir (listaux, pos($pp2$))

sino si pos($pp2$) > pos($pp1$)

then break

pp2 = siguiente ($pp2$)

mientras (listaux \neq []) and ($|\text{listaux}[0] - \text{pos}(pp1)| > k$)

hacer eliminar (listaux[0])

paratodo $ps \in$ listaux

hacer añadir(respuesta, [docID($p1$), pos($pp1$), ps])

pp1= siguiente($pp1$)

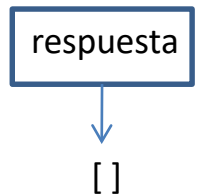
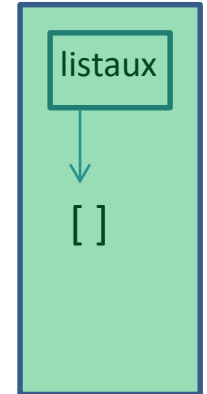
$p1$ = siguiente ($p1$)

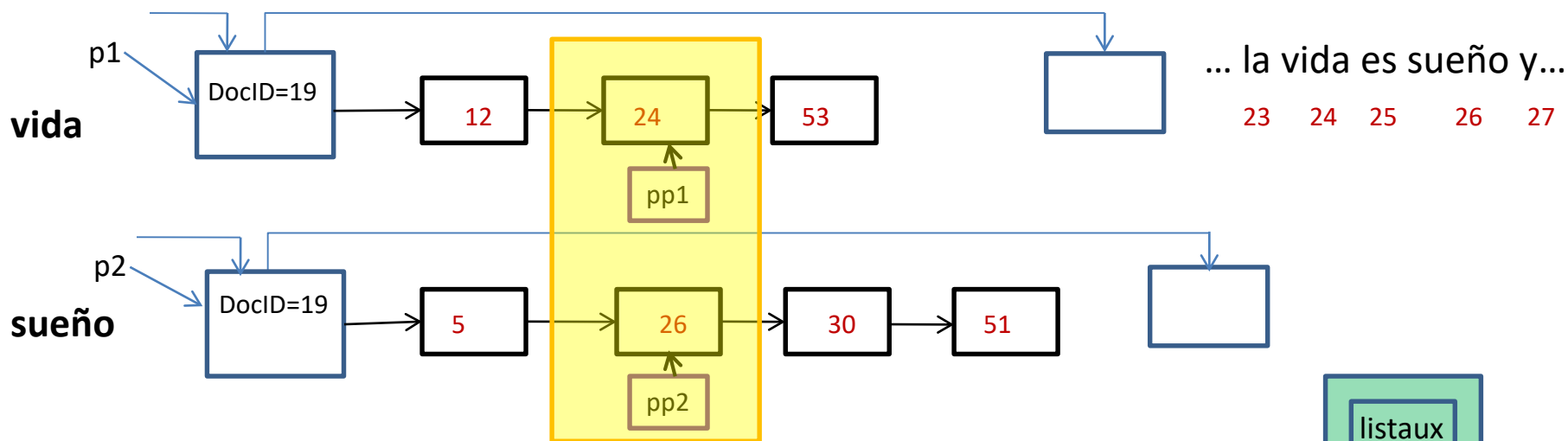
$p2$ = siguiente ($p2$)

sino si docID ($p1$) < docID ($p2$)

entonces $p1 \leftarrow$ siguiente($p1$)

sino $p2 \leftarrow$ siguiente($p2$)





ALGORITMO INTERSECCION_POSICIONAL (p1, p2, k)

respuesta ← []

mientras No_final(p1) and No_final(p2)

hacer si docID (p1) = docID (p2)

entonces listaux = []

pp1= posición (p1)

pp2= posición (p2)

mientras No_final (pp1)

hacer **mientras** No_final (pp2)

➡ **hacer** si $|\text{pos}(\text{pp1}) - \text{pos}(\text{pp2})| \leq k$

entonces añadir (listaux, pos(pp2))

sino si $\text{pos}(\text{pp2}) > \text{pos}(\text{pp1})$

then break

pp2 = siguiente (pp2)

mientras (listaux ≠ []) and $(|\text{listaux}[0] - \text{pos}(\text{pp1})| > k)$

hacer eliminar (listaux[0])

paratodo ps ∈ listaux

hacer añadir(respuesta, [docID(p1), pos(pp1), ps])

pp1= siguiente(pp1)

