# Capítulo 1.

# Capítulo 2.

# Capítulo 3.

# Capítulo 4.
# Paginación de la Estructura S-D

Si el tamaño de la estructura es tal que no es posible ubicarla en memoria interna, se plantea el problema de su paginación con el fin de minimizar el número de accesos a disco. Los criterios a tener en cuenta son: el tiempo de respuesta a las peticiones y la ocupación. No se consideran los problemas relativos al mantenimiento –inserciones, extracciones y reorganizaciones– debido al carácter estático del diccionario.

El nodo *raíz* se considera una página. En las restantes, se sitúan los nodos de la *parte_árbol*, los de la *parte_cadena* y los *nodos_SIT* según el tipo de paginación utilizado. Para ello, se considera la cantidad de memoria necesaria para almacenar los pares **αf** de cada lista de la *parte_cadena* conjuntamente con la del *nodo_SIT* correspondiente y se organizan de forma secuencial.

# 4.1. Paginación Según el Recorrido.

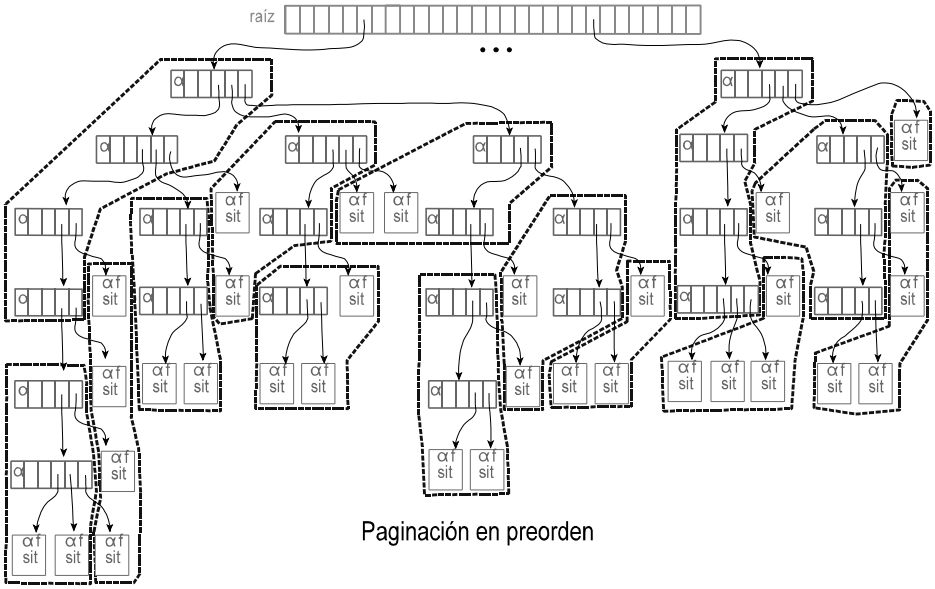


**Figura 25**

Un primer intento para resolver el problema consiste en llenar las páginas con los nodos al recorrer la estructura en *preorden* o en *postorden* –generalización a un árbol multirrama de los recorridos en un árbol binario. La información inscrita en cada página creada de esta forma se refiere a palabras con igual longitud. Las figuras 25 y 26 muestran un bosquejo de ambas paginaciones, suponiendo que todos los nodos ocupan lo mismo y que la capacidad de cada página es de cuatro nodos.

