(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 830 746**

(51) Int. CI.:

G06F 16/901 (2009.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **29.01.2016 PCT/JP2016/052664**

(87) Fecha y número de publicación internacional: **15.09.2016 WO16143405**

(96) Fecha de presentación y número de la solicitud europea: **29.01.2016 E 16761377 (7)**

(97) Fecha y número de publicación de la concesión europea: **14.10.2020 EP 3270551**

(54) Título: **Dispositivo de recuperación, método de recuperación, programa y medio de registro**

(30) Prioridad:

11.03.2015 JP 2015048657

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
04.06.2021

(73) Titular/es:

NTT COMMUNICATIONS CORPORATION (100.0%)
1-6, Uchisaiwai-cho 1-chome
Chiyoda-kuTokyo 100-8019, JP

(72) Inventor/es:

ASAI, HIROCHIKA y
OHARA, YASUHIRO

(74) Agente/Representante:

CARVAJAL Y URQUIJO, Isabel

DESCRIPCIÓN

Dispositivo de recuperación, método de recuperación, programa y medio de registro

Campo técnico

La presente invención se refiere a una técnica de búsqueda para obtener datos deseados al buscar datos de objetivo de búsqueda representados como una estructura de árbol.

Antecedente de la técnica

En un aparato tal como un enrutador y similares, el procesamiento se realiza para determinar un destino de transferencia de un paquete al buscar en una tabla de enrutamiento basada en una dirección de destino de un paquete recibido. En el procesamiento, se realiza la búsqueda de coincidencia más larga. Para ese fin, en la técnica convencional, se utilizó Patricia trie (Trie), Radix tree (árbol raíz) y similares. En la técnica convencional, las técnicas de árbol binario son la corriente principal, y el rendimiento es a lo sumo de unos pocos Mlps (Mega Consulta por segundo). También se inventaron técnicas de árbol de múltiples vías (N-ary/Multivías), pero esas no eran la corriente principal en el uso práctico. Dado que estos rendimientos no son deseables, el hardware llamado TCAM que realiza cientos de Mlps es el estándar de facto. TCAM tiene dificultades en economía, densidad, escala, consumo de energía y generación de calor.

Con el fin de superar el problema de TCAM, ha aparecido recientemente una técnica para buscar una ruta al combinar un dispositivo y software disponibles comercialmente. PacketShader, GPU Click, GAMT y similares realizan una búsqueda de ruta de alto rendimiento utilizando GPU, pero debido a que utiliza GPU, tiene el problema de calor y así sucesivamente TCAM y similares como se conoce del documento JP2000-083054.

El documento US2012/239664A1 divulga un aparato de búsqueda que utiliza una estructura de árbol de múltiples vías.

Resumen de la invención

Problema a resolver por la invención

Como se mencionó anteriormente, dado que utilizar un dispositivo específico tal como el TCAM o GPU o similares tiene problemas como la generación de calor, no es preferible acelerar la búsqueda de ruta al utilizar el dispositivo específico.

Se han propuesto técnicas (ejemplo: DXR, SAIL) para acelerar la búsqueda de rutas por software con hardware de propósito general (ejemplo: CPU disponible comercialmente, etc.) sin presuponer el uso del hardware específico. Sin embargo, las técnicas tienen el problema de que el rendimiento se degrada cuando el número de rutas en la tabla de enrutamiento se hace grande o cuando la longitud de la dirección es larga.

En el procesamiento de búsqueda que utiliza hardware de propósito general, se produce el problema de que se deteriora el rendimiento de la búsqueda cuando la escala de datos de los datos de objetivo de búsqueda se hace grande o cuando la longitud de los datos clave se hace larga, no se limita a la búsqueda de ruta.

La presente invención está ideada en vista de los puntos mencionados anteriormente, y un objeto de la misma es proporcionar una técnica que permita la búsqueda a alta velocidad de datos de objetivo de búsqueda representados por una estructura de árbol incluso cuando se utiliza hardware de propósito general.

Medios para resolver el problema

De acuerdo con un primer aspecto de la presente invención, se proporciona un aparato de búsqueda que comprende las características de la reivindicación 1.

De acuerdo con un segundo aspecto de la presente invención, se proporciona un método de búsqueda que comprende las características de la reivindicación 15.

Efecto de la presente invención

De acuerdo con una realización de la presente invención, es posible buscar datos de objetivo de búsqueda, a alta velocidad, representados por una estructura de árbol incluso cuando se utiliza hardware de propósito general.

Breve descripción de los dibujos

La Fig. 1 es un diagrama para explicar un método de búsqueda raíz de múltiples vías;

La Fig. 2 es un diagrama de bloques de un aparato 10 de búsqueda en una realización de la presente invención;

La Fig. 3 es un diagrama que muestra un aparato 20 de búsqueda en una realización de la presente invención;

La Fig. 4 es un diagrama que muestra un ejemplo de datos de objetivo de búsqueda almacenados en una unidad 12 de almacenamiento;

La Fig. 5 es un diagrama para explicar una estructura de datos de objetivo de búsqueda y un esquema de procesamiento de búsqueda en la presente realización;

La Fig. 6 es un diagrama que muestra un ejemplo más concreto de un nodo interno y un nodo de hoja;

La Fig. 7 es un diagrama de flujo para explicar un procedimiento del procesamiento de búsqueda;

La Fig. 8A es un diagrama para explicar un señalamiento directo;

La Fig. 8B es un diagrama para explicar un señalamiento directo;

La Fig. 9 es un diagrama para explicar un ejemplo de compresión de datos de un nodo de hoja;

La Fig. 10 es un diagrama para explicar un ejemplo de una estructura de datos en un ejemplo de compresión;

La Fig. 11 es un diagrama para explicar un ejemplo de compresión de datos de un nodo de hoja;

La Fig. 12 es un diagrama para explicar un ejemplo de una estructura de datos en un ejemplo de compresión;

La Fig. 13 es un diagrama para explicar un ejemplo de compresión de datos de un nodo interno;

La Fig. 14 es un diagrama que muestra una estructura de datos de un nodo interno cuando se aplica una máscara de hoja;

La Fig. 15 es un diagrama de flujo del procesamiento para obtener un valor de una hoja al utilizar una máscara de hoja;

La Fig. 16 es un diagrama para explicar un método de generación de datos sobre una máscara de hoja.

Realizaciones para la realización de la invención

A continuación, se describe una realización de la presente invención con referencia a las Figuras. Nótese que las realizaciones descritas a continuación son simplemente ejemplos, y las realizaciones a las que se aplica la presente invención no se limitan a las siguientes realizaciones.

(En cuanto al método de búsqueda)

En la presente realización, como ejemplo de un destino de aplicación de la técnica de búsqueda de la presente invención, se asume un procesamiento en el que un enrutador busca una tabla de enrutamiento (más específicamente, una tabla de reenvío) con la coincidencia más larga al utilizar una dirección de destino de un paquete recibido como clave, adquiriendo de esta manera la información de un siguiente salto como destino de transferencia del paquete. Sin embargo, la aplicación de la presente invención no se limita a esto, y la presente invención no se limita a la coincidencia más larga, sino que se puede aplicar a varios tipos de búsquedas, como la coincidencia perfecta. En lo sucesivo, los datos que se van a buscar (búsqueda) se denominan datos de objetivo de búsqueda y los datos que sirven como clave de búsqueda, tales como una dirección de destino, se denominan datos clave.

En la presente realización, como método de búsqueda de datos de objetivo de búsqueda, se utiliza un método de búsqueda raíz de múltiples vías representado por un árbol de múltiples vías. Por lo tanto, en primer lugar, se describe un esquema de un método de búsqueda raíz de múltiples vías con referencia a la Fig. 1.

En el método de búsqueda raíz de múltiples vías, los datos clave se dividen en un número predeterminado de varios bits (en lo sucesivo, un fragmento) desde la cabeza, de modo que la transición de un árbol se realiza en toda la pluralidad de bits. La Fig. 1 es un ejemplo en el que cada fragmento consta de dos bits. Dado que cada fragmento puede tomar cuatro tipos de valores (representados como a, b, c, d en la Fig. 1), cada nodo del árbol se ramifica en cuatro direcciones. El destino de la ramificación es un nodo interno (un nodo indicado por un círculo en la Fig. 1) o un nodo de hoja (un nodo indicado por un cuadrado en la Fig. 1).

Se inicia una búsqueda en un nodo de la primera etapa desde el primer fragmento de los datos clave, y se ramifica en un nodo hijo de un valor correspondiente, luego la clave avanza al siguiente fragmento, de modo que la búsqueda se realiza secuencialmente. Cuando llega a un nodo de hoja, la búsqueda finaliza.

En el ejemplo de la Fig. 1, por ejemplo, cuando el dato clave es dxxxx (x es un valor arbitrario), llega al nodo de hoja indicado por 5. Cuando el dato clave es bbxx, llega al nodo de hoja indicado por 6. Por ejemplo, la información que indica un próximo salto (ejemplo: dirección, información IF, etc.) se almacena en un nodo de hoja. Cuando llega a un nodo de hoja, se puede adquirir información sobre el siguiente salto correspondiente a los datos clave.

El ejemplo anterior es un ejemplo en el que la longitud del fragmento es de 2 bits. Sin embargo, por ejemplo, cuando se utiliza una arquitectura de CPU de 64 bits, para que el cálculo sea eficiente al utilizar el mismo ancho de bits, la longitud del fragmento se establece en 6 bits, de modo que no se puede utilizar una estructura de datos en la que hay 64 ramificaciones en cada nodo.

En el método de búsqueda raíz de múltiples vías descrito anteriormente, generalmente, cada nodo tiene indicadores (cada uno es una dirección para almacenar un nodo hijo), cuyo número es el número de ramificaciones, cada una de las cuales es para señalar un nodo hijo. Sin embargo, dado que cada indicador designa un nodo hijo utilizando 64 bits, por ejemplo, existe el problema de que la cantidad total de datos se vuelve muy grande. Por lo tanto, de acuerdo con la configuración que utiliza indicadores como se describió anteriormente, subsiste el problema de que los datos del árbol no se pueden almacenar en una caché de la CPU de propósito general y los datos se deben almacenar en una memoria fuera de la CPU, de modo que se deteriora la velocidad de búsqueda.

Por otro lado, de acuerdo con la técnica de la presente realización, en comparación con la técnica mencionada anteriormente, la cantidad de datos de cada nodo interno se puede reducir en gran medida, y los nodos que tienen los mismos datos se pueden comprimir, de modo que la cantidad de datos del árbol se puede reducir y es posible realizar el procesamiento al almacenar los datos de un árbol en una caché de una CPU de propósito general. Por lo tanto, incluso cuando se utiliza hardware de propósito general, como una CPU de propósito general, el procesamiento de búsqueda de alta velocidad es posible. A continuación, se describe con más detalle la técnica de acuerdo con esta realización.

(Ejemplo de configuración de aparato)

En primer lugar, se describe un ejemplo de configuración de un aparato de búsqueda que ejecuta el procesamiento de búsqueda de la presente realización. La Fig. 2 es un diagrama que muestra un ejemplo de configuración de un aparato 10 de búsqueda de la presente realización.

Como se muestra en la Fig. 2, el aparato 10 de búsqueda incluye una unidad 11 de cálculo, una unidad 12 de almacenamiento, una unidad 13 de entrada y una unidad 14 de salida. La unidad 11 de cálculo es una unidad funcional configurada para ejecutar el procesamiento de búsqueda para los datos de objetivo de búsqueda al utilizar datos clave mediante un método mencionado posteriormente. La unidad 12 de almacenamiento es una unidad funcional configurada para almacenar datos de objetivo de búsqueda. La unidad 13 de entrada es una unidad funcional configurada para introducir datos clave. La unidad 14 de salida es una unidad funcional configurada para generar un resultado de búsqueda.

Por ejemplo, el aparato 10 de búsqueda es un ordenador de propósito general, y la unidad 11 de cálculo y la unidad 12 de almacenamiento forman una CPU. En este caso, la unidad 12 de almacenamiento corresponde a una caché en la CPU. También, una parte de la unidad 12 de almacenamiento puede ser una memoria fuera de la CPU. La CPU opera de acuerdo con un programa que tiene una lógica de procesamiento de acuerdo con la presente realización. El programa se almacena en la unidad 12 de almacenamiento. El programa se puede almacenar en una unidad de almacenamiento tal como una memoria distinta de la unidad 12 de almacenamiento.

El programa se puede almacenar en un medio de almacenamiento tal como una memoria portátil y similares, y cargarse en un ordenador de propósito general desde la memoria portátil, de modo que el ordenador se puede utilizar como aparato 10 de búsqueda.

También, la unidad 11 de cálculo y la unidad 12 de almacenamiento se pueden configurar como un aparato en el que una lógica de procesamiento de la presente realización está incorporada como un circuito de hardware.

La Fig. 3 muestra un aparato 20 de búsqueda que es otro ejemplo del aparato de búsqueda de la presente realización. El aparato 20 de búsqueda, por ejemplo, es un aparato que funciona como enrutador. Como se muestra en la Fig. 3, el aparato 20 de búsqueda incluye una unidad 21 de procesamiento de paquetes, una unidad 22 de determinación de destino y una pluralidad de interfaces (IF). La unidad 21 de procesamiento de paquetes recibe un paquete a través de una IF y emite el paquete desde una IF correspondiente a un destino (siguiente salto) determinado por la unidad 22 de determinación de destino. La Fig. 3 muestra un ejemplo en el que se recibe un paquete desde una IF-A y sale desde una IF-B.

La unidad 22 de determinación de destino incluye una unidad de almacenamiento para almacenar una tabla de enrutamiento (tabla de reenvío). La unidad 22 de determinación de destino recibe una dirección de destino de un paquete de la unidad 21 de procesamiento de paquetes como datos clave, y determina el siguiente salto del paquete al buscar en la tabla de reenvío en base a los datos clave para dar salida a la información del siguiente salto a la unidad 21 de procesamiento de paquetes. El aparato 10 de búsqueda mostrado en la Fig. 2, por ejemplo, se puede utilizar como unidad 22 de determinación de destino.

