Algoritmo de Siklóssy

Definición

Basado en conceptos vertidos anteriormente sobre estructuras lineales utilizando doble linkeo, es válido pensar que el espacio utilizado para la representación de una cola con linkeo doble es mayor al utilizado por una cola de linkeo simple, aunque el costo que se asume es el de no poder recorrer la cola en ambos sentidos. El problema que se presenta nuevamente es asumir el costo que representa un mayor tiempo de acceso o un mayor espacio utilizado. El algoritmo de Sikloosy presenta la posibilidad de minimizar el espacio ocupado, utilizando un único campo link, sin aumentar el tiempo de acceso, mediante el concepto algebraico del or exclusivo.

Ejemplo :

p	q	p+q
0	0	0
0	1	1
1	0	1
1	1	0

De donde:
(p + q) + p = q
(p + q) + q = p

En el ejemplo anterior

p	q	(p+q) (1)	(1)+p	(1)+q
0	0	0	0	0
0	1	1	1	0
1	0	1	0	1
1	1	0	1	1

Ejemplo : Sea p = 001101 y q = 100111

p	q	(p+q)	(p+q)+p	(p+q)+q
001101	100111	101010	100111	001101

Desarrollo del algoritmo : En el ejemplo de la Figura 6.1., se analiza una cola de doble linkeo y links con valores binarios .

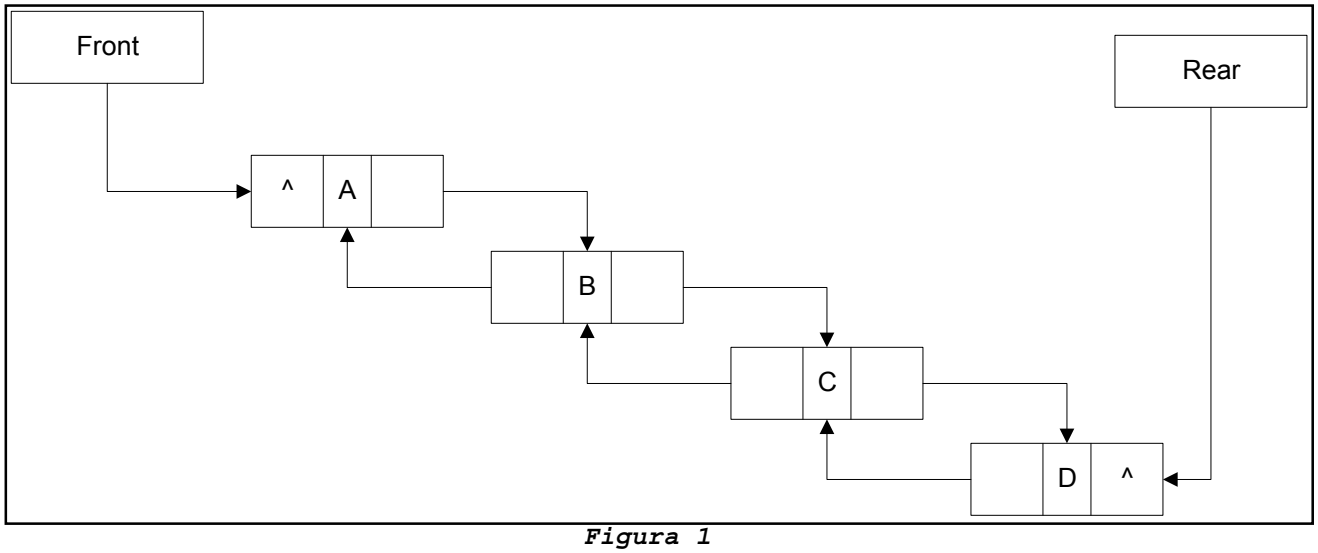

Figura 1

Suponiendo que los nodos ocupan las siguientes direcciones binarias :

dir A : 0101
dir B : 1000
dir C : 0111
dir D : 1100

se reemplazan en la Figura 1 los punteros con el valor de las direcciones resultando la Figura 2.