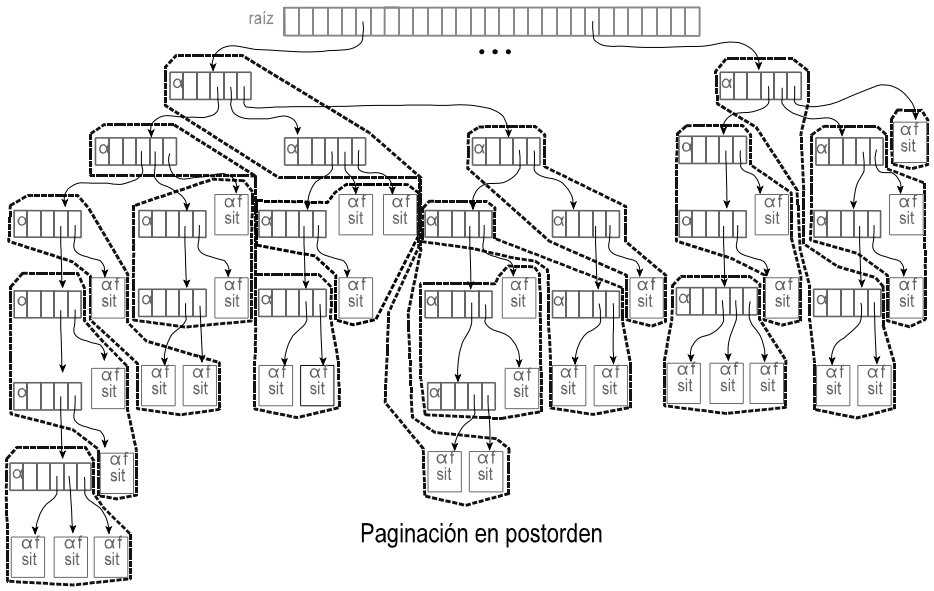


**Figura 26**

Obviamente, ambas paginaciones conducen a un alto porcentaje de ocupación. Sin embargo, presentan la posibilidad de favorecer múltiples accesos a una página –en la que existen ramos no conexos, figuras 25 y 26– al seleccionar las alternativas de descenso, desde el mismo o distintos nodos. Se introduce un buffer con el fin de intentar evitar las lecturas repetidas de una página en memoria secundaria.

Se estudian distintas políticas de buffer: liberar la página visitada con menor frecuencia, **MFU**, –cada página del buffer tiene asociado un valor que indica el número de veces que se ha examinado–; liberar la página menos recientemente usada, **MRU**, –a cada página del buffer se le asigna un contador de orden de utilización–; la primera en entrar es la primera en salir, **PEPS**, y la última en entrar es la primera en salir, **UEPS**.

La política **MFU** tiene posibilidad de bloquear el buffer con las páginas más reutilizadas y proporciona un buen rendimiento cuando existe un gran número de accesos a la misma página. La política **MRU** se adapta

al grado de reutilización de las páginas cuando se explora una determinada longitud de palabras y mantiene el grado de eficacia al cambiar de longitud, en contraposición con la **MFU** que retiene en el buffer páginas de otras longitudes. La política **PEPS** es eficaz para un bajo grado de reutilización y funciona de forma análoga a la **MRU** al cambiar de longitud. La política **UEPS** actúa como si trabajara con un buffer que sólo contiene una página, ya que una vez que se completa, únicamente se puede sustituir la última.

### 4.1.1. Resultados Experimentales.

Se ha realizado una simulación de las dos formas de paginación sobre la estructura **S-D**; se ha medido tanto el número de páginas que se han utilizado como el porcentaje de ocupación de las mismas; los resultados se muestran en la tabla 2. Se observa que el número de páginas y la ocupación de las mismas es prácticamente independiente del tipo de paginación.

| | Preorden | | Postorden | |
|---|---|---|---|---|
| Tamaño de la página | Número de páginas | % ocupación | Número de páginas | % ocupación |
| 1K | 3.143 | 98.50% | 3.144 | 98.47% |
| 2K | 1.569 | 98.66% | 1.569 | 98.66% |
| 4K | 789 | 98.10% | 789 | 98.10% |
| 8K | 400 | 96.75% | 400 | 96.75% |

**Tabla 2**

El nodo *raíz* se mantiene en memoria interna durante todo el proceso de búsqueda.

Sobre cada una de las dos formas de paginación de la estructura se ejecutan las búsquedas de las más similares, con el fin de determinar con qué política de buffer –**MFU**, **MRU**, **PEPS** o **UEPS**– y bajo qué esquema de búsqueda –*creciente* o *decreciente*– se alcanza un mejor rendimiento.