A continuación, se describen en detalle los procesos de búsqueda ejecutados por el aparato 10 de búsqueda. A continuación, se describe un esquema para realizar el procesamiento básico como un ejemplo 1, y se describen ejemplos en los que se agregan funciones que permiten la compresión de nodos para el ejemplo 1, como ejemplos 2-4.

(Ejemplo 1)

La Fig. 4 muestra un ejemplo de datos de objetivo de búsqueda almacenados en la unidad 12 de almacenamiento del aparato 10 de búsqueda. La Fig. 4 es común a los ejemplos 1-4. Como se describió anteriormente, en la presente realización, dado que se realiza el procesamiento de búsqueda en base al método de búsqueda raíz de múltiples vías, los datos de objetivo de búsqueda incluyen una matriz de nodos (matriz de nodos) que contiene datos de cada nodo interno del árbol, y una matriz de hojas (matriz de hojas) que son datos de cada nodo de hoja en el árbol. Se puede acceder a los datos de cada nodo almacenados como matriz al designar un índice de cada matriz.

Los datos de objetivo de búsqueda que incluyen la matriz de hojas y la matriz de nodos se pueden almacenar en un medio de almacenamiento tal como una memoria portátil y similares, por ejemplo, y se cargan en el aparato 10 de búsqueda desde la memoria portátil, de modo que el aparato 10 de búsqueda se puede utilizar como un aparato de búsqueda para los datos de objetivo de búsqueda. Los datos de objetivo de búsqueda también se pueden cargar en el aparato 10 de búsqueda a través de una red desde un ordenador.

Una estructura de datos de un nodo interno en el ejemplo 1 se describe con referencia a la Fig. 5. La Fig. 5 es un ejemplo del caso en el que la longitud de bit del fragmento es dos, es decir, cada nodo de las ramificaciones de los árboles en cuatro direcciones. Sin embargo, se aplica la misma estructura para cualquier longitud de bit del fragmento.

Como se muestra en la Fig. 5, el nodo interno incluye un vector, una base 0 y una base1. El vector es un vector de bits que incluye bits cuyo número es el número de ramificaciones del nodo interno. Cuando la parte de los datos clave es de dos bits, puede tomar cuatro tipos de valores de 00, 01, 10 y 11. Cada bit del vector corresponde a cada uno de los cuatro tipos de valores anteriores en un orden desde el extremo derecho. Tenga en cuenta que “desde el extremo derecho” es un ejemplo, y puede ser “desde el extremo izquierdo”. Por ejemplo, cuando se utiliza una CPU little endian, se cuenta desde el extremo derecho, y cuando se utiliza una CPU big endian, se cuenta desde el extremo izquierdo.

En el ejemplo de la Fig. 5, por ejemplo, el bit más a la derecha (0-ésimo) del vector corresponde al fragmento 00, el primer bit corresponde al fragmento 01, el segundo bit corresponde al fragmento 10, y el tercer bit corresponde al fragmento 11. Cada bit del vector indica si el destino de la transición (nodo hijo) del nodo interno es un nodo interno o un nodo de hoja. En la presente realización, 1 indica un nodo interno y 0 indica un nodo de hoja. Sin embargo, este es un ejemplo y el sistema se puede configurar de modo que 1 indique un nodo de hoja y 0 indique un nodo interno.

Por ejemplo, en el caso en el que el fragmento correspondiente al nodo interno mostrado en la FIG. 5 es 01 de 00, 01, 10 y 11, la unidad 11 de cálculo se refiere al primer bit (1) contado desde el bit 0-ésimo del vector, para determinar que el siguiente nodo es un nodo interno. También, por ejemplo, en el caso en el que el fragmento es 00 de 00, 01, 10 y 11, la unidad 11 de cálculo se refiere al bit 0-ésimo (0) del vector, para determinar que el siguiente nodo es un nodo de hoja.

Como se describió anteriormente, la unidad 11 de cálculo puede determinar si un nodo de destino de transición es un nodo interno o un nodo de hoja por el vector. Sin embargo, en este estado, para adquirir datos de un nodo interno/nodo de hoja, no se sabe a qué índice de elemento en la matriz de nodos/matriz de hojas se debe acceder. Por lo tanto, en la presente realización, el nodo interno contiene la base0 y la base1.

La base1 contiene un índice de inicio de almacenamiento de un nodo interno hijo que corresponde al bit 1 del vector en el nodo interno en la matriz de nodos. La base0 contiene un índice de inicio de almacenamiento de un nodo de hoja hijo correspondiente al bit 0 del vector en el nodo interno en la matriz de hojas.

En la presente realización, en la matriz de nodos, en cuanto a cada nodo interno, las piezas de datos de los nodos internos hijos del nodo interno se almacenan en un orden de índice. Por ejemplo, en el caso en el que hay tres nodos internos hijos para un nodo interno, las tres piezas de datos de los nodos internos hijos se almacenan en la matriz de nodos como tres piezas de datos cuyos índices son consecutivos. El índice de datos cuyo índice es el superior (el más pequeño) entre los tres datos es la base1.

También, en la matriz de hojas, en cuanto a cada nodo interno, las piezas de datos de los nodos de hojas hijos del nodo interno se almacenan en un orden de índice. Por ejemplo, en el caso en el que hay tres nodos de hoja hijo para un nodo interno, los tres datos de los nodos de hoja hijo se almacenan en la matriz de hojas como tres datos cuyos índices son consecutivos. El índice de datos cuyo índice es el superior (el más pequeño) entre las tres piezas de datos es la base0. Nótese que el índice utilizado en la presente realización es un indicador que indica una ubicación de almacenamiento, y esto puede reemplazarse con “dirección”.

Dado que los datos se almacenan en la matriz de nodos/matriz de hojas de la manera mencionada anteriormente, la unidad 11 de cálculo accede a los datos de un nodo interno/nodo de hoja hijo al utilizar base0/base1 como se describe a continuación.

En cuanto a un acceso a un nodo interno hijo de una posición de bit (se supone que es la posición v-ésima contada desde la posición 0-ésima) en el vector, la unidad 11 de cálculo calcula (cuenta) el número de 1s en posiciones de bit desde la posición 0-ésima hasta la posición v-ésima del vector. Es decir, la unidad 11 de cálculo calcula el número de 1s en (v + 1) bits desde el extremo derecho del vector. Si este número se representa como bc (recuento de bits), la

unidad 11 de cálculo accede a un índice de un valor (be + base1 - 1) obtenido al restar 1 desde un valor obtenido al agregar base1 a bc en la matriz de nodos para ser capaz de obtener datos del nodo interno.

En cuanto a un acceso a un nodo de hoja hijo de una posición de bit (se supone que es la posición v-ésima contada desde la posición 0-ésima) en el vector, la unidad 11 de cálculo calcula (cuenta) el número de 0s en posiciones de bit desde la posición 0-ésima hasta la posición v-ésima del vector. Es decir, la unidad 11 de cálculo calcula el número de 0s en bits (v+1) desde el extremo derecho del vector. Si este número se representa como bc (recuento de bits), la unidad 11 de cálculo accede a un índice de un valor (be + base0 - 1) obtenido al restar 1 desde un valor obtenido al agregar base0 a bc en la matriz de hojas para ser capaz de obtener datos del nodo de hoja.

La Fig. 5 muestra el acceso a un nodo interno hijo (índice: 2498) y a nodos de hoja hijos (índice: 3127~3129) de la manera mencionada anteriormente.

En general, la CPU tiene una función llamada popcnt que calcula el número de bits a alta velocidad. En la presente realización, esta función se puede utilizar de forma eficaz de modo que se pueda realizar una búsqueda de alta velocidad. Tenga en cuenta que el uso de popcnt es un ejemplo, y es posible que no se utilice popcnt.

La Fig. 5 muestra un ejemplo en el que la longitud del fragmento es de 2 bits, es decir, el vector es de 4 bits. Sin embargo, este es un ejemplo, y la longitud del fragmento/vector puede tener otras longitudes. La Fig. 6 muestra un ejemplo en el que la longitud del fragmento es de 6 bits, es decir, el vector es de 64 (2^6) bits. La Fig. 6 muestra, como ya se ha descrito, que el nodo interno incluye el vector, base0/base1, y que se puede acceder a un nodo interno/nodo de hoja hijo mediante recuentos de bits y base0/base1.

En la presente realización, el nodo interno solo necesita tener un vector de bits y dos bases. En comparación con el esquema que tiene indicadores para cada ramificación, la cantidad de datos de cada nodo se puede reducir en gran medida y, como resultado, se puede reducir la cantidad de datos de los datos de objetivo de búsqueda.

Con referencia a la FIG. 7, se describe un procedimiento de proceso de procesamiento de búsqueda ejecutado por la unidad 11 de cálculo. Como premisa de este procesamiento, se supone que los datos clave se han introducido en la unidad 11 de cálculo y los datos de objetivo de búsqueda (matriz de nodos/matriz de hojas) que tienen la estructura descrita anteriormente se han almacenado en la unidad 12 de almacenamiento. Fig. 7 muestra un ejemplo en el que el proceso de búsqueda finaliza al llegar a un nodo de hoja.

La unidad 11 de cálculo obtiene un vector de un primer nodo interno en la matriz de nodos (etapa 101) y obtiene un primer fragmento de los datos clave (etapa 102).

La unidad 11 de cálculo lee un bit en una posición del vector correspondiente al fragmento para determinar si el bit es 1 (etapa 103). Cuando el bit es 1, como se describió anteriormente, la unidad 11 de cálculo calcula el número de bits be y accede a un nodo interno almacenado en un índice de (bc+base1-1) para obtener un vector del nodo interno (etapa 104).

La unidad 11 de cálculo obtiene un siguiente fragmento de los datos clave (etapa 105) y ejecuta la determinación de la etapa 103 de nuevo.

Como resultado de la determinación de la etapa 103, cuando un bit de la posición del vector correspondiente al fragmento es 0 (No en la etapa 103), el proceso pasa a la etapa 106. En la etapa 106, como se describió anteriormente, la unidad 11 de cálculo calcula el recuento de bits be y accede a un nodo de hoja almacenado en un índice de (bc+base0-1) para obtener un valor del nodo de hoja.

Tenga en cuenta que, por ejemplo, en una tabla de reenvío, dado que la longitud de los prefijos que son hojas tiende a concentrarse en un rango específico (ejemplo:/11~/24), es posible alcanzar una entrada de destino directamente al omitir la búsqueda de nodos iniciales. A esto se le llama señalamiento directo. Se describe un ejemplo con referencia a las Figs. 8A y 8B

La FIG. 8A muestra un ejemplo en el que no se realiza el señalamiento directo. En este ejemplo, la búsqueda se realiza utilizando fragmentos que tienen cada uno 6 bits cada uno. La Fig. 8B muestra un ejemplo de señalamiento directo en el que la búsqueda se realiza utilizando primero un fragmento de 12 bits. También en el caso de la Fig. 8B, cuando el fragmento no puede alcanzar un nodo de hoja, la búsqueda se realiza utilizando fragmentos que tienen cada uno 6 bits después de eso, por ejemplo, de la misma manera que se describió anteriormente. El señalamiento directo también se puede aplicar a otros ejemplos.

(Ejemplo 2)

A continuación, como ejemplo 2, se describe un esquema en el que se pueden comprimir nodos de hoja para el esquema descrito en el ejemplo 1. Por ejemplo, al aplicar el esquema del ejemplo 1 para buscar una tabla de reenvío, se puede considerar que se producen muchos nodos de hoja que tienen valores superpuestos (siguientes saltos). En el ejemplo 2, en base al esquema del ejemplo 1, los nodos de hoja se pueden mantener al comprimirlos. A

continuación, se describen principalmente partes diferentes del ejemplo 1.

La Fig. 9 es un diagrama que muestra un nodo interno en el ejemplo 2. Como se muestra en la Fig. 9, en el ejemplo 2, se agrega un leafvec además del vector, la base0 y la base1 descrita en el ejemplo. 1. La longitud de bits de leafvec es la misma que la del vector.

En cuanto a los nodos de hoja que se convierten cada uno en un hijo de cada nodo interno (es decir, nodos de hoja de cada etapa) en la matriz de hojas, solo se mantiene un primer nodo de hoja en el que comienza la consecución para los nodos de hoja consecutivos que tienen el mismo valor. En el ejemplo de la Fig. 9, en cuanto a los nodos de hoja de índices de 3127, 3128 y 3129, todos los valores son iguales y es 7. En este caso, sólo se mantiene el nodo de hoja del índice 3127. Como resultado de dicha compresión, incluso cuando hay una pluralidad de nodos de hoja, no se incluyen una pluralidad de nodos de hoja que tienen el mismo valor, y cada nodo de hoja tiene un valor diferente.

El elemento de leafvec es un bit de 0 o 1, y 1 se asigna en un bit correspondiente a una posición en la que comienza la consecución de los nodos de hoja antes de la compresión, desde el extremo derecho. Por ejemplo, en el ejemplo de la Fig. 9, dado que la consecución comienza desde el primer nodo de hoja, se establece 1 en el primer bit (0-ésimo) correspondiente al primer nodo de hoja. También, en el caso en el que la consecución termina de modo que comienza un nodo de hoja de otro valor (cuando cambia un nodo de hoja), se establece 1 en la posición. El caso en el que un nodo de hoja cambia incluye el primer nodo de hoja. La “consecución” aquí incluye el caso de un nodo de hoja. Es decir, cuando cada pieza de datos de los nodos de hoja es diferente, se establece 1 en cada posición de bit del leafvec correspondiente a los nodos de hoja. El uso de leafvec es el siguiente.

Cuando la unidad 11 de cálculo detecta que un bit (se supone que es un bit v-ésimo contado desde el bit 0-ésimo) del vector correspondiente a un fragmento es 0, la unidad 11 de cálculo determina que el hijo es un nodo de hoja. La unidad 11 de cálculo calcula el número de bits de 1 en los bits (bits v+1) contados desde el bit 0-ésimo del extremo derecho hasta el bit v-ésimo. Suponiendo que el número sea, como en el caso del vector, la unidad 11 de cálculo accede a un nodo de hoja de un índice de (bc+base0-1).