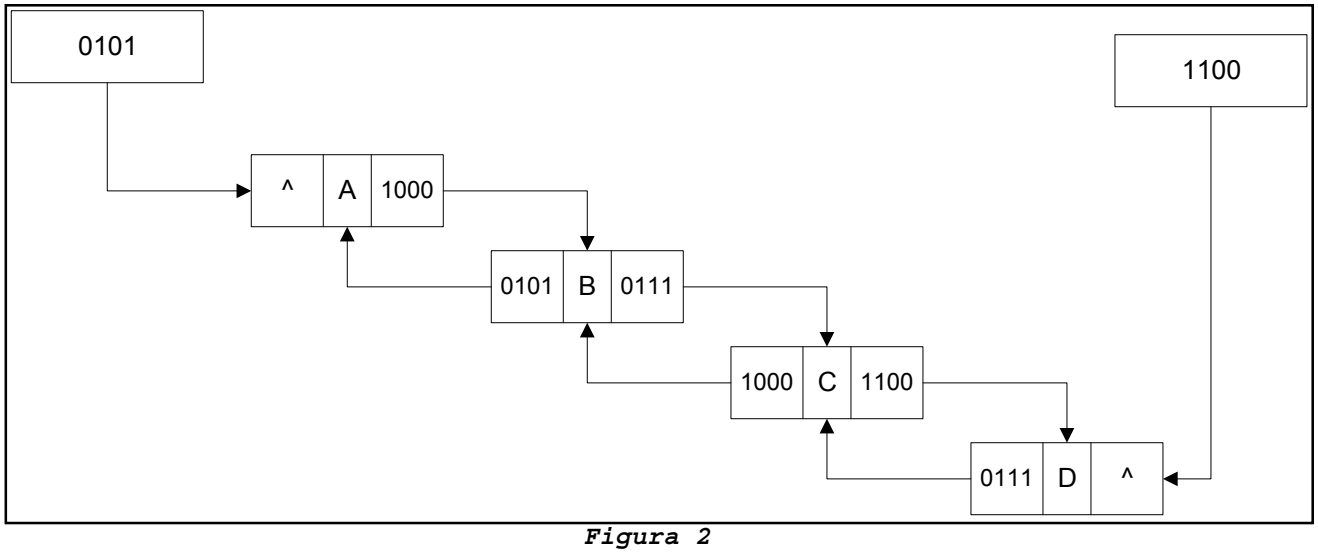

Figura 2

Para evitar el doble linkeo, Sikloosy propone almacenar en un mismo campo de dirección un valor que permita recorrer la estructura en ambas sentidos basándose en la suma del or exclusivo. En el siguiente ejemplo, se desarrolla básicamente la idea del algoritmo.

Ejemplo :

Si un nodo de la estructura lineal, ocupa la posición i, por ejemplo, deber tener un campo tipo pointer con un valor que permita conocer la posición del nodo anterior de la estructura (posición i-1) y la dirección del nodo posterior (posición i+1).