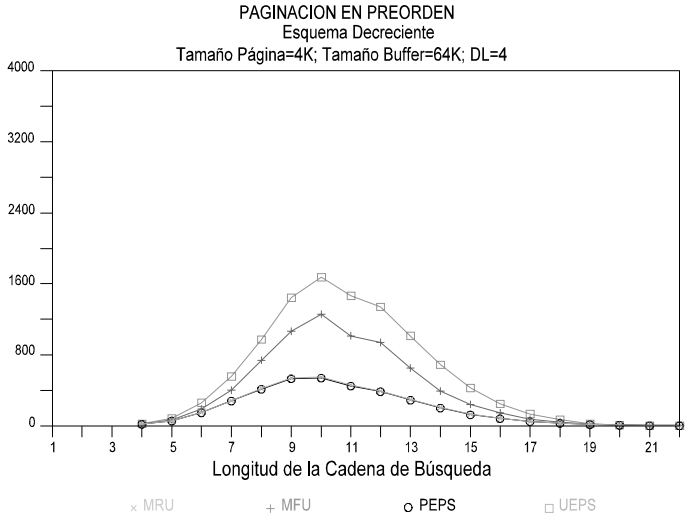


**Figura 27**

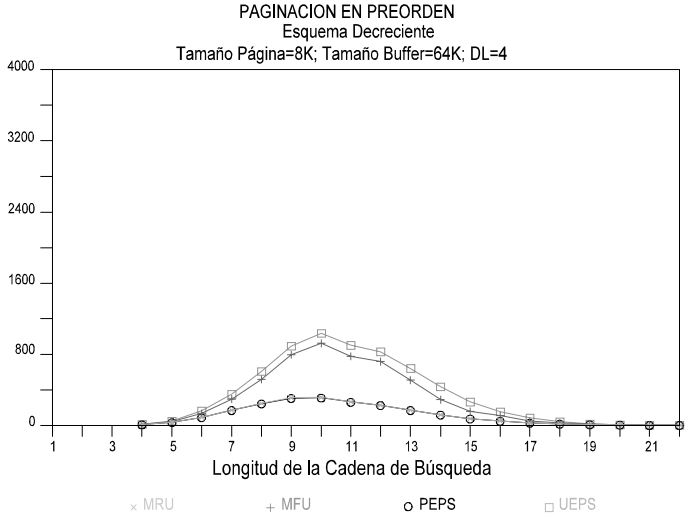


**Figura 28**

☞ Claramente, el *tamaño de la página* está en relación inversa con el número de accesos a memoria externa, cuanto más grande es la página menor es el número de transferencias, figuras 27 y 28.

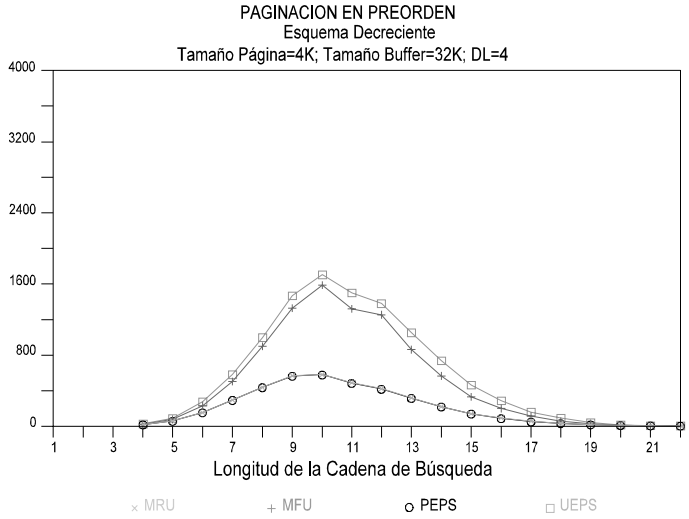


**Figura 29**

☞ Asimismo, el *tamaño del buffer* también está en relación inversa con el número de accesos, aunque se ha observado experimentalmente que su influencia es menor, figuras 27 y 29. La política **MFU** presenta una perdida de eficacia al disminuir el tamaño del buffer.

El *tamaño del buffer* se ha elegido de tal forma que como mínimo sea igual a la longitud máxima del camino medida en páginas. Se consideran como valores más adecuados para el *tamaño de la página* a partir de 4K y para el *tamaño del buffer* superior a 32K.