La Fig. 10 muestra un ejemplo de datos de un nodo interno y un nodo de hoja en el ejemplo 2. En el ejemplo de la Fig. 10, se muestra que la unidad 11 de cálculo detecta que el primer bit contado desde el bit 0-ésimo del vector en el nodo interno mostrado como (a) es 1 basado en el fragmento, y accede a un nodo interno de (c) correspondiente a este. También, por ejemplo, en el nodo interno de (a), cuando el fragmento corresponde al segundo bit (0) contado desde el bit 0-ésimo, la unidad 11 de cálculo calcula el número de 1s en 3 bits hasta el segundo bit en leafvec, y accede a un nodo de hoja (L(0)) correspondiente al número que utiliza la base0.

La compresión de los nodos de las hojas se puede realizar mediante un método diferente al método que utiliza leafvec como se describió anteriormente. A continuación, se describe otro método de compresión de nodos de hojas como un ejemplo 3. Nótese que el método del ejemplo 3 es sustancialmente el mismo que el método que utiliza leafvec.

(Ejemplo 3)

La Fig. 11 es un diagrama que muestra un nodo interno en el ejemplo 3. Como se muestra en la Fig. 11, en el ejemplo 3, se agrega una máscara además del vector, la base0 y la base1 descritos en el ejemplo 1. La longitud de bits de la máscara es la misma que la del vector.

En cuanto a los nodos de hoja que se convierten cada uno en un hijo de cada nodo interno (es decir, nodos de hoja de cada etapa) en la matriz de hojas, solo se mantiene un primer nodo de hoja en el que comienza la consecución para los nodos de hoja consecutivos que tienen el mismo valor. En el ejemplo de la Fig. 11, en cuanto a los nodos de hoja de índices de 3127, 3128 y 3129, todos los valores son iguales y éste es 7. En este caso, sólo se mantiene el nodo de hoja del índice 3127. Como resultado de dicha compresión, incluso cuando hay una pluralidad de nodos de hojas, no se incluye una pluralidad de nodos de hojas que tienen el mismo valor.

El elemento de la máscara es un bit de 0 o 1, y 0 se asigna a un bit correspondiente a una posición en la que inicia la consecución de los nodos de hoja antes de la compresión, desde el extremo derecho, y 1 (máscara) se asigna en posiciones de nodos conductores consecutivos que tienen el mismo valor desde la posición inicial. También, en el caso en el que la consecución termina de modo que comienza un nodo de hoja de otro valor (cuando cambia un nodo de hoja), se establece 0 en la posición. El caso en el que un nodo de hoja cambia incluye el primer nodo de hoja.

Tenga en cuenta que, aunque se puede establecer 1 o 0 en una posición correspondiente a un nodo interno, en este ejemplo se utiliza 0. En el ejemplo de la Fig. 11, dado que tres nodos de hoja son consecutivos, 0 se establece en una posición de bit correspondiente al primer nodo de hoja, y 1, que es una máscara, se establece en posiciones de bit correspondientes a los nodos de hoja posteriormente. El uso de la máscara es el siguiente.

Cuando la unidad 11 de cálculo detecta que un bit (se supone que es un bit v-ésimo contado desde el bit 0-ésimo) del vector correspondiente a un fragmento es 0, la unidad 11 de cálculo determina que el hijo es un nodo de hoja. En el ejemplo 3, la unidad 11 de cálculo realiza un cálculo OR sobre el vector y la máscara, y la unidad 11 de cálculo calcula el número de bits de 0 en los bits (bits v+1) contados desde el bit 0-ésimo del extremo derecho hasta el bit v-ésimo

del vector en el que se ha realizado el cálculo OR. Suponiendo que el número sea, la unidad 11 de cálculo accede a un nodo de hoja de un índice de (bc+base0-1).

La Fig. 12 muestra un ejemplo de datos de un nodo interno y un nodo de hoja en el ejemplo 3. En el ejemplo de la Fig. 12, se muestra que la unidad 11 de cálculo detecta que el primer bit contado desde el bit 0-ésimo del vector en el nodo interno mostrado como (a) es 1 basado en el fragmento, y accede a un nodo interno de (c) correspondiente a éste. También, por ejemplo, en el nodo interno de (a), cuando el fragmento corresponde al segundo bit (0) contado desde el bit 0-ésimo, la unidad 11 de cálculo calcula el número de 0s en 3 bits hasta el segundo bit en el vector después de la operación de máscara, y accede a un nodo de hoja (L(0)) correspondiente al número que utiliza la base0.

La máscara también se puede aplicar a la compresión de nodos internos. Como ejemplo en el que la máscara se aplica a la compresión de nodos internos se describe con referencia a la Fig. 13. Como en la Fig. 6, la Fig. 13 muestra un ejemplo en el que la longitud del fragmento es de 6 bits, es decir, el vector tiene 64 (2^6) bits. También en este ejemplo, la máscara se agrega además del vector, el base0 y el base1 descritos en el ejemplo 1. La longitud de bits de la máscara es la misma que la del vector.

También, en cuanto a los nodos internos de cada etapa, solo el primer nodo interno en el que comienza la consecución se mantiene para los nodos internos consecutivos que tienen el mismo valor. En el ejemplo de la Fig. 13, como se indica en (a), hay 3 nodos internos que tienen el mismo subárbol. En este caso, como se indica en (b), sólo se mantiene el primer nodo interno entre los tres. Como resultado de dicha compresión, incluso cuando hay una pluralidad de nodos internos, no se incluyen una pluralidad de nodos internos que tengan el mismo valor.

El elemento de la máscara es un bit de 0 o 1, y se asigna 1 en un bit correspondiente a una posición en la que inicia la consecución de los nodos internos antes de la compresión, desde el extremo derecho, y 0 (máscara) se asigna en posiciones de nodos internos consecutivos que tienen el mismo valor desde la posición inicial. También, en el caso de que la consecución finalice de modo que comience un nodo interno de otro valor (cuando cambie un nodo interno), se establece 1 en la posición.

En el ejemplo de la Fig. 13, dado que tres nodos internos son consecutivos, se establece 1 en una posición de bit correspondiente al primer nodo interno, y 0, que es una máscara, se establece en posiciones de bit correspondientes a nodos internos posteriormente. Es decir, como se muestra en la Fig. 13 (b), el bit de la máscara correspondiente al primer 1 del vector es 1, y los bits de la máscara correspondientes al siguiente 1 y más al siguiente 1 son 0. El uso de la máscara es el siguiente.

Cuando la unidad 11 de cálculo detecta que un bit (se supone que es un bit v-ésimo contado desde el bit 0-ésimo) del vector correspondiente a un fragmento es 1, la unidad 11 de cálculo determina que el hijo es un nodo interno. La unidad 11 de cálculo realiza el cálculo AND sobre el vector y la máscara, y la unidad 11 de cálculo calcula el número de bits de 1 en los bits (v + 1 bits) contados desde el bit 0-ésimo del extremo derecho hasta el bit v -ésimo del vector en el que se ha realizado el cálculo AND. Suponiendo que el número sea, la unidad 11 de cálculo accede a un nodo interno de un índice de (bc+base1-1).

(Ejemplo 4)

A continuación, se describe un ejemplo 4. El ejemplo 4 es un esquema mediante el cual los nodos de hoja se pueden comprimir adicionalmente que en las realizaciones 2 y 3. La estructura de los datos internos en el ejemplo 4 se muestra en la Fig. 14. Como se muestra en la Fig. 14, en los datos internos del ejemplo 4, se agregan una máscara de hoja y una hoja enmascarada como se muestra por “A” además del vector, el leafvec, el base0 y el base1. Una matriz de nodos y una matriz de hojas se almacenan en la unidad 12 de almacenamiento.

La máscara de hoja son datos, que constan de 0/1 bits, que tienen una longitud de bits igual a la del vector. La hoja enmascarada son datos de un nodo de hoja. A continuación, se describe la operación de la unidad 11 de cálculo cuando se utiliza la máscara de hoja y la hoja enmascarada.

Un ejemplo de operación de la unidad 11 de cálculo de los aparatos 10 de búsqueda en el ejemplo 4 se describe con referencia al diagrama de flujo de la Fig. 15. La Fig. 15 es especialmente para describir partes del procesamiento que son diferentes a aquellos de los ejemplos 1 y 2.

En la etapa 201, la unidad 11 de cálculo detecta que transita a un nodo de hoja al detectar que el bit correspondiente del vector del fragmento actual (el bit v-ésimo contado desde el bit 0-ésimo) es 0.

En la etapa 202, la unidad 11 de cálculo determina si el bit v-ésimo contado desde el bit 0-ésimo en la máscara de hoja es 1. Cuando es 1 (Sí en la etapa 202), la unidad 11 de cálculo obtiene el valor de la hoja enmascarada como un valor del nodo de hoja (etapa 203).

En la etapa 202, cuando el bit v-ésimo no es 1 (No en la etapa 202), de la misma forma que en el ejemplo 2, la unidad 11 de cálculo calcula el número (be) de 1s desde el bit 0-ésimo hasta el bit v-ésimo del leafvec, y obtiene un valor accediendo a un nodo de hoja de un índice de (bc+base0-1).

A continuación, se describe un método para generar datos relacionados con la máscara de hoja en el ejemplo 4 con referencia a la Fig. 16. La generación de datos descritos a continuación puede ser realizada por el aparato 10 de búsqueda o puede ser realizada por otro aparato (ordenador). A continuación, un aparato que genera datos se denomina aparato de generación. El aparato de generación es el aparato 10 de búsqueda u otro aparato. Cuando el aparato de generación es otro aparato, después de que se generan los datos, los datos generados se almacenan en la unidad 12 de almacenamiento del aparato 10 de búsqueda.

En primer lugar, el aparato de generación calcula una matriz de hojas sin compresión. De acuerdo con lo anterior, por ejemplo, en lo que respecta a un árbol cuádruple, como se muestra en L de la Fig. 5, por ejemplo, los datos de los nodos de hoja cuyos índices son consecutivos se generan para cada nodo interno del padre.

También, como un árbol de 64 ramificaciones, el número de elementos de la matriz de hojas se convierte en 64 como máximo para cada nodo interno del padre. Además, por ejemplo, en un ejemplo de un árbol de 16 ramificaciones, cuando hay tres tipos de información de hojas, que son A, B y C, la información de hojas se dispone en la matriz de hojas como ABAA BBBA BCBB CCCC, por ejemplo, como se muestra en la Fig. 16(a).

A continuación, el aparato de generación selecciona la información de la hoja que se va a enmascarar. En este ejemplo, se omite B al enmascararlo. En general, resulta eficaz enmascarar información para la que aparecen fragmentos consecutivos con mayor frecuencia. Por tanto, el aparato de generación determina enmascarar B para qué fragmentos consecutivos aparecen con mayor frecuencia. Tenga en cuenta que “fragmentos consecutivos” incluye el caso de uno como B en ABA. La información B de la hoja enmascarada se almacena en la hoja enmascarada.

A continuación, el aparato de generación almacena una ranura donde aparece información de hoja enmascarada en la máscara de hoja. La “ranura donde aparece la información de la hoja enmascarada” corresponde a una posición de bit correspondiente a la hoja en el vector. Por ejemplo, en el caso en el que el vector es 0010, cuando una hoja correspondiente a un bit 0 de la segunda ranura contada desde el extremo izquierdo como primer bit está enmascarada, 0100 se almacena en la máscara de hoja.

También, el aparato de generación hace que la ranura de la información de hoja enmascarada sea la misma que el valor anterior en la matriz de hojas. De acuerdo con lo anterior, a partir de la información de la hoja mostrada en la Fig. 16(a), se obtiene la “máscara de hoja: 0100 1110 1011 0000” y la “matriz de hojas: AAAA AAAA ACCC CCCC” como se muestra en la Fig. 16(b). Tenga en cuenta que, en este ejemplo, dado que se utiliza big endian, el recuento se realiza desde el extremo izquierdo. En la Fig. 16(b), las partes subrayadas son partes enmascaradas, que son valores iguales a los valores anteriores (valores de la izquierda) en la dirección de conteo.

A continuación, de la misma forma que en el caso sin máscara de hoja, se comprimen partes consecutivas. De acuerdo con lo anterior, como se muestra en la Fig. 16(c), se obtiene “leafvec: 1000 0000 0100 0000” y se obtiene “matriz de hojas: AC”.

Como resultado del procesamiento mencionado anteriormente, como se muestra en la Fig. 16(d), se obtienen “máscara de hoja: 0100 1110 1011 0000”, “hoja enmascarada: B”, “vector de hoja: 1000 0000 0100 0000”, y la “matriz de hojas: ACJ”.

Como referencia, la matriz de hojas cuando se comprime sin la máscara de hojas es “ABABABCBC”, lo que indica que se puede obtener un efecto de alta compresión mediante el ejemplo 4.

En el ejemplo 4, aunque se agregan una máscara (ejemplo: 64 bits) y una hoja, se pueden omitir algunas hojas no consecutivas, de modo que se pueda realizar una compresión adicional de la matriz de hojas. Esto se vuelve especialmente efectivo cuando la matriz de hojas se divide en muchas partes por los valores del siguiente salto (como rayas), o cuando el tamaño de una hoja (tamaño de una entrada de la matriz de hojas) es grande, tal como 16 bytes.

Obsérvese que los ejemplos 2, 3 y 4 muestran ejemplos para comprimir nodos de hoja, sin embargo, los nodos internos que tienen el mismo valor también se pueden comprimir como el caso de los nodos de hoja. También, se pueden realizar tanto la compresión de los nodos de hoja como la compresión de los nodos internos.