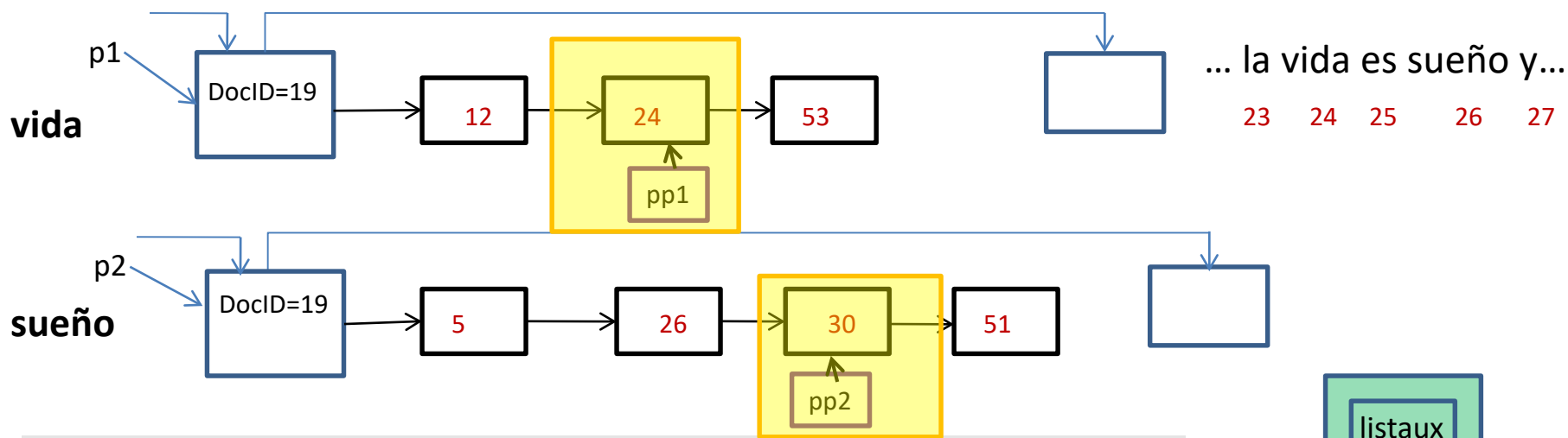
p1 = siguiente (p1)

p2 = siguiente (p2)

sino si docID (p1) < docID (p2)

entonces p1 ← siguiente(p1)

sino p2 ← siguiente(p2)



ALGORITMO INTERSECCION_POSICIONAL (p1, p2, k)

respuesta \leftarrow []

mientras No_final(p1) and No_final(p2)

hacer si docID (p1) = docID (p2)

entonces listaux = []

pp1= posición (p1)

pp2= posición (p2)

mientras No_final (pp1)

hacer **mientras** No_final (pp2)

hacer si $|\text{pos}(\text{pp1}) - \text{pos}(\text{pp2})| \leq k$

entonces añadir (listaux, pos(pp2))

sino si $\text{pos}(\text{pp2}) > \text{pos}(\text{pp1})$

then break

pp2 = siguiente (pp2)

mientras (listaux \neq []) and $(|\text{listaux}[0] - \text{pos}(\text{pp1})| > k)$

hacer eliminar (listaux[0])

paratodo ps \in listaux

hacer añadir(respuesta, [docID(p1), pos(pp1), ps])

pp1= siguiente(pp1)

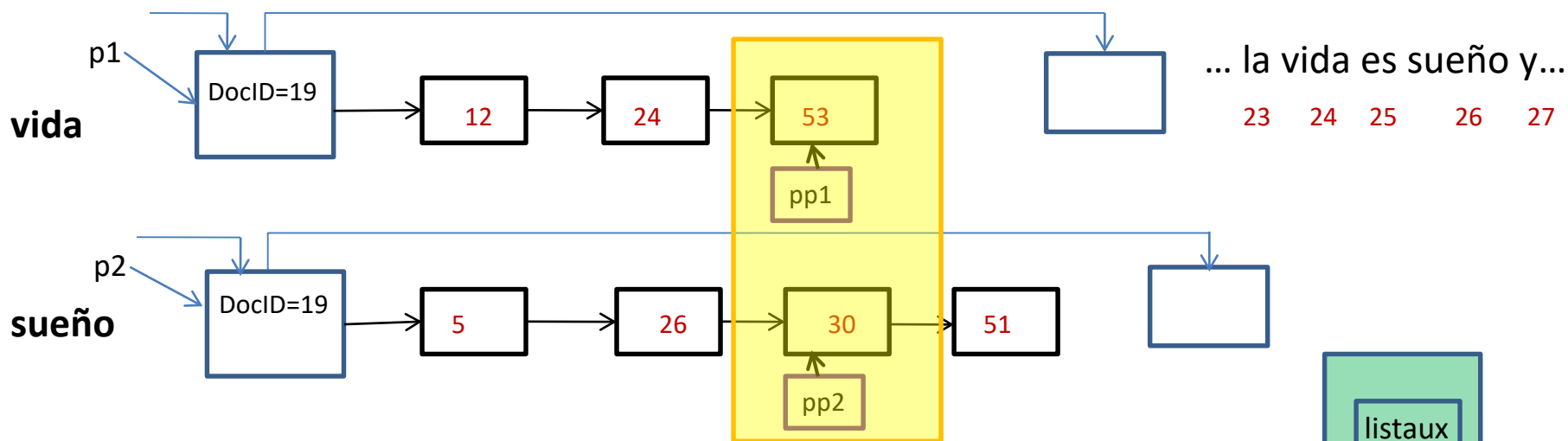
p1 = siguiente (p1)

p2 = siguiente (p2)

sino si docID (p1) < docID (p2)

entonces p1 \leftarrow siguiente(p1)

sino p2 \leftarrow siguiente(p2)