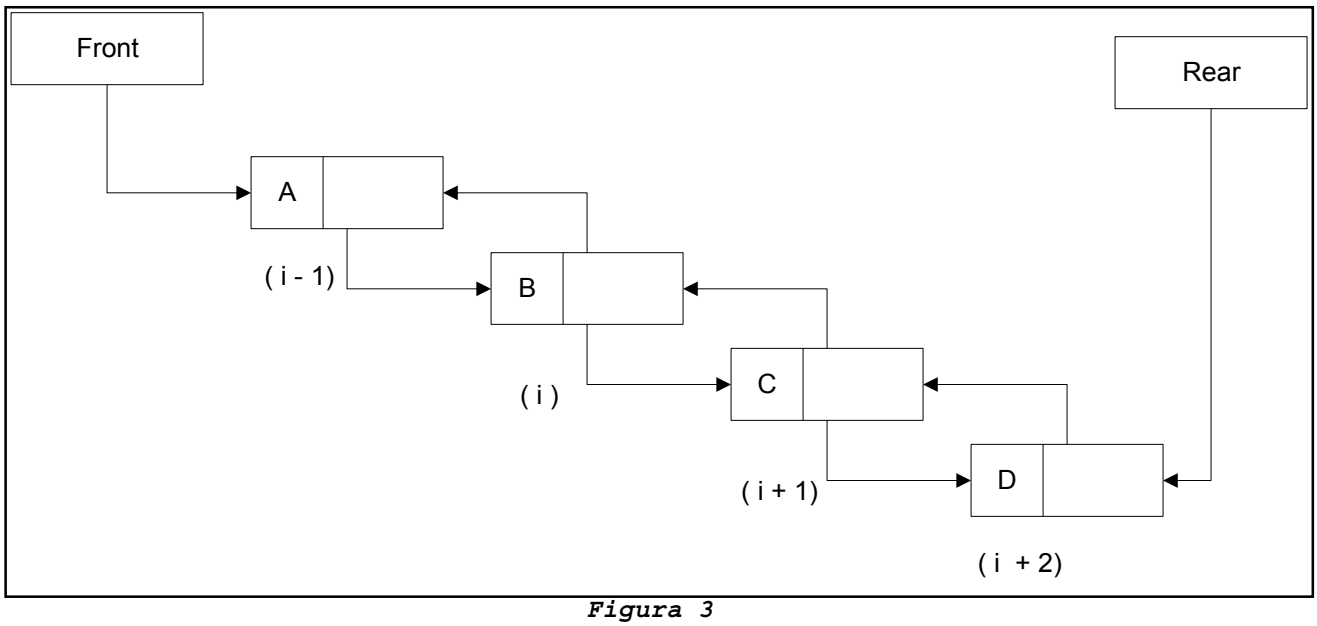

Figura 3

Si el nodo B, ocupa la posición i, es posible identificar a p como la dirección de memoria que contiene el nodo de posición (i - 1), es decir pX = dir(i - 1), donde X es el nodo en consideración, en este caso X = B. De la misma manera se identifica a q como la dirección de memoria que contiene el nodo de posición (i + 1), es decir qX = dir(i + 1).

Por lo tanto p se identifica con la dirección del nodo anterior y q con la dirección del nodo posterior. En el ejemplo,

pB = dir(A) y qB = dir(C)

Si se analiza la posibilidad de ir desde el nodo B al nodo anterior o posterior, sabiendo que (p + q) + q = p y que (p + q) + p = q, se deduce que, conociendo la dirección del nodo anterior al que sé est analizando, (en el ejemplo la dirección (i-1) que fue identificada como p y que corresponde a la dirección del nodo A), y realizando la suma con el valor (p + q) (que deber estar guardado en el campo dirección del nodo que se est analizando), es posible obtener la dirección del nodo posterior al que se est analizando,(en el ejemplo la dirección (i+1) que fue identificada como q y que corresponde a la dirección del nodo C).

Para poder implementar este algoritmo, deber tenerse en cuenta que la dirección del nodo anterior a un minimal y la dirección del nodo posterior a un maximal toman el valor cero binario. Es decir :
p min = 0000 y q max = 0000
En el ejemplo, pA = 0000 y qD = 0000

Alta de nodos

El primer elemento a dar de alta en la estructura del ejemplo es el nodo A. Las condiciones iniciales son : Front = Rear = 0000. Se pide una dirección al administrador de memoria, en el ejemplo devuelve la dirección 0101. Esta dirección de memoria ser ocupada por el registro que contiene el campo dato con el valor A y el campo dirección con valor 0000 pues este registro es maximal y minimal simultáneamente (pA = qA = 0000). Por último se actualizan los punteros externos a la estructura Front y Rear. La estructura queda de la siguiente manera :