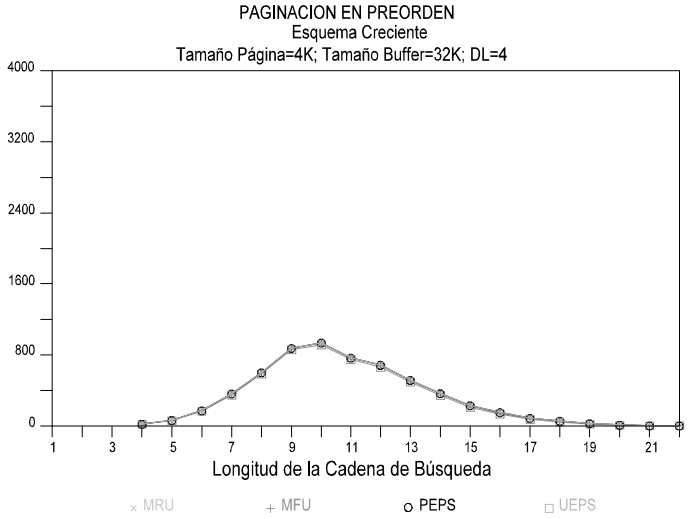


**Figura 30**

☞ Con el incremento del valor de **DL** aumenta la porción de estructura explorada y con ello el número de accesos, figuras 29, 35 y 30, 36.

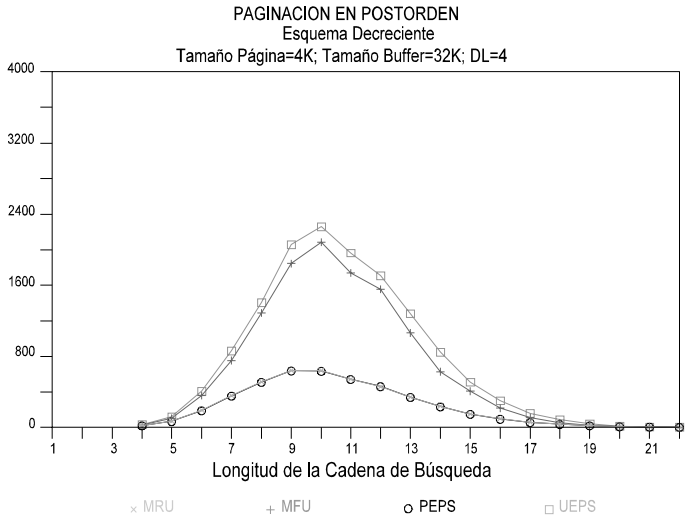


**Figura 31**

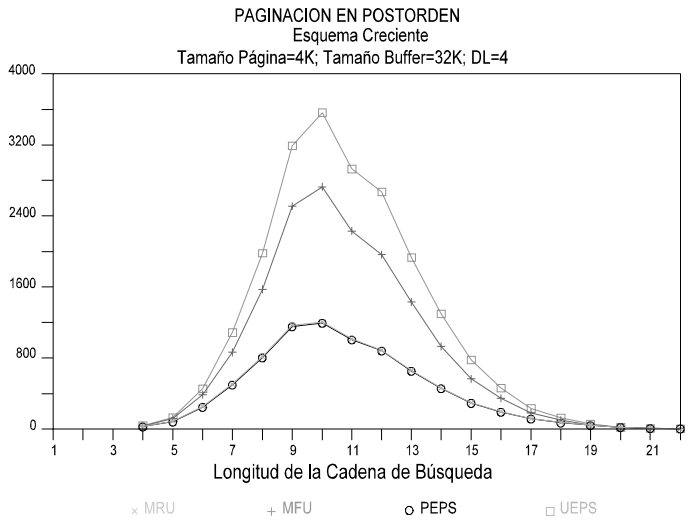


**Figura 32**

☞ Cualquier política de buffer sobre la estructura paginada en *preorden* siempre da lugar a un mejor rendimiento frente a la obtenida en *postorden*, figuras 29, 31 y 30, 32. Esto es una consecuencia de su mejor adaptación al perfil de la estructura –sesgado hacia la izquierda como se muestra en las figuras 33 y 34 para palabras de longitud 4 y 5 respectivamente– y, por tanto, existe una mayor utilización de la información que guarda cada página.