ALGORITMO INTERSECCION_POSICIONAL (p1, p2, k)

respuesta \leftarrow []

mientras No_final(p1) and No_final(p2)

hacer si docID (p1) = docID (p2)

entonces listaux = []

pp1= posición (p1)

pp2= posición (p2)

mientras No_final (pp1)

hacer **mientras** No_final (pp2)

hacer si $|\text{pos}(\text{pp1}) - \text{pos}(\text{pp2})| \leq k$

entonces añadir (listaux, pos(pp2))

sino si $\text{pos}(\text{pp2}) > \text{pos}(\text{pp1})$

then break

pp2 = siguiente (pp2)

mientras (listaux \neq []) and ($|\text{listaux}[0] - \text{pos}(\text{pp1})| > k$)

hacer eliminar (listaux[0])

paratodo ps \in listaux

hacer añadir(respuesta, [docID(p1), pos(pp1), ps])

pp1= siguiente(pp1)

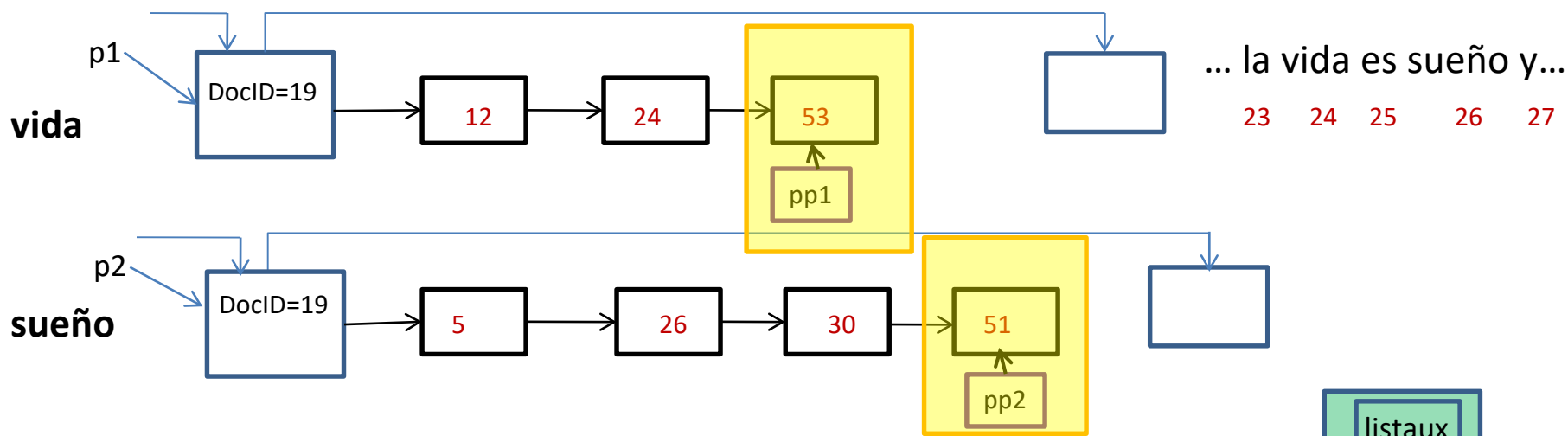
p1 = siguiente (p1)

p2 = siguiente (p2)

sino si docID (p1) < docID (p2)

entonces p1 \leftarrow siguiente(p1)

sino p2 \leftarrow siguiente(p2)



ALGORITMO INTERSECCION_POSICIONAL (p1, p2, k)

respuesta ← []

mientras No_final(p1) and No_final(p2)

hacer si docID (p1) = docID (p2)

entonces listaux = []

pp1= posición (p1)

pp2= posición (p2)

mientras No_final (pp1)

hacer **mientras** No_final (pp2)

➡ **hacer** si $|\text{pos}(\text{pp1}) - \text{pos}(\text{pp2})| \leq k$

entonces añadir (listaux, pos(pp2))

sino si $\text{pos}(\text{pp2}) > \text{pos}(\text{pp1})$

then break

pp2 = siguiente (pp2)

mientras (listaux ≠ []) and $(|\text{listaux}[0] - \text{pos}(\text{pp1})| > k)$

hacer eliminar (listaux[0])

paratodo ps ∈ listaux

hacer añadir(respuesta, [docID(p1), pos(pp1), ps])

pp1= siguiente(pp1)