(Efectos de las realizaciones)

Como se describió anteriormente, en la presente realización, la cantidad de datos del árbol se puede reducir en gran medida. Por lo tanto, por ejemplo, el procesamiento de búsqueda se puede realizar al almacenar los datos de objetivo de búsqueda en una caché (ejemplo: caché L1, L2, L3) de una CPU de propósito general, de modo que se pueda realizar un procesamiento de búsqueda de alta velocidad. Especialmente, como ejemplo, cuando los datos de objetivo de búsqueda son una tabla de enrutamiento, se resuelve el problema de que el rendimiento se deteriora cuando el número de rutas en la tabla de enrutamiento se hace grande. También, dado que la velocidad de procesamiento aumenta, también es posible resolver el problema de que el rendimiento se deteriora cuando la longitud de la dirección se vuelve larga.

También, en cada nivel del árbol, dado que la presencia o ausencia de un árbol parcial se expresa bit a bit, la eficiencia de la memoria es buena. Especialmente, como ejemplo, cuando se utiliza un árbol de 64 ramificaciones, dado que la

presencia o ausencia (disposición hija) de un árbol parcial está representada por 64 bits a la vez, existe la característica de que es buena la eficiencia de procesamiento de la CPU de 64 bits.

También, en el vector y similares, se cuenta el número de bits de 1, de modo que se puede acceder al hijo correspondiente en la matriz en una etapa. Por lo tanto, se puede realizar un procesamiento de alta velocidad y la eficiencia de la memoria es buena. También, dado que se cuenta el número de bits de 1, se pueden realizar la instrucción de CPU popcnt y el procesamiento de alta velocidad. También, dado que la presente técnica se basa en un árbol de múltiples vías de propósito general (trie de múltiples vías), la presente técnica tiene una alta extensibilidad y flexibilidad, y es aplicable no solo a la búsqueda de tablas de rutas sino también a varias búsquedas.

Adicionalmente, al realizar la compresión de la información de hoja descrita en los ejemplos 2~4, se puede reducir la cantidad de datos de objetivo de búsqueda y se puede realizar una aceleración adicional.

Como ejemplo, al utilizar la técnica de la presente realización, con una CPU de propósito general que tiene la instrucción de CPU popcnt, se pueden realizar aproximadamente 240 Mlps para un solo núcleo y aproximadamente 910 Mlps para 4 núcleos para una tabla de enrutamiento que contiene 500 mil rutas de ruta completa IPv4. Sin utilizar TCAM, la CPU de propósito general puede lograr un rendimiento igual que TCAM o varias veces mejor que TCAM.

(Resumen de la realización)

Como se describió anteriormente, de acuerdo con la presente realización, se proporciona un aparato de búsqueda que incluye:

medios de almacenamiento configurados para almacenar datos de objetivo de búsqueda; y

medios de cálculo configurados para realizar procesamiento de búsqueda para los datos de objetivo de búsqueda en base a datos clave, en los que

los datos de objetivo de búsqueda almacenados en los medios de almacenamiento son datos de una estructura de árbol de múltiples vías que incluye una matriz de nodos internos y una matriz de nodos de hoja,

cada nodo interno en los datos de objetivo de búsqueda incluye un vector de bit que representa si un destino de transición es un nodo interno o un nodo de hoja por un bit, y en el que

los medios de cálculo se configuran para ejecutar repetidamente, hasta que un nodo de transición se convierte en un nodo de hoja, el procesamiento de

obtener un fragmento de una longitud de bits predeterminada a partir de los datos clave, determinar si un destino de transición desde el nodo interno es un nodo interno o un nodo de hoja en base a un bit, en el vector de bit del nodo interno de acceso, que corresponde a un valor del fragmento, y acceder a un nodo del destino de transición.

Cada nodo interno en los datos de objetivo de búsqueda incluye primera información base que indica una posición de almacenamiento de un nodo interno de un destino de transición, y segunda información base que indica una posición de almacenamiento de un nodo de hoja de un destino de transición, y

se pueden configurar los medios de cálculo, cuando el destino de transición determinado en base al valor del bit del vector de bit es un nodo interno, para acceder al nodo interno del destino de transición utilizando la primera información base, y cuando el destino de transición es un nodo de hoja, para acceder al nodo de hoja del destino de transición utilizando la segunda información base.

Para cada nodo interno en los datos de objetivo de búsqueda, los nodos internos que se convierten en destinos de transición se almacenan en la matriz de nodos en la que son consecutivas las posiciones de almacenamiento, y nodos de hoja que se convierten en destinos de transición se almacenan en la matriz de nodos de hoja en la que son consecutivas las posiciones de almacenamiento, y

se pueden configurar los medios de cálculo, cuando el destino de transición determinado en base al valor del bit del vector de bit es un nodo interno, para acceder al nodo interno del destino de transición utilizando la primera información base y el número de bits que indica un nodo interno en el vector de bit, y

cuando el destino de transición es un nodo de hoja, para acceder al nodo de hoja del destino de transición utilizando la segunda información base y el número de bits que indica un nodo de hoja del vector de bit.

En cuanto a cada nodo interno en los datos de objetivo de búsqueda, los nodos de hoja que se convierten en destinos de transición se almacenan en la matriz de nodos de hoja en la que son consecutivas las posiciones de almacenamiento, se comprimen nodos de hoja que tienen el mismo valor, y una pluralidad de nodos de hoja no incluye nodos de hoja que tienen el mismo valor,

cada nodo interno en los datos de objetivo de búsqueda incluye un vector de hoja que tiene un bit que indica una posición de almacenamiento en la que se cambia un valor de un nodo de hoja antes de compresión, y

se pueden configurar los medios de cálculo, cuando el destino de transición determinado en base al valor del bit del vector de bit es un nodo de hoja, para acceder al nodo de hoja del destino de transición utilizando la segunda información base y el número de bits que indica la posición de almacenamiento en el vector de hoja.

Se pueden configurar los medios de cálculo para verificar que el vector de bit primero está entre el vector de bit y el vector de hoja, y para utilizar el vector de hoja en base al valor del bit del vector de bit.

En cuanto a cada nodo interno en los datos de objetivo de búsqueda, los nodos de hoja que se convierten en destinos de transición se almacenan en la matriz de nodos de hoja en la que son consecutivas las posiciones de almacenamiento, se comprimen nodos de hoja que tienen el mismo valor, y una pluralidad de nodos de hoja no incluye nodos de hoja que tienen el mismo valor,

cada nodo interno en los datos de objetivo de búsqueda incluye un vector de máscara que tiene un bit que indica una posición de almacenamiento en la que se cambia un valor de un nodo de hoja antes de compresión, y

se pueden configurar los medios de cálculo, cuando el destino de transición determinado en base al valor del bit del vector de bit es un nodo de hoja, para acceder al nodo de hoja del destino de transición utilizando la segunda información base y el número de bits que indica los nodos de hoja en el vector de bit enmascarado por el vector de máscara.

En cuanto a cada nodo interno en los datos de objetivo de búsqueda, los nodos internos que se convierten en destinos de transición se almacenan en la matriz de nodos en la que son consecutivas las posiciones de almacenamiento, se comprimen los nodos internos que tienen el mismo valor, y una pluralidad de nodos internos no incluye nodos internos que tienen el mismo valor,

cada nodo interno en los datos de objetivo de búsqueda incluye un vector de máscara que tiene un bit que indica una posición de almacenamiento en la que cambia un valor de un nodo interno antes de compresión, y

se pueden configurar los medios de cálculo, cuando el destino de transición determinado en base al valor del bit del vector de bit es un nodo interno, para acceder al nodo interno del destino de transición utilizando la primera información base y el número de bits que indica los nodos internos en el vector de bit enmascarado por el vector de máscara.

En cuanto a cada nodo interno en los datos de objetivo de búsqueda, se comprimen nodos de hoja que tienen el mismo valor después de que se enmascara un valor predeterminado en los nodos de hoja que se convierten en destinos de transición y el valor enmascarado se cambia a un valor diferente, de tal manera que una pluralidad de nodos de hoja no incluye nodos de hoja que tienen el mismo valor, y se almacenan en la matriz de nodos de hoja en la que son consecutivas las posiciones de almacenamiento,

cada nodo interno en los datos de objetivo de búsqueda incluye el valor predeterminado enmascarado, una máscara de hoja que tiene un bit que indica una posición del vector de hoja que tiene el valor predeterminado antes de compresión, y un vector de hoja que tiene un bit que indica una posición de almacenamiento en la que se cambia un valor de un nodo de hoja antes de compresión, y

se pueden configurar los medios de cálculo, cuando el destino de transición determinado en base al valor del bit del vector de bit es un nodo de hoja,

para determinar si un bit se establece en la máscara de hoja en una posición igual a la del bit en el vector de bit, obtener el valor predeterminado como un valor del nodo de hoja del destino de transición cuando se establece el bit, y acceder al nodo de hoja del destino de transición utilizando la segunda información base y el número de bits que indica la posición de almacenamiento en el vector de hoja cuando no se establece el bit.

Se pueden configurar los medios de cálculo para calcular el número de bits utilizando un comando popcnt de una CPU configurada por los medios de cálculo.

Los medios de cálculo y los medios de almacenamiento se pueden configurar sobre una CPU de 64 bit, y el fragmento puede tener una longitud de 6 bit, y el vector de bit puede tener una longitud de 64 bit.

También, los medios de cálculo y los medios de almacenamiento se configuran sobre una CPU de 64 bit, el fragmento tiene una longitud de 6 bit, y el vector de bit tiene una longitud de 64 bit, y se pueden configurar los medios de cálculo para calcular el número de los bits utilizando un comando popcnt de la CPU de 64 bit, y para acceder un nodo del destino de transición utilizando una compensación, a partir de información base, en base al número de los bits.

Se pueden configurar los medios de cálculo para obtener un fragmento de una longitud de bit más larga que la longitud de bits predeterminada a partir de los datos clave, y alcanzar directamente un nodo de hoja al realizar la búsqueda utilizando el fragmento.

También, de acuerdo con la presente realización, se puede proporcionar un programa que hace que un ordenador funcione como cada medio en el aparato de búsqueda. También, de acuerdo con la presente realización, se puede proporcionar un medio de registro legible por ordenador que almacene los datos de objetivo de búsqueda.

Nótese que los “medios de almacenamiento” descritos anteriormente se pueden reemplazar por cualquiera de una unidad de almacenamiento, un circuito de almacenamiento y un dispositivo de almacenamiento. También, los “medios de cálculo” descritos anteriormente se pueden reemplazar por cualquiera de una unidad de cálculo, un circuito de cálculo y un dispositivo de cálculo.

También, se puede configurar un método de búsqueda de la presente realización como un método de búsqueda para realizar el procesamiento de búsqueda sobre datos de objetivo de búsqueda en base a datos clave, en el que los datos de objetivo de búsqueda son datos de una estructura de árbol de múltiples vías que incluyen una matriz de nodos internos y una matriz de nodos de hoja,

cada nodo interno en los datos de objetivo de búsqueda incluye un vector de bit que representa si un destino de transición es un nodo interno o un nodo de hoja por un bit,

el método de búsqueda incluye: ejecutar repetidamente, hasta que un nodo de transición se convierte en un nodo de hoja, el procesamiento incluye obtener un fragmento de una longitud de bits predeterminada a partir de los datos clave, determinar si un destino de transición desde el nodo interno es un nodo interno o un nodo de hoja en base a un bit, en el vector de bit del nodo interno de acceso, que corresponde a un valor del fragmento, y acceder a un nodo del destino de transición.

La presente invención no se limita a las realizaciones descritas específicamente, y se pueden realizar variaciones y modificaciones sin apartarse del alcance de las reivindicaciones.

Descripción de las señales de referencia

10, 20 aparato de búsqueda

11 unidad de cálculo

12 unidad de almacenamiento

13 unidad de entrada

14 unidad de salida

21 unidad de procesamiento de paquetes

22 unidad de determinación de destino

REIVINDICACIONES

1. Un aparato (10, 20) de búsqueda que comprende:

medios (12) de almacenamiento configurados para almacenar datos de objetivo de búsqueda; y

medios (11) de cálculo configurados para realizar procesamiento de búsqueda para los datos de objetivo de búsqueda en base a datos clave, en el que

los datos de objetivo de búsqueda almacenados en los medios (12) de almacenamiento son datos de una estructura de árbol de múltiples vías que incluye una matriz de nodos internos y una matriz de nodos de hoja,

caracterizado porque:

cada nodo interno en los datos de objetivo de búsqueda incluye un vector de bit que representa si un destino de transición es un nodo interno o un nodo de hoja por un bit, primera información base que indica una posición de almacenamiento de un nodo interno de un destino de transición, y segunda información base que indica una posición de almacenamiento de un nodo de hoja de un destino de transición, y en el que

se configuran los medios (11) de cálculo para ejecutarse repetidamente, hasta que un nodo de transición se convierte en un nodo de hoja, el procesamiento incluye

obtener un fragmento de una longitud de bits predeterminada a partir de los datos clave,

determinar si un destino de transición del nodo interno es un nodo interno o un nodo de hoja en base a un bit, en el vector de bit del nodo interno de acceso, que corresponde a un valor del fragmento, y

cuando el destino de transición determinado en base al valor del bit del vector de bit es un nodo interno, acceder al nodo interno del destino de transición utilizando la primera información base, y cuando el destino de transición es un nodo de hoja, acceder al nodo de hoja del destino de transición utilizando la segunda información base.

2. El aparato (10, 20) de búsqueda como se reivindica en la reivindicación 1, en el que, para cada nodo interno en los datos de objetivo de búsqueda, los nodos internos que se vuelven destinos de transición se almacenan en la matriz de nodos internos en la que son consecutivas las posiciones de almacenamiento, y los nodos de hoja que se vuelven destinos de transición se almacenan en la matriz de nodos de hoja en la que son consecutivas las posiciones de almacenamiento, y en el que

se configuran los medios (11) de cálculo, cuando el destino de transición determinado en base al valor del bit del vector de bit es un nodo interno, para acceder al nodo interno del destino de transición utilizando la primera información base y el número de bits que indica un nodo interno en el vector de bit, y

cuando el destino de transición es un nodo de hoja, para acceder al nodo de hoja del destino de transición utilizando la segunda información base y el número de bits que indica un nodo de hoja del vector de bit.