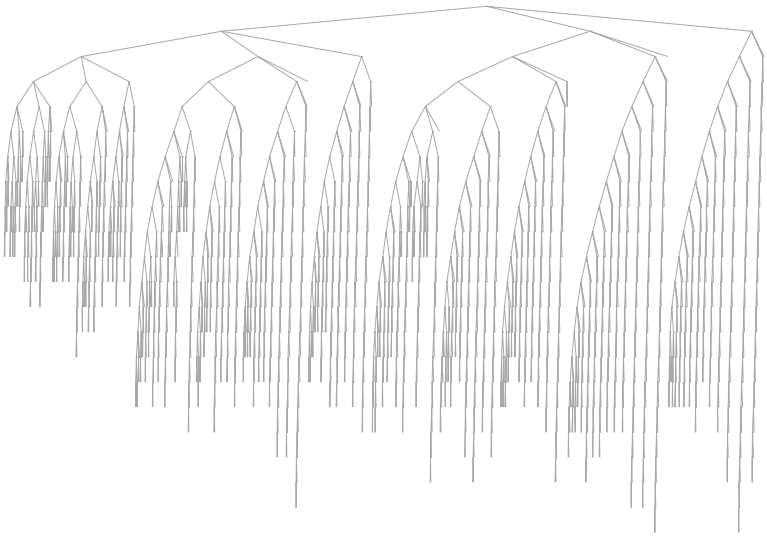

**Figura 33**

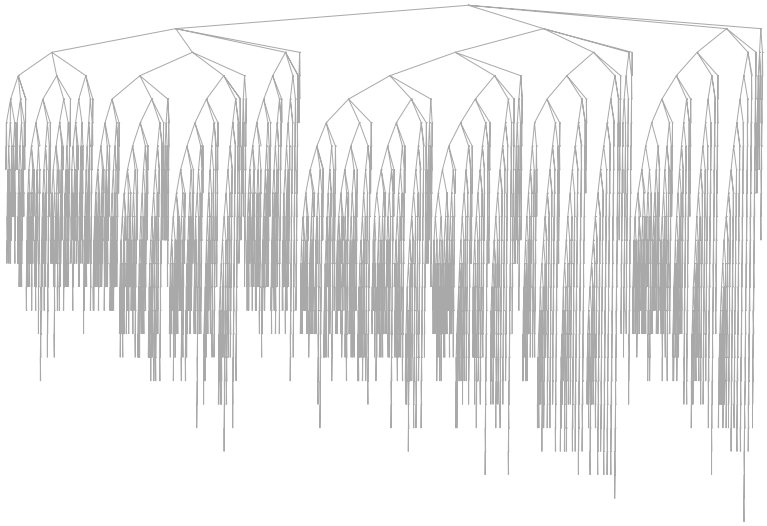

**Figura 34**

☞ Las políticas del buffer que presentan mejores resultados son la **MRU** y la **PEPS** –esta coincidencia indica que la reutilización es baja–; le siguen **MFU** y **UEPS**. La diferencia de **MRU** y **PEPS** con **MFU** y **UEPS** se debe fundamentalmente al bloqueo del buffer de las dos últimas cuando se explora cada longitud y tal diferencia se acentúa al crecer **DL**. Indudablemente, **MFU** es mejor que **UEPS** porque funciona con un tamaño de buffer mayor que una página. Se ha

comprobado experimentalmente que en el esquema *decreciente* **MFU** y **UEPS** mejoran el rendimiento cuando se libera el buffer al cambiar de longitud, aunque no lo suficiente como para alcanzar a las otras dos políticas. En el esquema *creciente*, el uso de esta opción empeora los resultados.

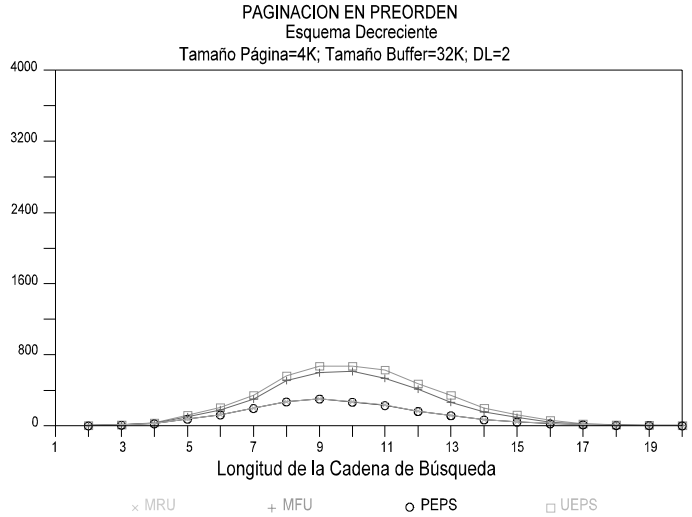


**Figura 35**

☞ La búsqueda con el esquema *creciente* sobre la estructura paginada en *preorden* presenta una excepción respecto al punto anterior. El número de accesos es independiente de la política del buffer, figuras 30 y 36. La selección de alternativas del esquema *creciente* y la paginación en *preorden* determinan que, en cada fase, sólo se reutiliza la última página accedida, por tanto, no tiene incidencia significativa la política de desalojo de páginas. En cada nueva fase se exploran al menos las mismas páginas que en la anterior, la única política que tiene posibilidad de proporcionar algún beneficio en esta situación es la **UEPS** –mantiene el camino inicial de una fase a