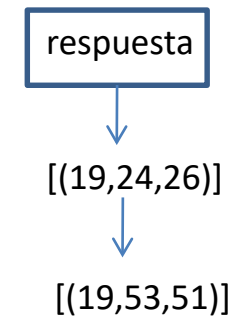
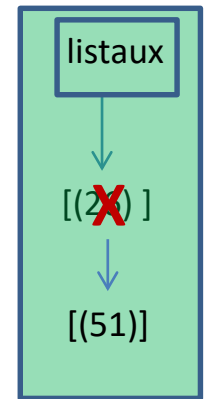
p1 = siguiente (p1)

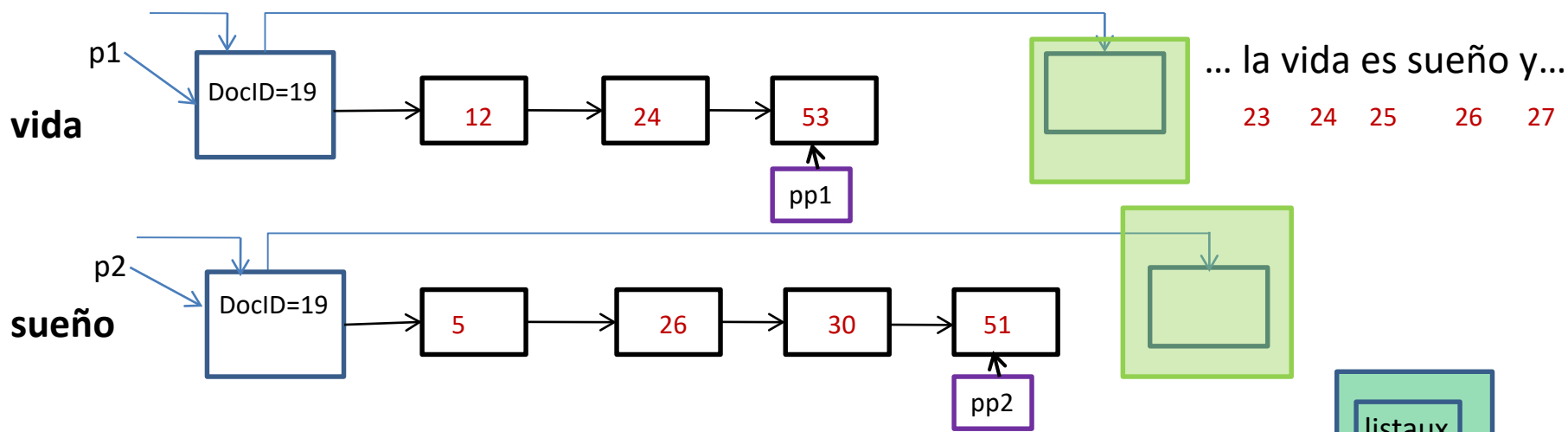
p2 = siguiente (p2)

sino si docID (p1) < docID (p2)

entonces p1 ← siguiente(p1)

sino p2 ← siguiente(p2)





ALGORITMO INTERSECCION_POSICIONAL (p1, p2, k)

respuesta ← []

mientras No_final(p1) and No_final(p2)

hacer si docID (p1) = docID (p2)

entonces listaux = []

pp1= posición (p1)

pp2= posición (p2)

mientras No_final (pp1)

hacer **mientras** No_final (pp2)

hacer si |pos(pp1)-pos(pp2)| ≤ k

entonces añadir (listaux,pos(pp2))

sino si pos(pp2)>pos(pp1)

then break

pp2 = siguiente (pp2)

mientras (listaux ≠ []) and (|listaux[0]- pos(pp1)|>k)

hacer eliminar (listaux[0])

paratodo ps∈listaux

hacer añadir(respuesta, [docID(p1), pos(pp1),ps])

pp1= siguiente(pp1)

p1 = siguiente (p1)