3. El aparato (10, 20) de búsqueda como se reivindica en la reivindicación 1, en el que, en cuanto a cada nodo interno en los datos de objetivo de búsqueda, los nodos de hoja que se convierten en destinos de transición se almacenan en la matriz de nodos de hoja en la que son consecutivas las posiciones de almacenamiento, se comprimen los nodos de hoja que tienen el mismo valor, y una pluralidad de nodos de hoja no incluye nodos de hoja que tienen el mismo valor,

cada nodo interno en los datos de objetivo de búsqueda incluye un vector de hoja que tiene un bit que indica una posición de almacenamiento en la que cambia un valor de un nodo de hoja antes de compresión, y en el que

se configuran los medios (11) de cálculo, cuando el destino de transición determinado en base al valor del bit del vector de bit es un nodo de hoja, para acceder al nodo de hoja del destino de transición que utiliza la segunda información base y el número de bits que indica la posición de almacenamiento en el vector de hoja.

4. El aparato (10, 20) de búsqueda como se reivindica en la reivindicación 3, en el que se configuran los medios (11) de cálculo para verificar que el vector de bit está primero entre el vector de bit y el vector de hoja, y utilizar el vector de hoja en base al valor del bit del vector de bit.

5. El aparato (10, 20) de búsqueda como se reivindica en la reivindicación 1, en el que, en cuanto a cada nodo interno en los datos de objetivo de búsqueda, los nodos de hoja que se convierten en destinos de transición se almacenan en la matriz de nodos de hoja en la que son consecutivas las posiciones de almacenamiento, se comprimen nodos de hoja que tienen el mismo valor, y una pluralidad de nodos de hoja no incluye nodos de hoja que tienen el mismo valor,

cada nodo interno en los datos de objetivo de búsqueda incluye un vector de máscara que tiene un bit que indica una posición de almacenamiento en la que se cambia un valor de un nodo de hoja antes de compresión, y en el que

se configuran los medios (11) de cálculo, cuando el destino de transición determinado en base al valor del bit del vector de bit es un nodo de hoja, para acceder al nodo de hoja del destino de transición utilizando la segunda información base y el número de bits que indica los nodos de hoja en el vector de bit enmascarado por el vector de máscara.

6. El aparato (10, 20) de búsqueda como se reivindica en la reivindicación 1, en el que, en cuanto a cada nodo interno en los datos de objetivo de búsqueda, los nodos internos que se convierten en destinos de transición se almacenan en la matriz de nodos en la que son consecutivas las posiciones de almacenamiento, se comprimen los nodos internos que tienen el mismo valor, y una pluralidad de nodos internos no incluye nodos internos que tienen el mismo valor,

cada nodo interno en los datos de objetivo de búsqueda incluye un vector de máscara que tiene un bit que indica una posición de almacenamiento en la que cambia un valor de un nodo interno antes de compresión, y en el que se

configuran los medios (11) de cálculo, cuando el destino de transición determinado en base al valor del bit del vector de bit es un nodo interno, para acceder al nodo interno del destino de transición utilizando la primera información base y el número de bits que indica los nodos internos en el vector de bit enmascarado por el vector de máscara.

7. El aparato (10, 20) de búsqueda como se reivindica en la reivindicación 1, en el que, en cuanto a cada nodo interno en los datos de objetivo de búsqueda, se comprimen nodos de hoja que tienen el mismo valor después de que se enmascara un valor predeterminado en los nodos de hoja que se convierten en destinos de transición y el valor enmascarado se cambia a un valor diferente, de tal manera que una pluralidad de nodos de hoja no incluye nodos de hoja que tienen el mismo valor, y se almacena en la matriz de nodos de hoja en la que son consecutivas las posiciones de almacenamiento,

cada nodo interno en los datos de objetivo de búsqueda incluye el valor predeterminado enmascarado, una máscara de hoja que tiene un bit que indica una posición del vector de hoja que tiene el valor predeterminado antes de compresión, y un vector de hoja que tiene un bit que indica una posición de almacenamiento en la que se cambia un valor de un nodo de hoja antes de compresión, y en el que

se configuran los medios (11) de cálculo, cuando el destino de transición determinado en base al valor del bit del vector de bit es un nodo de hoja,

para determinar si un bit se establece en la máscara de hoja en una posición igual a la del bit en el vector de bit, obtener el valor predeterminado como un valor del nodo de hoja del destino de transición cuando se establece el bit, y acceder al nodo de hoja del destino de transición utilizando la segunda información base y el número de bits que indica la posición de almacenamiento en el vector de hoja cuando no se establece el bit.

8. El aparato (10, 20) de búsqueda como se reivindica en una cualquiera de las reivindicaciones 2 a 7, en el que se configuran los medios (11) de cálculo para calcular el número de bits utilizando un comando popcnt de una CPU configurada por los medios (11) de cálculo.

9. El aparato (10, 200) de búsqueda como se reivindica en una cualquiera de las reivindicaciones 1 a 8, en el que los medios (11) de cálculo y los medios (12) de almacenamiento se configuran sobre una CPU de 64 bit.

10. El aparato (10, 20) de búsqueda como se reivindica en una cualquiera de las reivindicaciones 1 a 9, en el que el fragmento tiene una longitud de 6 bit, y el vector de bit tiene una longitud de 64 bit.

11. El aparato (10, 20) de búsqueda como se reivindica en una cualquiera de las reivindicaciones 2 a 7, en el que los medios (11) de cálculo y los medios (12) de almacenamiento se configuran sobre una CPU de 64 bit, el fragmento tiene una longitud de 6 bit, y el vector de bit tiene una longitud de 64 bit, y en el que

se configuran los medios (11) de cálculo para calcular el número de los bits utilizando un comando popcnt de la CPU de 64 bit, y para acceder a un nodo del destino de transición utilizando una compensación, a partir de la información base, en base al número de los bits.

12. El aparato (10, 20) de búsqueda como se reivindica en una cualquiera de las reivindicaciones 1 a 11, en el que se configuran los medios (11) de cálculo para obtener un fragmento de una longitud de bit mayor que la longitud de bits predeterminada a partir de los datos clave, y alcanzar directamente un nodo de hoja al realizar la búsqueda utilizando el fragmento.

13. Un programa que hace que un ordenador ejecute instrucciones que hacen que el ordenador funcione de acuerdo con el aparato (10, 20) de búsqueda como se reivindica en una cualquiera de las reivindicaciones 1 a 12.

14. Un medio de registro legible por ordenador configurado para almacenar instrucciones de acuerdo con la Reivindicación 13 y configurado adicionalmente para almacenar los datos de objetivo de búsqueda como se reivindica en una cualquiera de las reivindicaciones 1 a 12.

15. Un método de búsqueda para realizar el procesamiento de búsqueda sobre los datos de objetivo de búsqueda en base a datos clave, en los que los datos de objetivo de búsqueda son los datos de una estructura de árbol de múltiples vías que incluyen una matriz de nodos internos y una matriz de nodos de hoja,

caracterizado porque:

cada nodo interno en los datos de objetivo de búsqueda incluye un vector de bit que representa si un destino de transición es un nodo interno o un nodo de hoja por un bit, primera información base que indica una posición de almacenamiento de un nodo interno de un destino de transición, y segunda información base que indica una posición de almacenamiento de un nodo de hoja de un destino de transición,

el método de búsqueda comprende: ejecutar repetidamente, hasta que un nodo de transición se convierte en un nodo de hoja, el procesamiento incluye

obtener un fragmento de una longitud de bits predeterminada a partir de los datos clave,

determinar si un destino de transición del nodo interno es un nodo interno o un nodo de hoja en base a un bit, en el vector de bit del nodo interno de acceso, que corresponde a un valor del fragmento, y

cuando el destino de transición determinado en base al valor del bit del vector de bit es un nodo interno, acceder al nodo interno del destino de transición utilizando la primera información base, y cuando el destino de transición es un nodo de hoja, acceder al nodo de hoja del destino de transición utilizando la segunda información base.

FIG.1

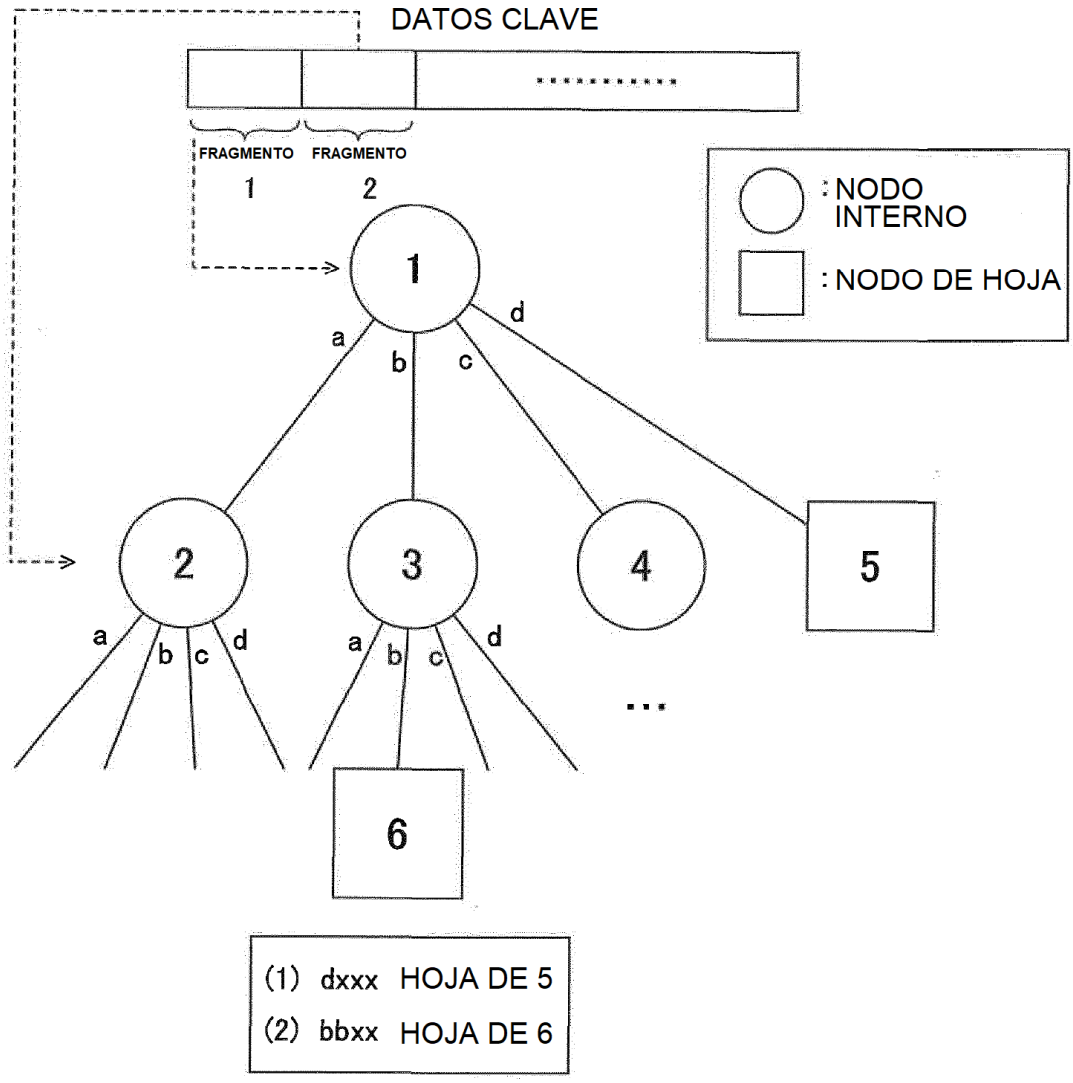

DATOS CLAVE
FRAGMENTO
FRAGMENTO
1
2
: NODO INTERNO
: NODO DE HOJA
1
2
3
4
5
6
a
b
c
d
(1) dxxx HOJA DE 5
(2) bbxx HOJA DE 6