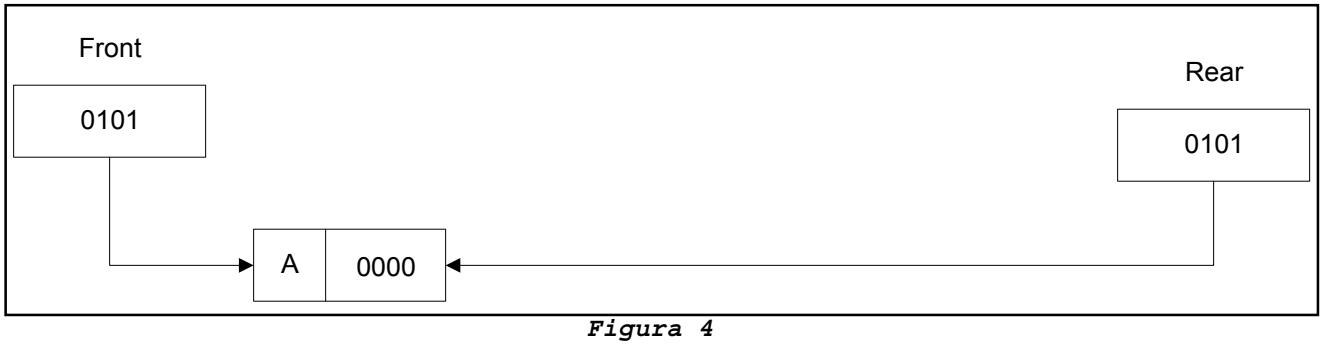

Figura 4

El siguiente nodo a dar de alta es el nodo B. El administrador de memoria devuelve la dirección 1000 como libre. Esta dirección será ocupada por el registro con el campo dato B y un campo dirección cuyo valor resultara de la suma de la dirección del nodo anterior con la dirección del nodo posterior. Como B es el último registro ingresado, la dirección posterior corresponde al qmax es decir 0000. La dirección del nodo anterior puede obtenerse del puntero externo Rear (antes de que sea actualizado por el ingreso del nuevo nodo), pues como B acaba de ser dado de alta y Rear apunta siempre al último registro ingresado, puede asegurarse que en esta variable se guarda el valor del nodo anterior al recientemente ingresado. Por lo tanto :

pB = 0101, qB = 0000 => pB + qB = 0101

Este valor, 0101, es el valor que toma el campo dirección del registro que guarda el campo dato B. Como el nodo A, tiene un nodo posterior, deber recalcularse su campo dirección. Por lo tanto :

pA = 0000, qA = 1000 => pA + qA = 1000

La estructura queda entonces de la siguiente manera :

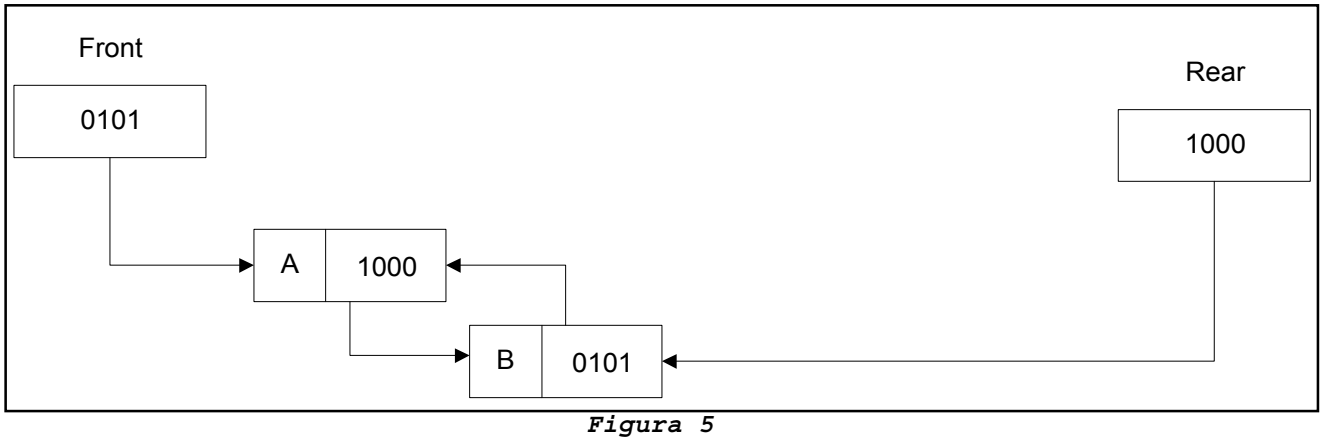

Figura 5

Dando sucesivamente de alta los nodos C y D de la forma considerada, la estructura queda de la siguiente manera :