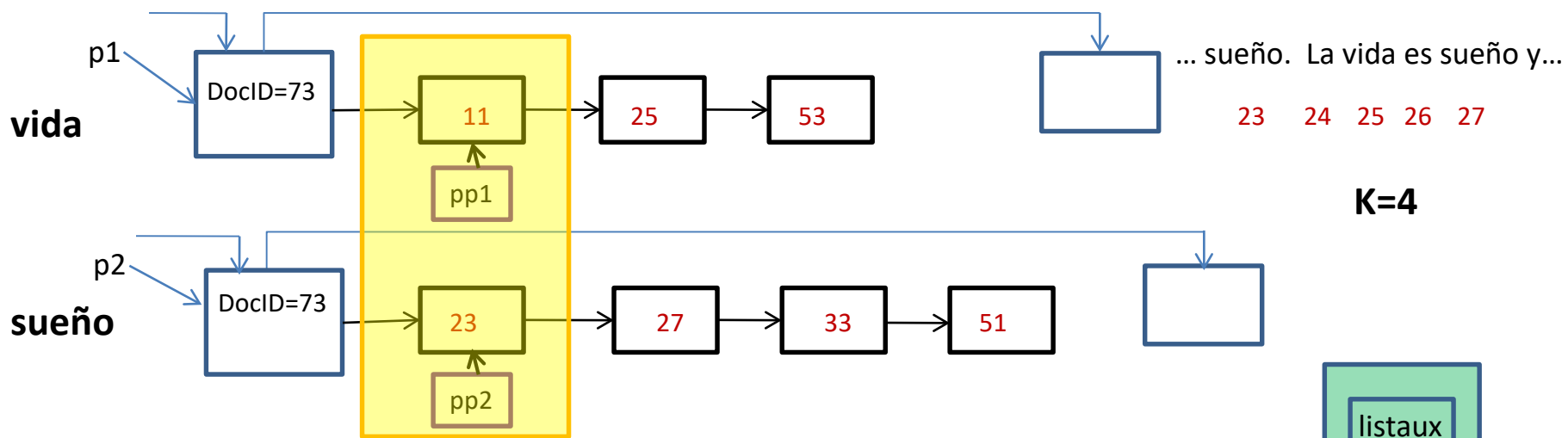
p2 = siguiente (p2)

sino si docID (p1) < docID (p2)

entonces p1 ← siguiente(p1)

sino p2 ← siguiente(p2)

una situación más compleja...



ALGORITMO INTERSECCION_POSICIONAL (p1, p2, k)

respuesta \leftarrow []

mientras No_final(p1) and No_final(p2)

hacer si docID (p1) = docID (p2)

entonces listaux = []

pp1 = posición (p1)

pp2 = posición (p2)

mientras No_final (pp1)

hacer **mientras** No_final (pp2)

hacer si $|\text{pos}(\text{pp1}) - \text{pos}(\text{pp2})| \leq k$

entonces añadir (listaux, pos(pp2))

sino si $\text{pos}(\text{pp2}) > \text{pos}(\text{pp1})$

then break

pp2 = siguiente (pp2)

mientras (listaux \neq []) and $(|\text{listaux}[0] - \text{pos}(\text{pp1})| > k)$

hacer eliminar (listaux[0])

paratodo ps \in listaux

hacer añadir(respuesta, [docID(p1), pos(pp1), ps])

pp1 = siguiente(pp1)

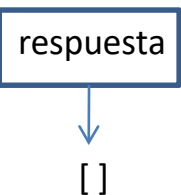
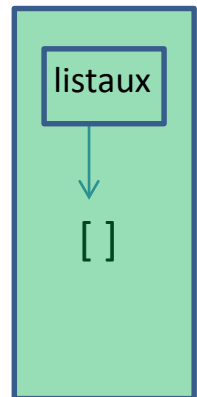
p1 = siguiente (p1)

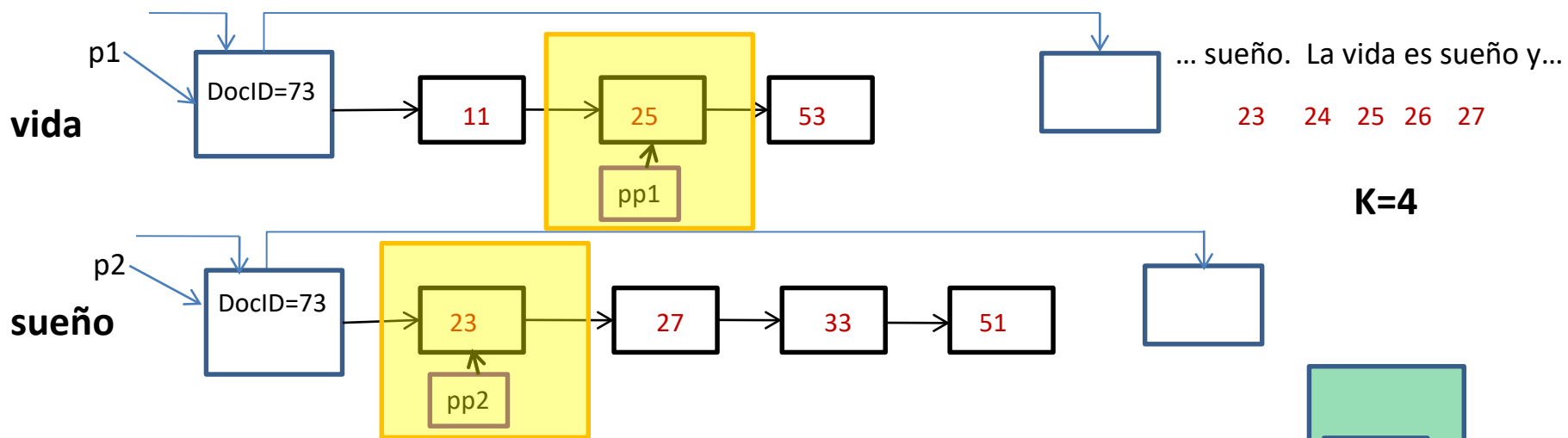
p2 = siguiente (p2)