FIG.2

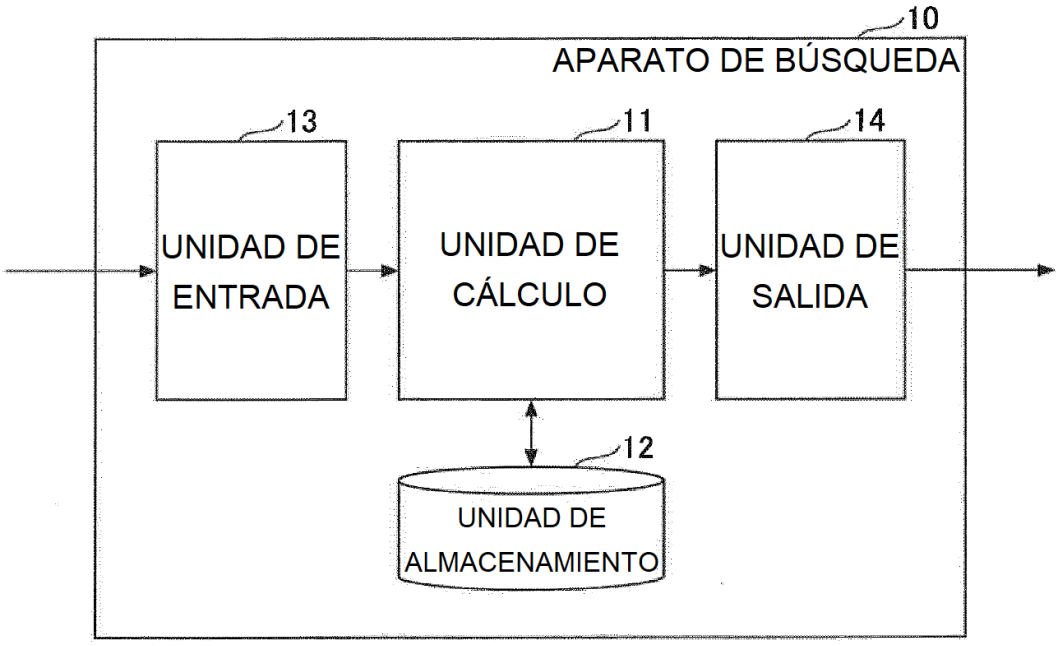

10
APARATO DE BÚSQUEDA
13
UNIDAD DE ENTRADA
11
UNIDAD DE CÁLCULO
14
UNIDAD DE SALIDA
12
UNIDAD DE ALMACENAMIENTO

FIG.3

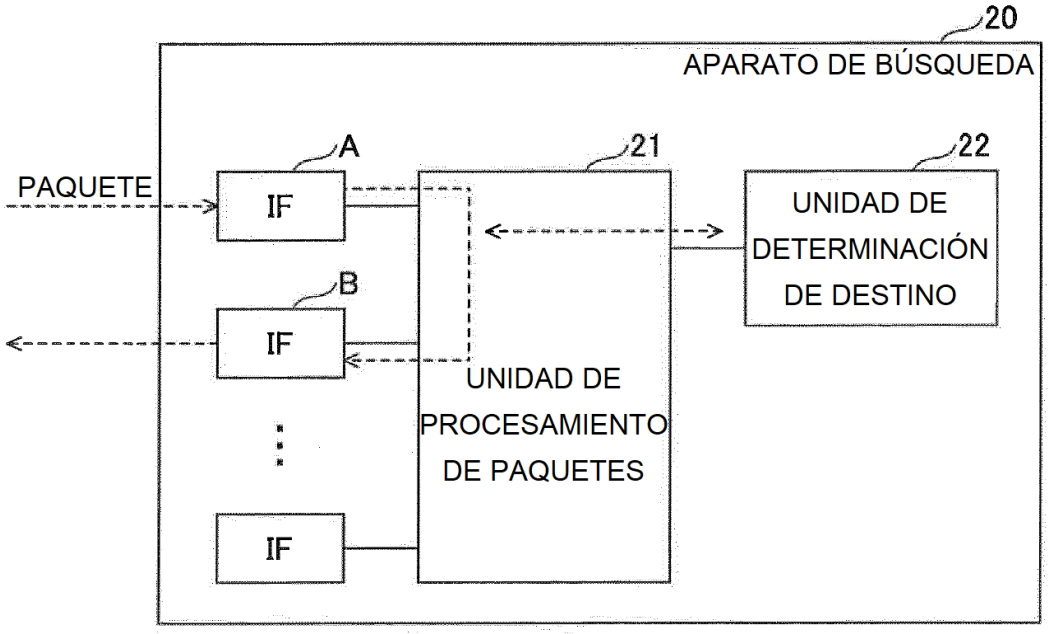

20
APARATO DE BÚSQUEDA
PAQUETE
A
IF
B
IF
IF
21
UNIDAD DE PROCESAMIENTO DE PAQUETES
22
UNIDAD DE DETERMINACIÓN DE DESTINO

FIG.4

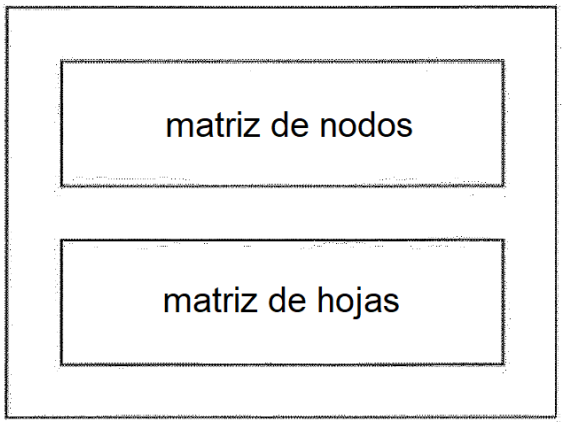

matriz de nodos
matriz de hojas

FIG.5

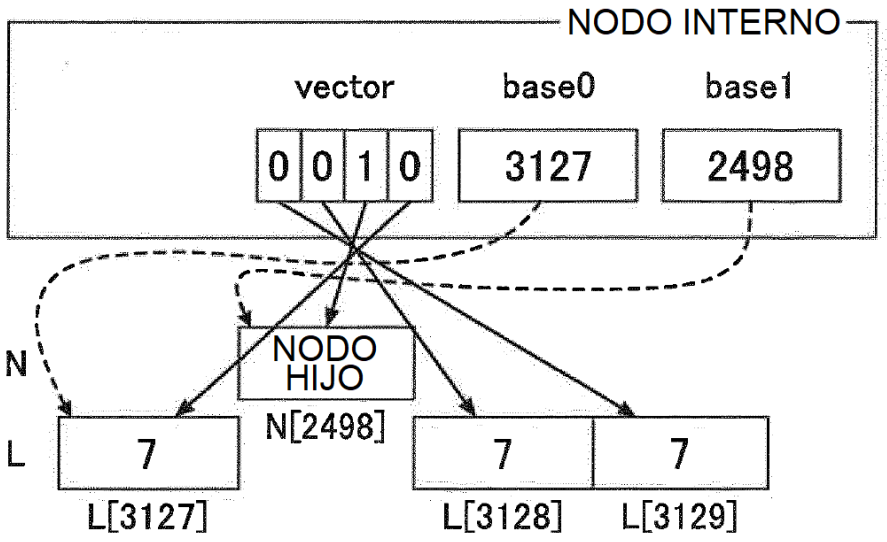

NODO INTERNO
vector
base0
base1
0 0 1 0
3127
2498
N
NODO HIJO
N[2498]
L
7
7
7
L[3127]
L[3128]
L[3129]

FIG.6

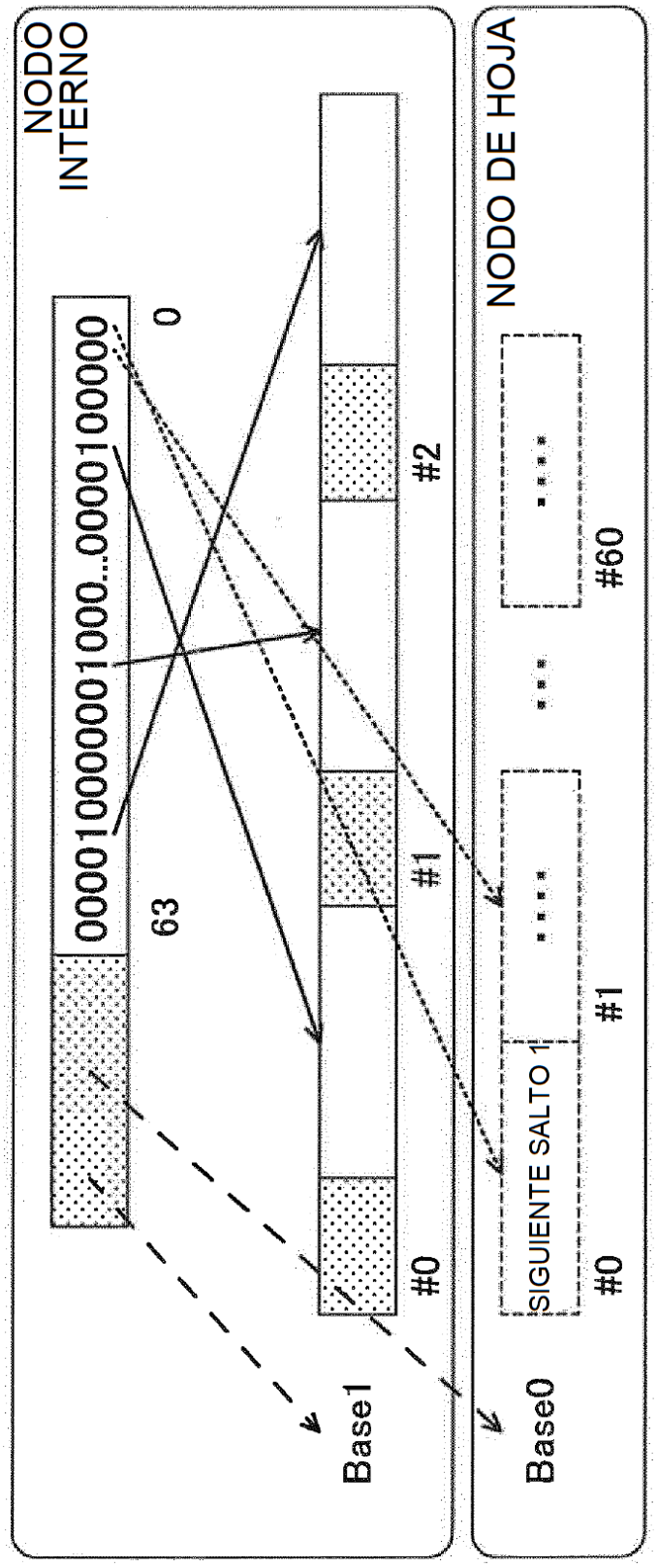

NODO INTERNO
NODO DE HOJA
0
63
00001000000010000...0000100000
#2
#1
#0
Base1
Base0
SIGUIENTE SALTO 1
#60
#1
#0

FIG.7

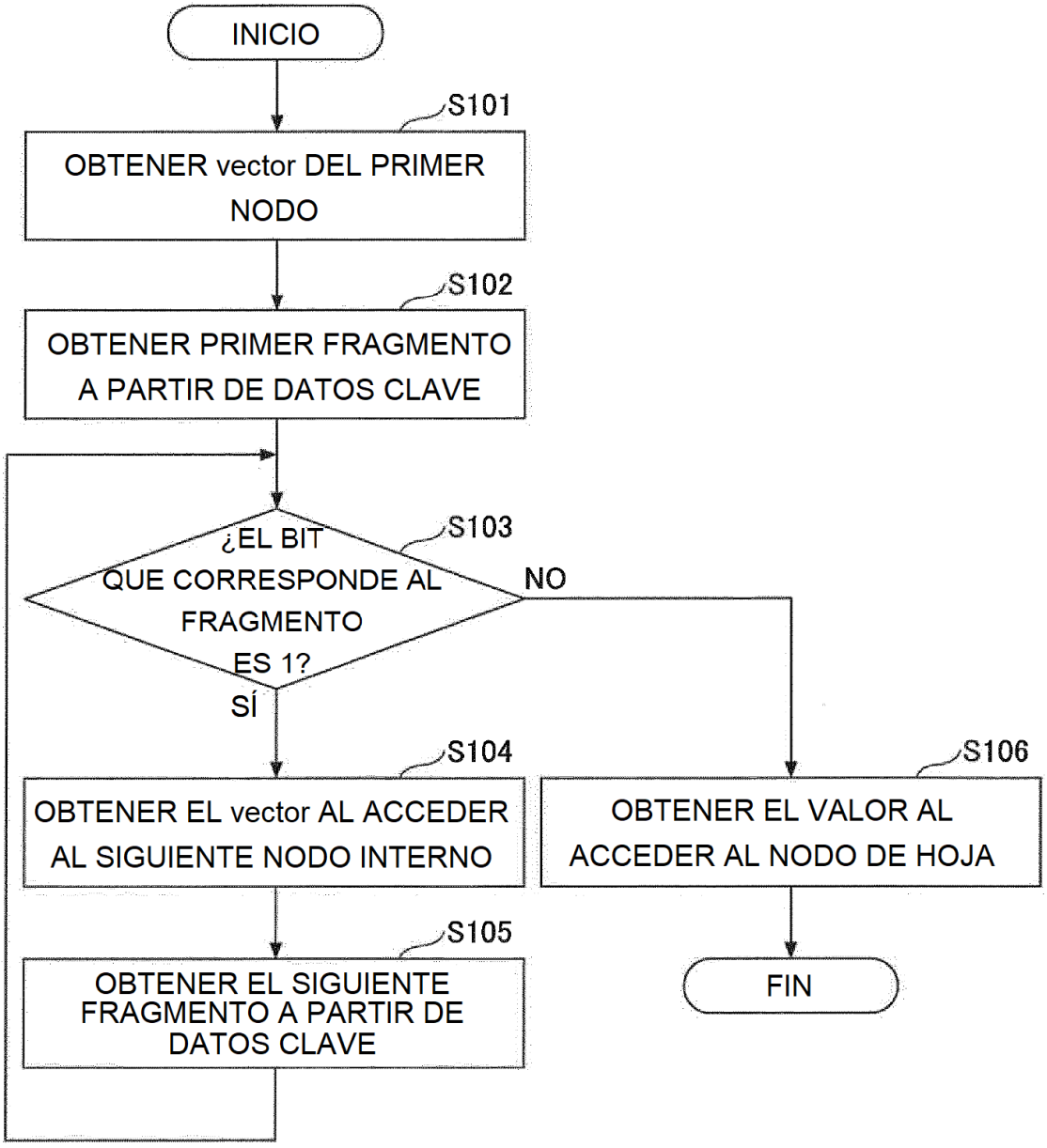

INICIO
S101
OBTENER vector DEL PRIMER NODO
S102
OBTENER PRIMER FRAGMENTO A PARTIR DE DATOS CLAVE
S103
¿EL BIT QUE CORRESPONDE AL FRAGMENTO ES 1?
NO
SÍ
S104
OBTENER EL vector AL ACCEDER AL SIGUIENTE NODO INTERNO
S105
OBTENER EL SIGUIENTE FRAGMENTO A PARTIR DE DATOS CLAVE
S106
OBTENER EL VALOR AL ACCEDER AL NODO DE HOJA
FIN