otra–; sin embargo no es significativo tal y como muestran los resultados.

☞ En la búsqueda sobre la estructura paginada en *postorden* se obtienen siempre mejores resultados para el esquema *decreciente* que para el *creciente* cuando *DL>2*, independientemente de la política, figuras 31 y 32. El número de accesos del esquema *creciente* se incrementa con el número de fases que coincide con el valor de la **DL** entre la cadena de búsqueda y sus más similares. Para *tamaño de página* 4K y *DL=2*, el comportamiento de ambos esquemas con **MRU** y **PEPS** es prácticamente el mismo.

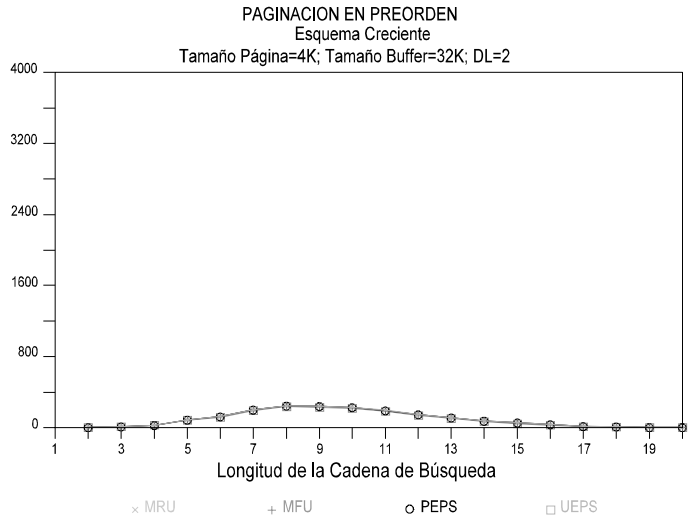


**Figura 36**

☞ Se ha obtenido que en la búsqueda sobre la estructura paginada en *preorden* el esquema *decreciente* con **MRU** y **PEPS** es mejor que el *creciente* con cualquier política para *DL>2*, figuras 29 y 30. Las políticas **MRU** y **PEPS** mantienen en el buffer páginas utilizadas por el esquema *decreciente* y

que son posteriormente visitadas en función del recorrido en vaivén de este esquema.

- ☞ Para *DL ≤2* y longitudes intermedias de la palabra de búsqueda, el esquema *creciente*, para todas las políticas del buffer, es ligeramente mejor que el *decreciente* con **MRU** y **PEPS**, figuras 35 y 36, –el *creciente* realiza como mucho dos fases y la porción de estructura implicada es menor. Se ha comprobado que la mejoría disminuye con el aumento del *tamaño de la página* –para un *tamaño de página* 8K y *DL=2* coinciden.

Como conclusión se obtiene que el menor número de accesos se alcanza con el esquema de búsqueda *decreciente*, excepto para *DL=1*, sobre la estructura paginada en *preorden* y con las políticas **MRU** y **PEPS**. Se selecciona **PEPS** por ser menos costosa en tiempo de ejecución que **MRU**, ya que ésta última necesita controlar el orden de utilización.

## 4.2. Paginación Atendiendo al Esquema de Búsqueda Decreciente.

Se propone una nueva forma de paginar la estructura, *segunbusca*, de tal modo que se respete el siguiente principio: durante la búsqueda, una vez que se abandone una página no se vuelve a acceder a ella por otro camino. Esta condición da lugar a que se utilice solamente una pila cuyo

tamaño no excederá la longitud en páginas del camino más largo, para mantener en memoria las páginas necesarias.

Es necesario tener construida la estructura **S-D** antes de realizar este proceso de paginación.