sino si docID (p1) < docID (p2)

entonces p1 \leftarrow siguiente(p1)

sino p2 \leftarrow siguiente(p2)





ALGORITMO INTERSECCION_POSICIONAL (p1, p2, k)

respuesta ← []

mientras No_final(p1) and No_final(p2)

hacer si docID (p1) = docID (p2)

entonces listaux = []

pp1= posición (p1)

pp2= posición (p2)

mientras No_final (pp1)

hacer **mientras** No_final (pp2)

➡ **hacer** si $|\text{pos}(\text{pp1}) - \text{pos}(\text{pp2})| \leq k$

entonces añadir (listaux, pos(pp2))

sino si $\text{pos}(\text{pp2}) > \text{pos}(\text{pp1})$

then break

➡ pp2 = siguiente (pp2)

mientras (listaux ≠ []) and $(|\text{listaux}[0] - \text{pos}(\text{pp1})| > k)$

hacer eliminar (listaux[0])

paratodo ps ∈ listaux

hacer añadir(respuesta, [docID(p1), pos(pp1), ps])

➡ pp1 = siguiente(pp1)

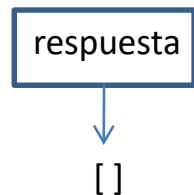
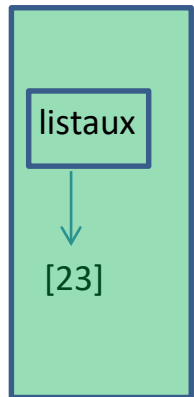
p1 = siguiente (p1)

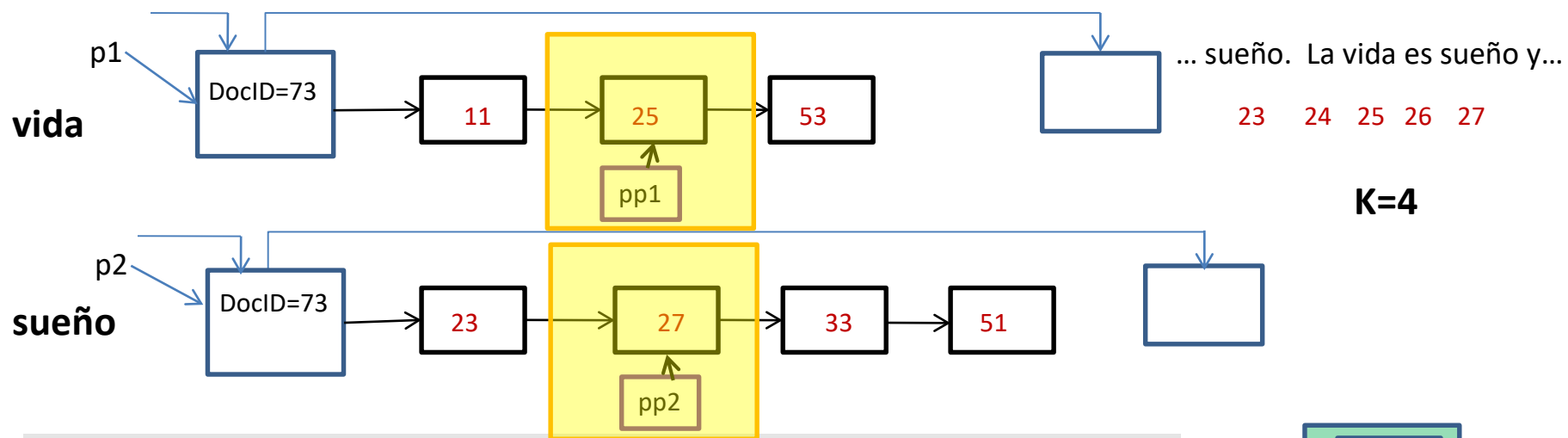
p2 = siguiente (p2)

sino si docID (p1) < docID (p2)

entonces p1 ← siguiente(p1)

sino p2 ← siguiente(p2)