FIG.8A

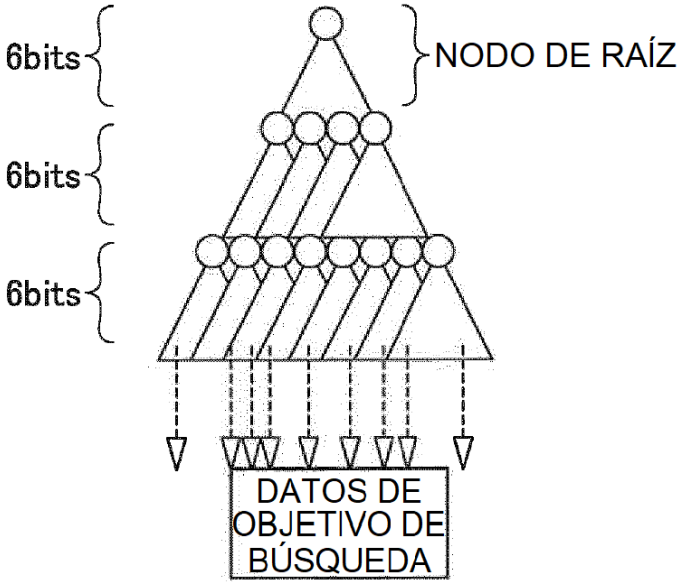

6bits
6bits
6bits
NODO DE RAÍZ
DATOS DE OBJETIVO DE BÚSQUEDA

FIG.8B

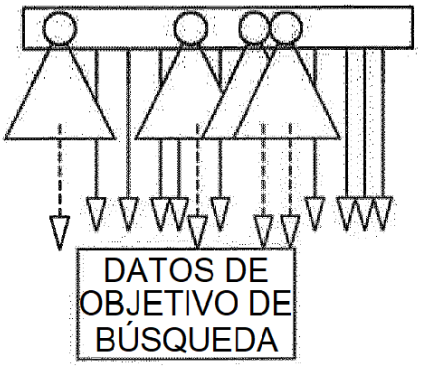

DATOS DE OBJETIVO DE BÚSQUEDA

FIG.9

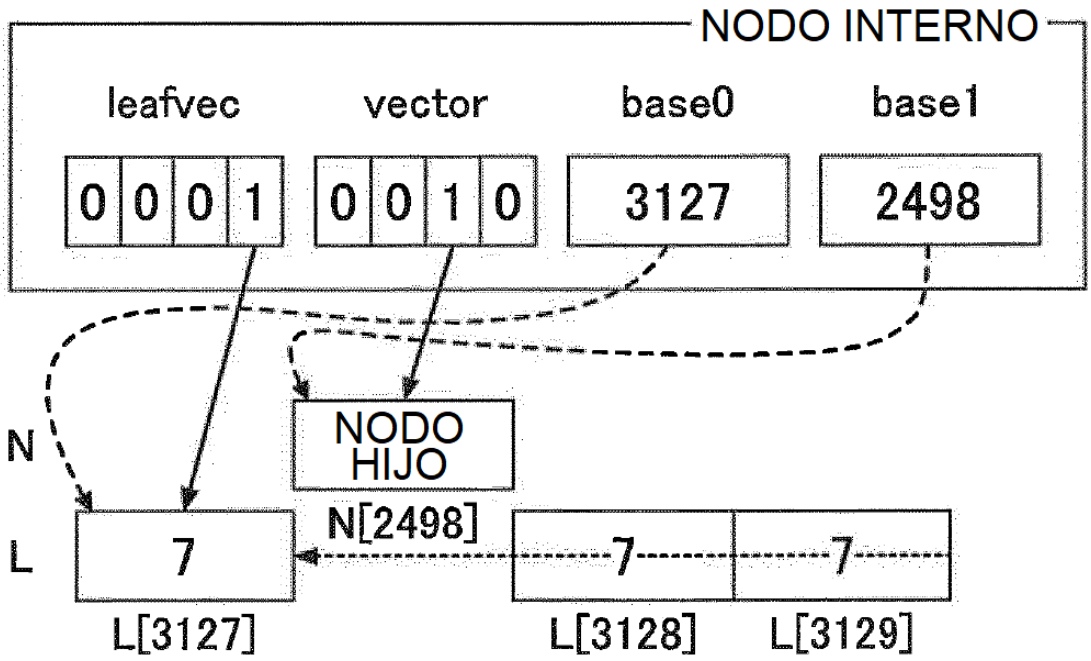

NODO INTERNO
leafvec
vector
base0
base1
0 0 0 1
0 0 1 0
3127
2498
N
NODO HIJO
N[2498]
L
7
7
7
L[3127]
L[3128]
L[3129]

FIG.10

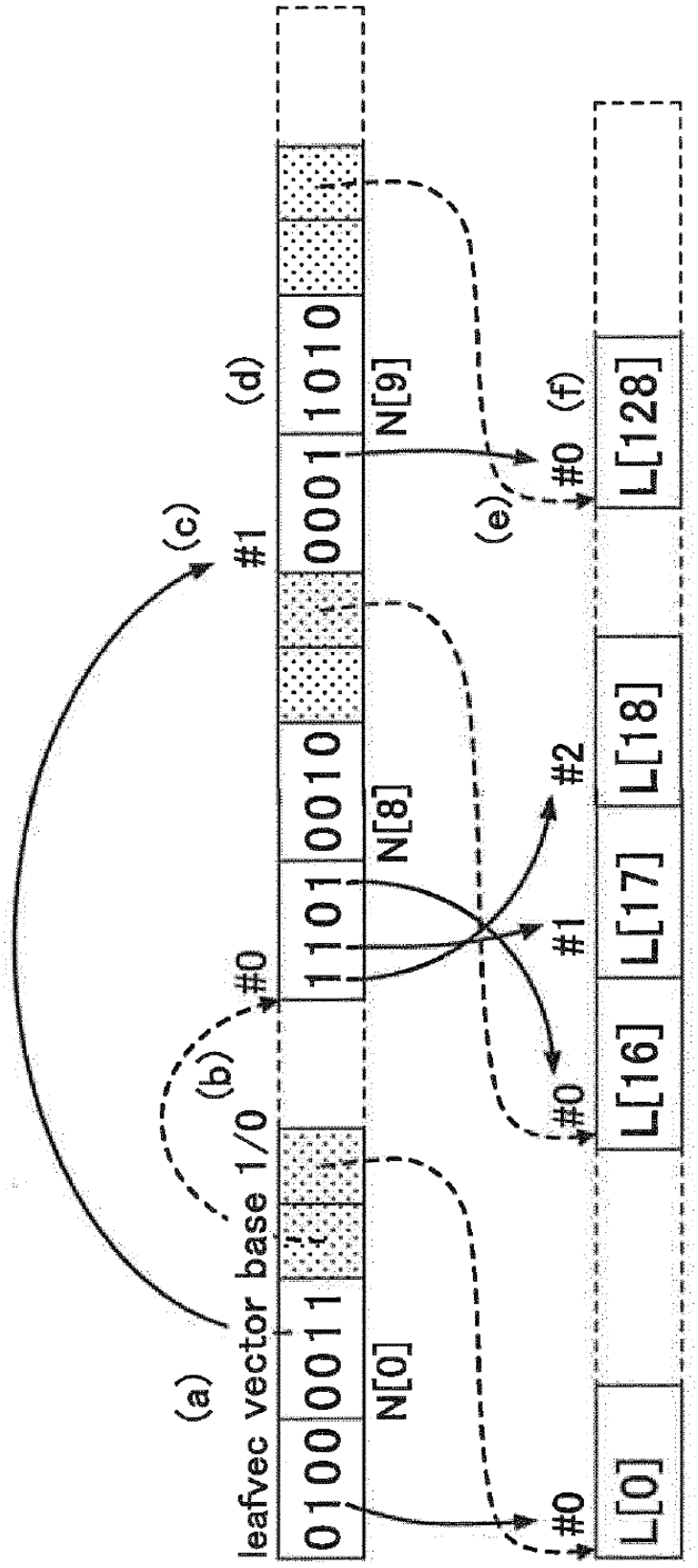

leafvec
vector
base
1/0
0100
0011
1101
0010
0001
1010
N[0]
N[8]
N[9]
L[0]
L[16]
L[17]
L[18]
L[128]
#0
#1
#2
(a)
(b)
(c)
(d)
(e)
(f)

FIG.11

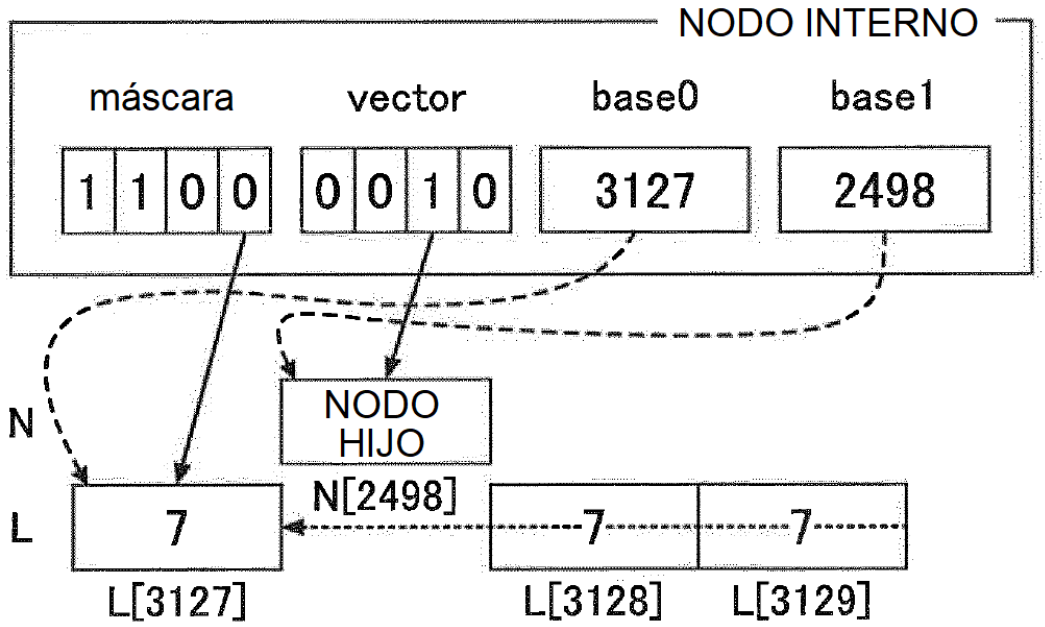

NODO INTERNO
máscara
vector
base0
base1
1 1 0 0
0 0 1 0
3127
2498
N
NODO HIJO
N[2498]
L
7
7
7
L[3127]
L[3128]
L[3129]

FIG.12

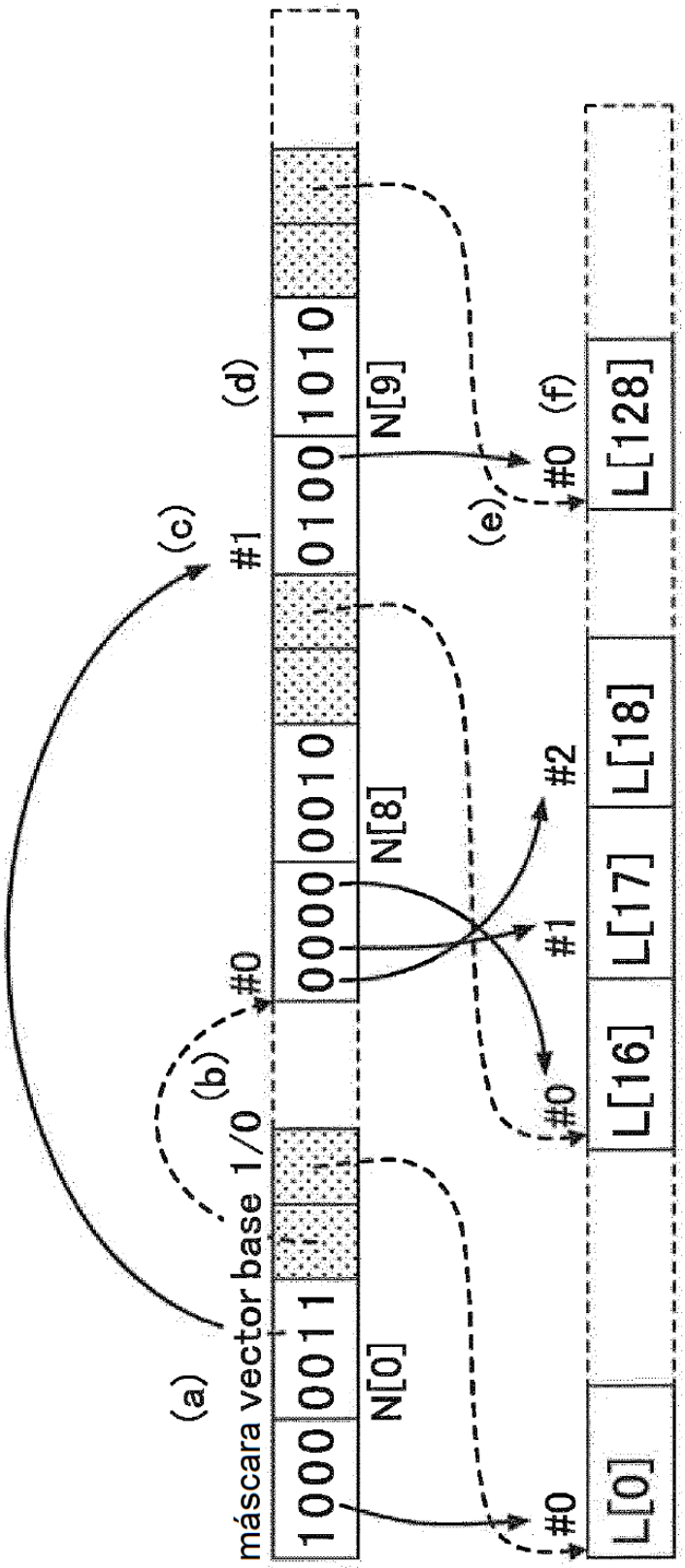

máscara vector base 1/0
1000
0011
0000
0010
0100
1010
N[0]
N[8]
N[9]
(a)
(b)
(c)
(d)
(e)
(f)
#0
#1
#2
#0
#1
#0
#0
L[0]
L[16]
L[17]
L[18]
L[128]