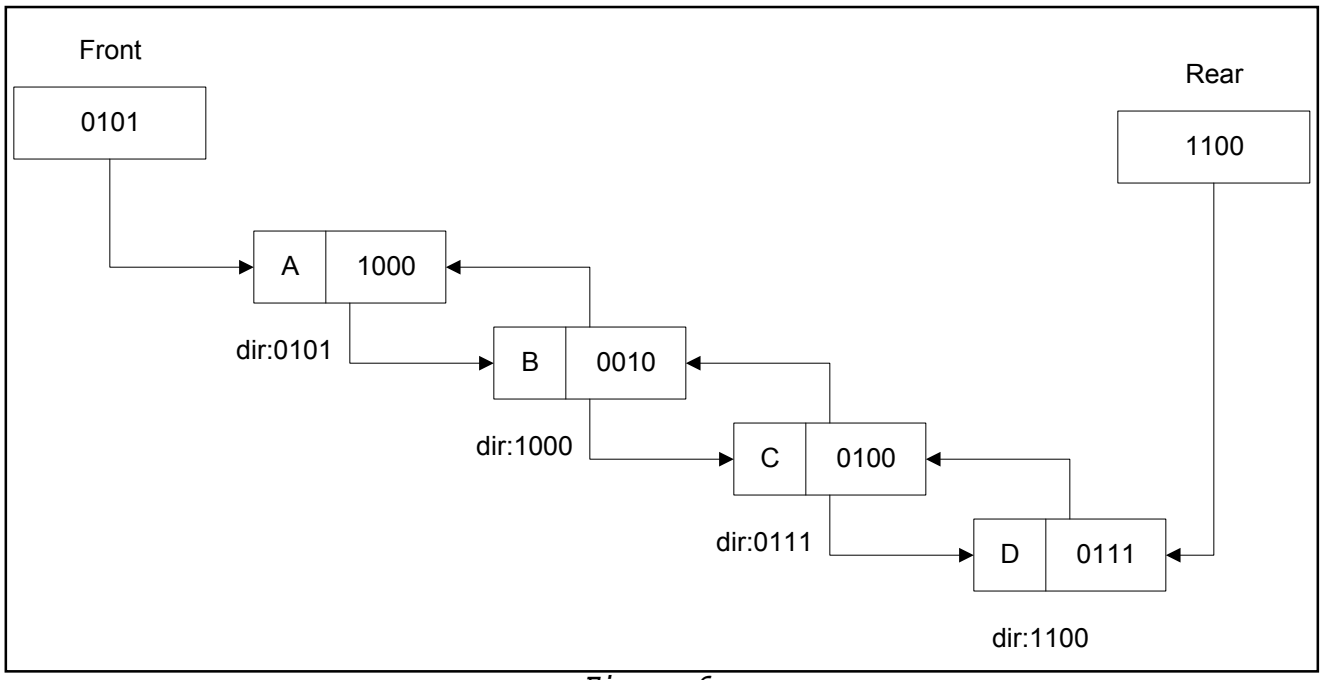

Figura 6

Pseudocódigo del Algoritmo de Sikloosy

Procedure Altas

```
REAR := 0000
begin
   read(DATO)
   while DATO <> '0' do
   begin
      new(DIR)
      if   REAR <> 0000
           REAR^.LINK := REAR^.LINK + DIR
      else
           FRONT := DIR
      endif
      DIR^.CARACTER := DATO
      DIR^.LINK     := REAR
      REAR          := DIR
      read(DATO)
   end
end.
```

Procedure Recorre_de_atras

```
p := REAR;
q := 0000;
writeln(p^.CARACTER);
while p <> FRONT do
begin
   x := p;
   p := p^.link + q;
   writeln(p^.CARACTER);
   q := x;
end.
```