ALGORITMO INTERSECCION_POSICIONAL (p1, p2,k)

respuesta \leftarrow []

mientras No_final(p1) and No_final(p2)

hacer si docID (p1) = docID (p2)

entonces listaux = []

pp1= posición (p1)

pp2= posición (p2)

mientras No_final (pp1)

hacer **mientras** No_final (pp2)

hacer si $|\text{pos}(\text{pp1}) - \text{pos}(\text{pp2})| \leq k$

entonces añadir (listaux, pos(pp2))

sino si $\text{pos}(\text{pp2}) > \text{pos}(\text{pp1})$

then break

pp2 = siguiente (pp2)

mientras (listaux \neq []) and $(|\text{listaux}[0] - \text{pos}(\text{pp1})| > k)$

hacer eliminar (listaux[0])

paratodo ps \in listaux

hacer añadir(respuesta, [docID(p1), pos(pp1), ps])

pp1= siguiente(pp1)

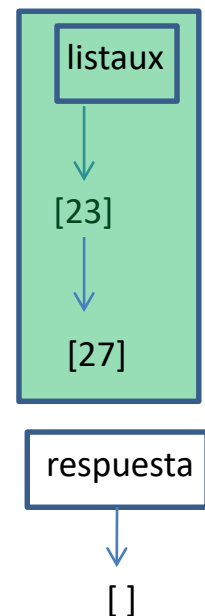
p1 = siguiente (p1)

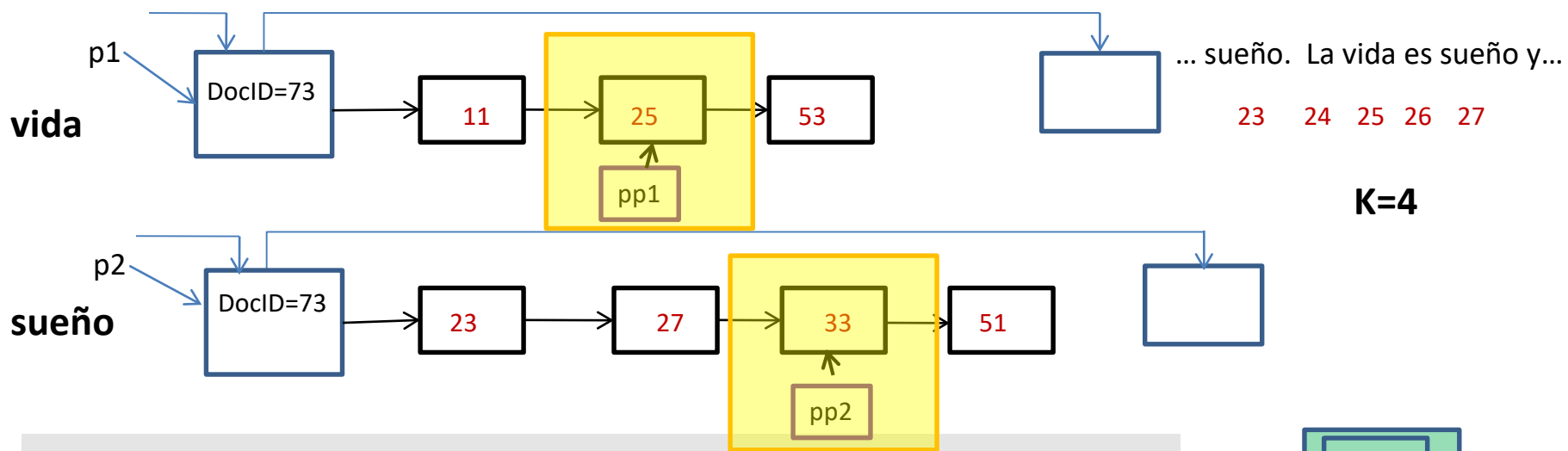
p2 = siguiente (p2)

sino si docID (p1) < docID (p2)

entonces p1 \leftarrow siguiente(p1)

sino p2 \leftarrow siguiente(p2)





ALGORITMO INTERSECCION_POSICIONAL (p1, p2, k)

respuesta \leftarrow []

mientras No_final(p1) and No_final(p2)

hacer si docID (p1) = docID (p2)

entonces listaux = []

pp1= posición (p1)

pp2= posición (p2)

mientras No_final (pp1)

hacer **mientras** No_final (pp2)

hacer si $|\text{pos}(\text{pp1}) - \text{pos}(\text{pp2})| \leq k$

entonces añadir (listaux, pos(pp2))

sino si $\text{pos}(\text{pp2}) > \text{pos}(\text{pp1})$

then break

pp2 = siguiente (pp2)

mientras (listaux \neq []) and $(|\text{listaux}[0] - \text{pos}(\text{pp1})| > k)$

hacer eliminar (listaux[0])

paratodo ps \in listaux

hacer añadir(respuesta, [docID(p1), pos(pp1), ps])

pp1= siguiente(pp1)