FIG.13

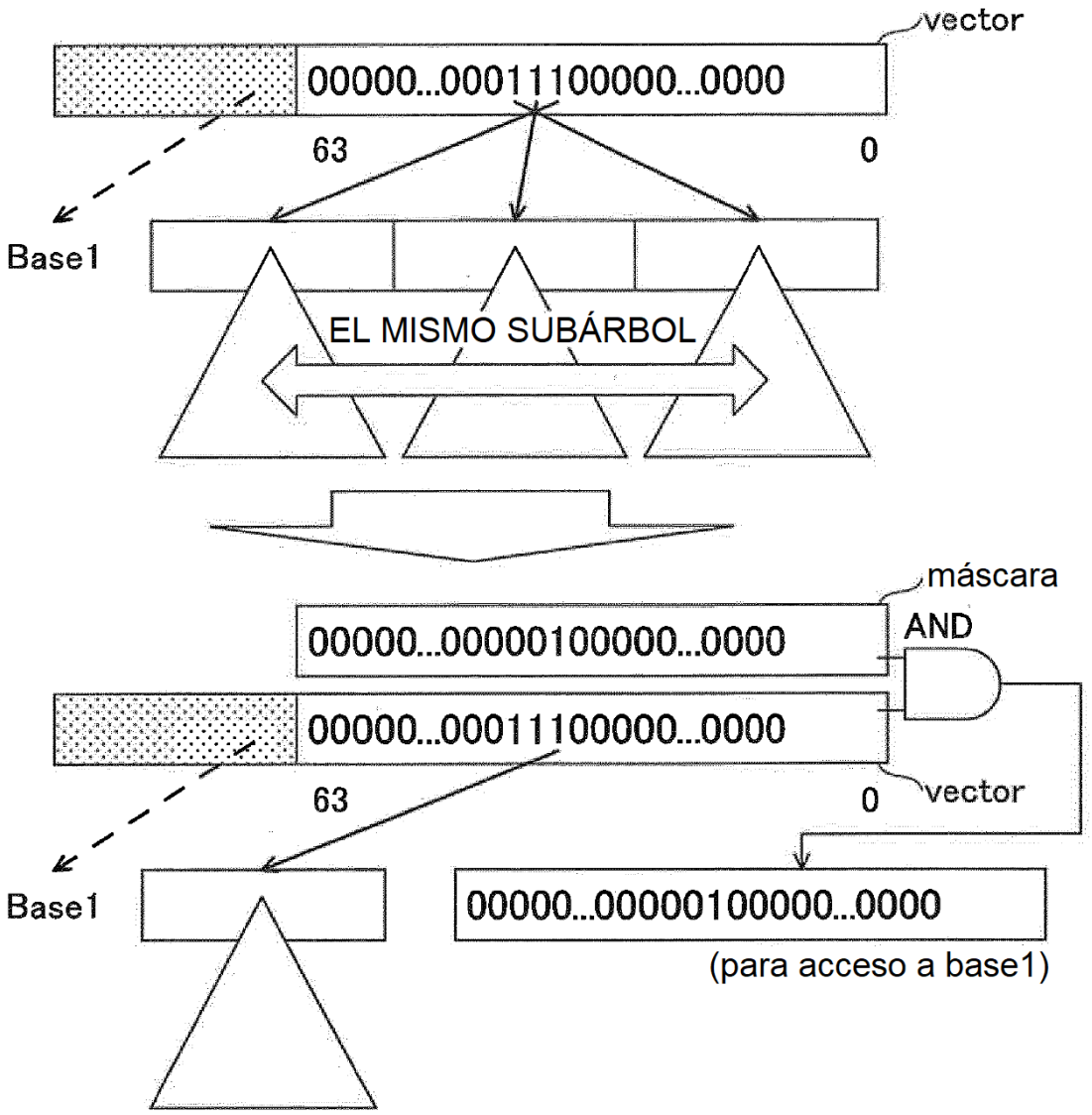

vector
00000...00011100000...0000
63
0
Base1
EL MISMO SUBÁRBOL
máscara
AND
00000...00000100000...0000
00000...00011100000...0000
63
0
vector
Base1
00000...00000100000...0000
(para acceso a base1)

FIG.14

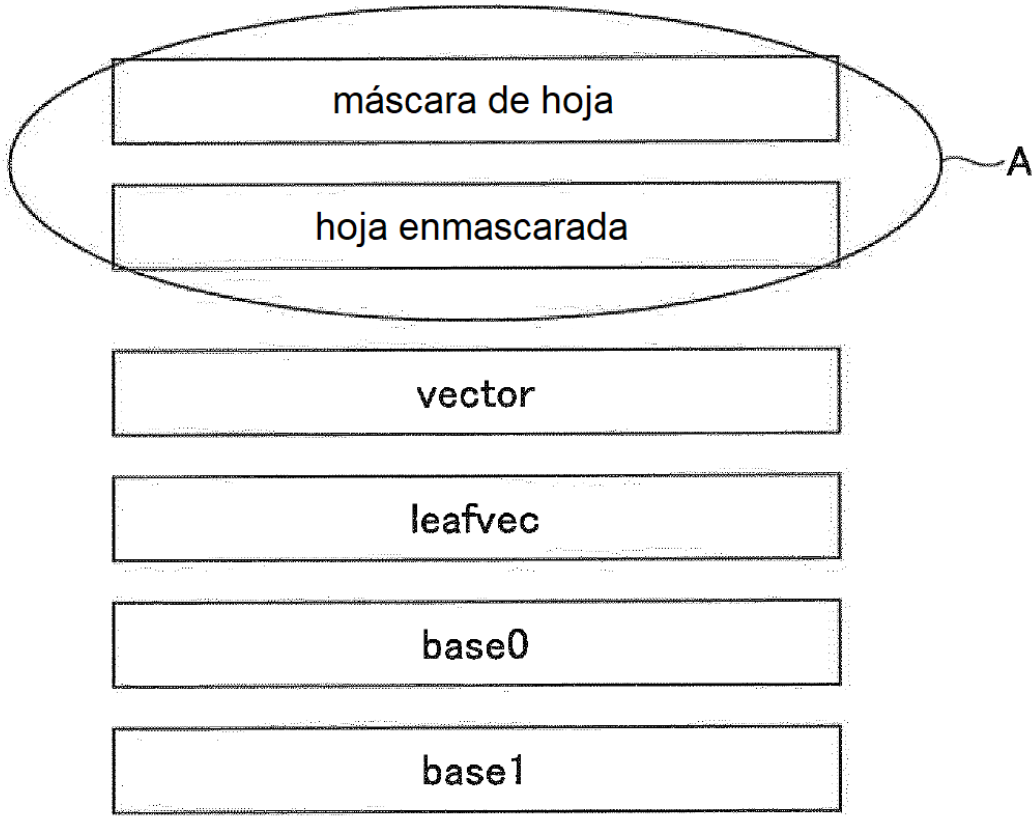

máscara de hoja
hoja enmascarada
A
vector
leafvec
base0
base1

FIG.15

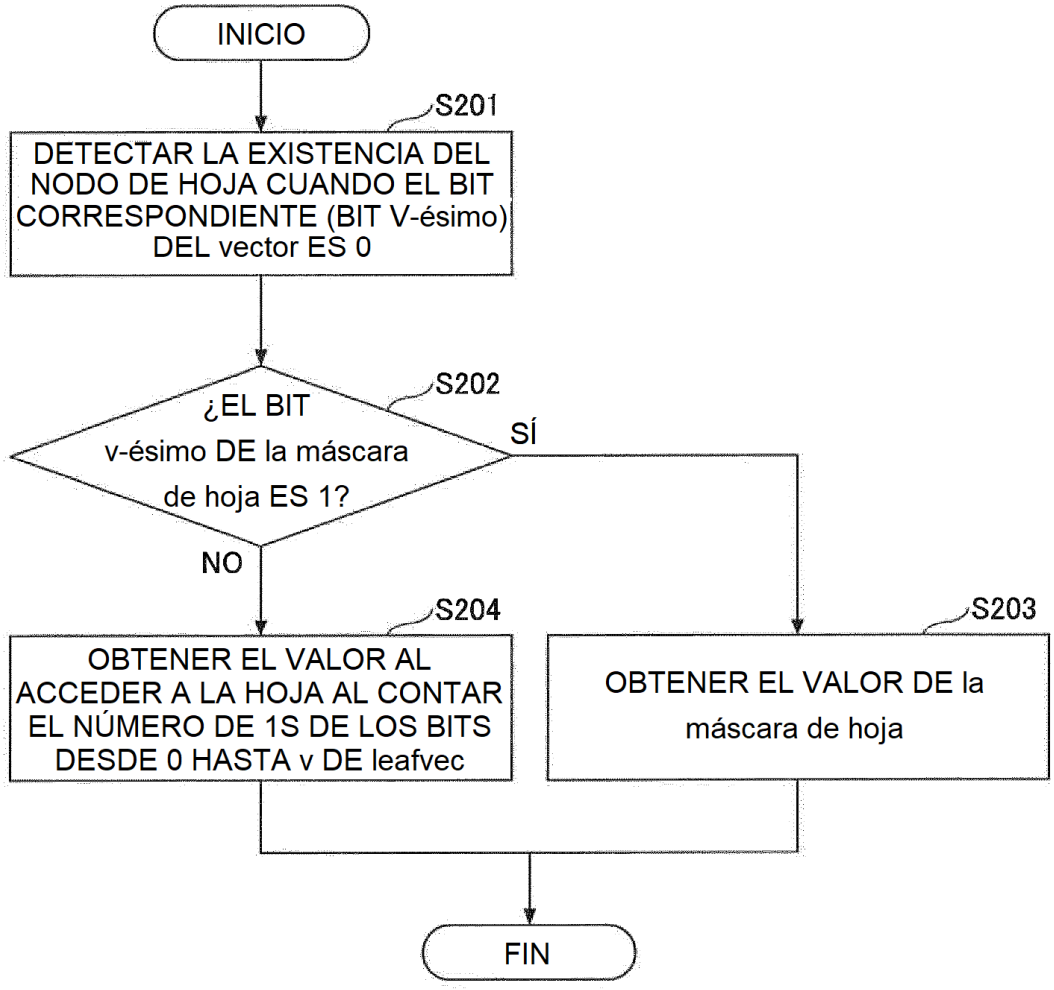
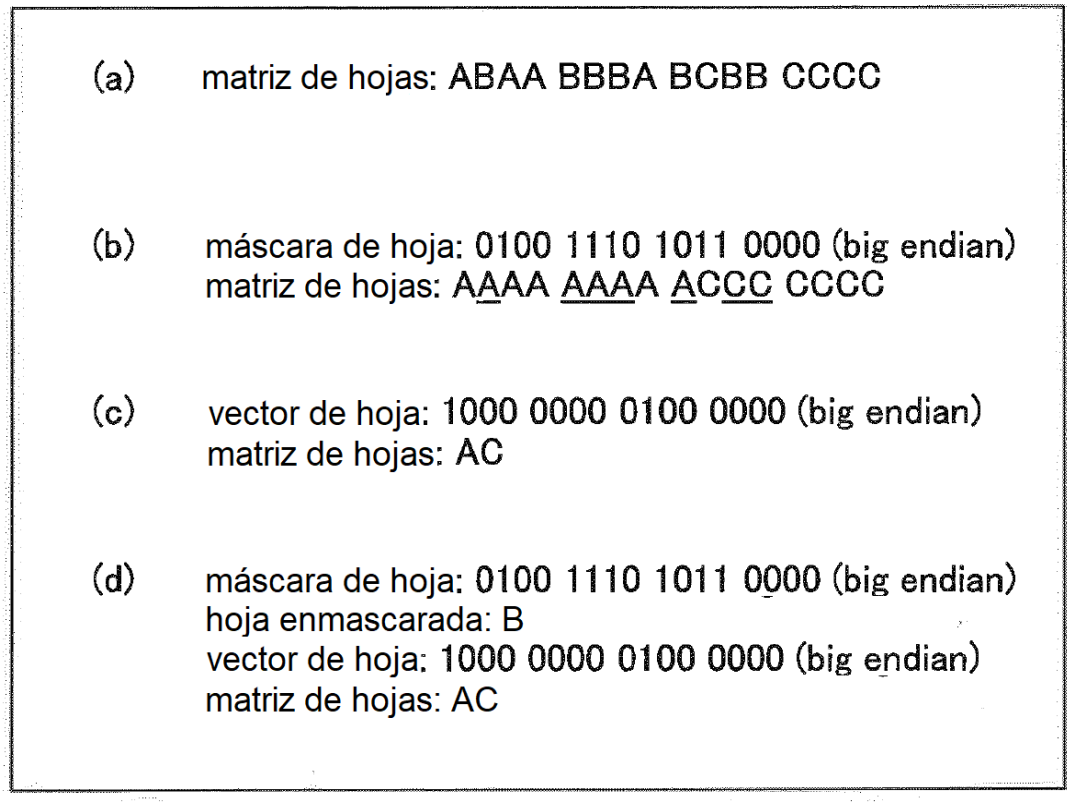

INICIO
S201
DETECTAR LA EXISTENCIA DEL NODO DE HOJA CUANDO EL BIT CORRESPONDIENTE (BIT V-ésimo) DEL vector ES 0
S202
¿EL BIT v-ésimo DE la máscara de hoja ES 1?
SÍ
NO
S204
OBTENER EL VALOR AL ACCEDER A LA HOJA AL CONTAR EL NÚMERO DE 1S DE LOS BITS DESDE 0 HASTA v DE leafvec
S203
OBTENER EL VALOR DE la máscara de hoja
FIN

FIG.16


(a) matriz de hojas: ABAA BBBA BCBB CCCC
(b) máscara de hoja: 0100 1110 1011 0000 (big endian)
matriz de hojas: AAAA AAAA ACCC CCCC
(c) vector de hoja: 1000 0000 0100 0000 (big endian)
matriz de hojas: AC
(d) máscara de hoja: 0100 1110 1011 0000 (big endian)
hoja enmascarada: B
vector de hoja: 1000 0000 0100 0000 (big endian)
matriz de hojas: AC