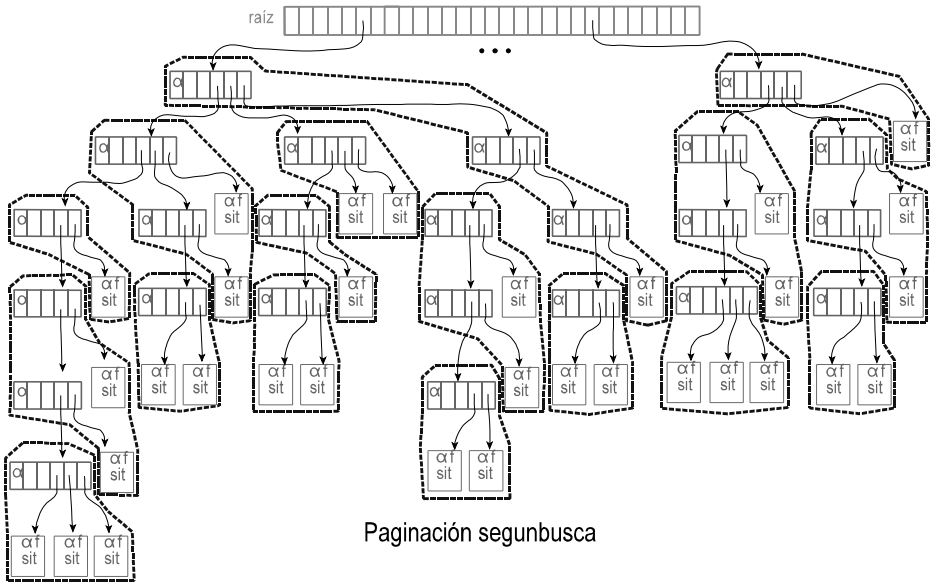


**Figura 37**

Esta paginación, figura 37, sigue un proceso recursivo tal que en cada *nodo* se comprueba si todos sus hijos y él mismo caben en una página, en cuyo caso, se traslada la decisión de completar la página al *padre de nodo.* En caso contrario, todos los hijos que quepan, empezando por los de menor ocupación, se agrupan en una página junto con *nodo* y se traslada la decisión de completar tal página al *padre de nodo*; el resto de los hijos de *nodo* –los de mayor ocupación– se guardan en páginas separadas. Consecuentemente, se produce un descenso del porcentaje de ocupación. Nótese que los hijos de *nodo* mencionados incluyen a sus descendientes aunque no tienen por qué ser los subárboles completos, sino que pueden ser restos de subárboles que han quedado sin paginar.

*Algoritmo de construcción de la estructura S-D con la paginación segunbusca:*

Procedimiento Construye_S-Dpag (raiz)
raiz(.): contiene los punteros que señalan a la parte_árbol asociada a cada longitud
**para** long=longmin **hasta** longmax **hacer**
**si** Ocupa_parte_arbol(raiz(long)) > 0 **entonces**
situar raiz(long) y sus descendientes no paginados en una nueva página
almacenar la página
**fin si**
**fin para**

La función Ocupa_parte_arbol invoca a las siguientes funciones:

Ocupa(nodo): Devuelve la ocupación en bytes de un nodo de la parte_árbol.

Ocupa_nodo_SIT(nodo): Devuelve la ocupación en bytes de un nodo_SIT.

Ocupa_parte_cadena(nodo): Calcula la ocupación en bytes de los pares αf a partir del primer nodo de la parte cadena, nodo, acumulada con la del correspondiente nodo_SIT.

```
Función Ocupa_parte_arbol (nodo)
{Realiza la paginación de los hijos de nodo y devuelve la ocupación residual}
si nodo = nulo entonces
    devolver (0)
si no
    si nodo es un nodo_SIT entonces
        devolver (Ocupa_nodo_SIT(nodo))
    si no
        si nodo es un nodo de la parte_cadena entonces
            devolver (Ocupa_parte_cadena(nodo))
        si no
            ocupacion=Ocupa(nodo)
            para i=nodo.pp hasta nodo.pu hacer
                ocupahijo(1,i)=i
                ocupahijo(2,i)=Ocupa_parte_arbol(nodo.e(i))
                ocupacion=ocupacion+ocupahijo(2,i)
            fin para
            si ocupacion ≤ tamaño_pagina entonces
                devolver (ocupacion)
            si no
                Ordena(ocupahijo)
                {ordena ocupahijo en valores crecientes de su 2ª fila}
                i=nodo.pu
                repetir
                    ocupacion=ocupacion-ocupahijo(2,i)
                    situar nodo.e(ocupahijo(1,i)) y sus descendientes no paginados en
                    una nueva página
                    almacenar la página
                hasta que ocupacion < tamaño_pagina
                devolver (ocupacion)
            fin si
        fin si
    fin si
fin si
```