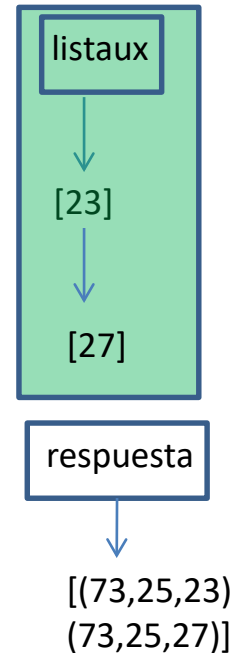
p1 = siguiente (p1)

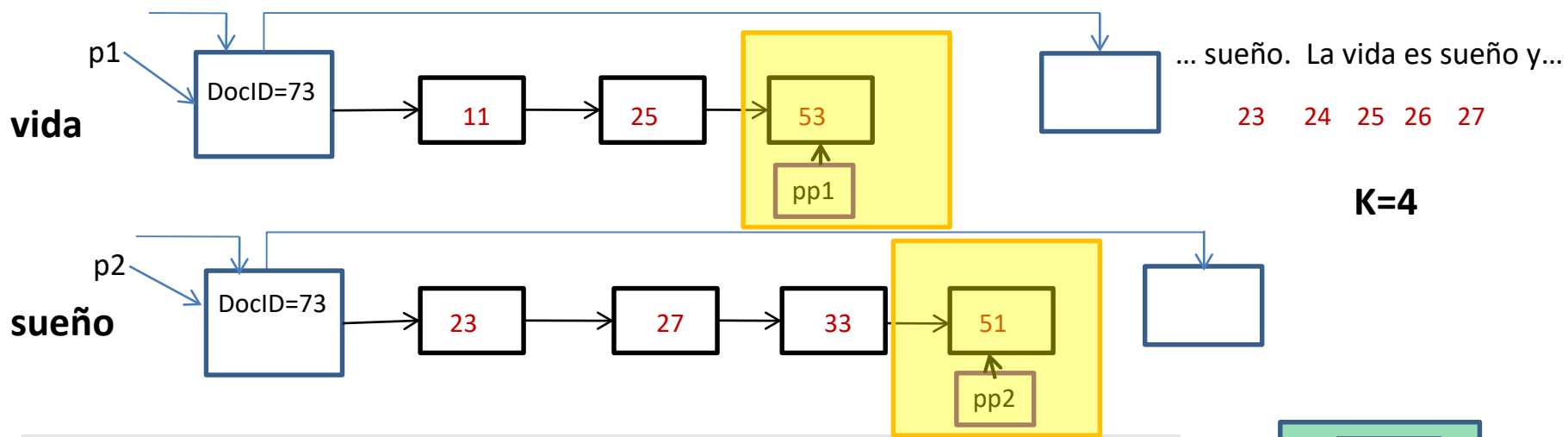
p2 = siguiente (p2)

sino si docID (p1) < docID (p2)

entonces p1 \leftarrow siguiente(p1)

sino p2 \leftarrow siguiente(p2)





ALGORITMO INTERSECCION_POSICIONAL ($p1, p2, k$)

respuesta \leftarrow []

mientras No_final($p1$) and No_final($p2$)

hacer si docID ($p1$) = docID ($p2$)

entonces listaux = []

pp1= posición ($p1$)

pp2= posición ($p2$)

mientras No_final (pp1)

hacer mientras No_final (pp2)

\rightarrow **hacer** si $|\text{pos}(\text{pp1}) - \text{pos}(\text{pp2})| \leq k$

entonces añadir (listaux, pos(pp2))

sino si $\text{pos}(\text{pp2}) > \text{pos}(\text{pp1})$

then break

pp2 = siguiente (pp2)

mientras (listaux \neq []) and ($|\text{listaux}[0] - \text{pos}(\text{pp1})| > k$)

hacer eliminar (listaux[0])

paratodo $ps \in \text{listaux}$

hacer añadir(respuesta, [docID($p1$), pos(pp1), ps])

pp1= siguiente(pp1)

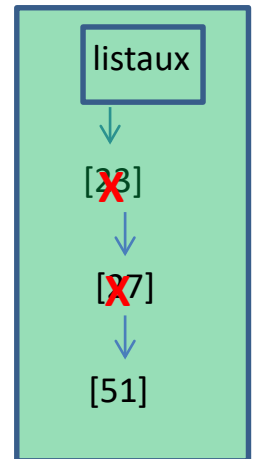
$p1$ = siguiente ($p1$)

$p2$ = siguiente ($p2$)

sino si docID ($p1$) < docID ($p2$)

entonces $p1 \leftarrow$ siguiente($p1$)

sino $p2 \leftarrow$ siguiente($p2$)



respuesta

\downarrow

[(73,25,23)
(73,25,27)
(73,53,51)]