### 4.2.1. Resultados Experimentales.

Se ha realizado una simulación sobre la estructura **S-D** de la paginación propuesta, *segunbusca*, los resultados del número de páginas necesarias y su porcentaje de ocupación se muestran en la tabla 3.

| | Preorden | | Segunbusca | |
|---|---|---|---|---|
| Tamaño de la página | Número de páginas | % ocupación | Número de páginas | % ocupación |
| 4K | 789 | 98.10% | 1.044 | 74.10% |
| 8K | 400 | 96.75% | 535 | 72.30% |

**Tabla 3**

☞ Como se esperaba, el número de páginas que requiere el almacenamiento de la estructura ha aumentado con respecto a la paginación en *preorden* dando lugar a una pérdida en el porcentaje de ocupación en torno al 24%.

Se ha medido el número de accesos que requiere el esquema de búsqueda *decreciente* sobre la estructura con la nueva paginación, *segunbusca*, y se compara con los resultados obtenidos para el mismo esquema de búsqueda sobre la estructura paginada en *preorden* con la política **PEPS**.

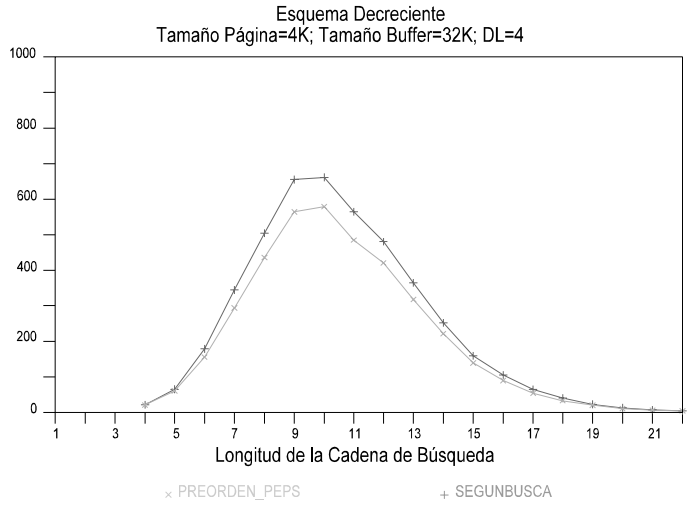


**Figura 38**

☞ Se ha obtenido que independientemente del valor de **DL** y del *tamaño de la página*, la paginación *segunbusca* no supera el rendimiento alcanzado por la paginación en *preorden*, figuras 38 y 39. El descenso del porcentaje de ocupación conduce a un mayor número de accesos.

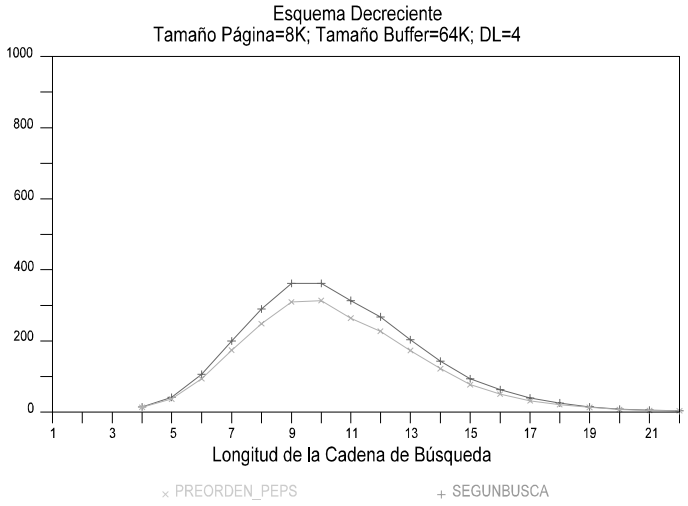


**Figura 39**

Como conclusión se obtiene que la paginación en *preorden* es la más adecuada.