Arboles de Fusión

Patrichs Inocente Valle[1], Dashiel Sanchez Ramos[1]
[1]Escuela profesional de Ciencias de la Computación.
Universidad Nacional de Ingenieria

25 de septiembre de 2018

SECCIÓN 1
Introducción

Un árbol de fusión es un tipo de estructura de datos, esencialmente, un b-tree (ver figura 1) con un factor de ramificación de w $^{1/5}$, lo que le proporciona una altura de O($\log_w$n) en el peor de los casos. Esta estructura opera en una colección de n pares clave-valor,modelo que le permite usar O(n) espacio y realizar búsquedas en O($\log_w$n) tiempo, que es asintóticamente más rápido que un árbol de búsqueda binaria autoequilibrante tradicional.

SECCIÓN 2
¿Cómo logramos un tiempo de O($\log_w$n) para una busqueda?

Para lograr los tiempos de ejecución deseados para una búsqueda o consulta, el árbol de fusión debe poder encontrar un nodo que contenga hasta w$^{1/5}$ claves en tiempo constante,es decir, puede leer y comparar una palabra de w bits con otra de igual tamaño en un tiempo constante.

Sin embargo, esto parece ser imposible,ya que la construcción inherente de un b-tree exige la lectura y comparación de nuetra consulta con k=w $^{1/5}$ claves de w bits (en cada nodo),es decir un tamaño de O(w)xO(w$^{1/5}$)=O(w$^{6/5}$) para la lectura,claramente esto es mayor a O(w) y como consecuencia nuestro tiempo de busqueda no podra ser constante.

Por lo tanto, buscaremos la forma de comprimir nuestros bits y lograr una cantidad igual o menor a O(w) bits de lectura en cada comparación que se realize. Este proceso estará facilitado por el método sketching,que veremos a continuación.

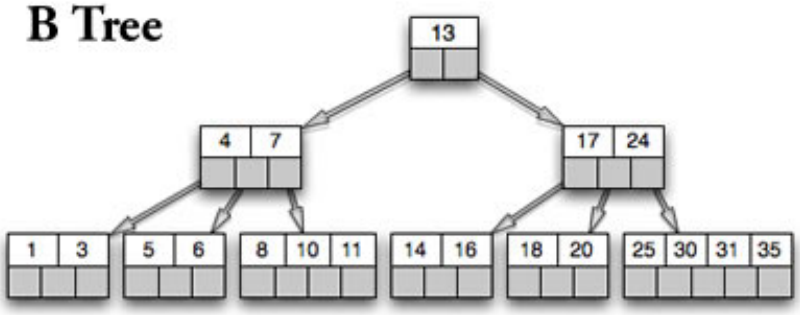

Figura 1: Ejemplo de un arbol b-tree

SECCIÓN 3

Sketching (bosquejo)

Si a cada clave de un nodo la llevamos a una representación binaria generará un dibujo o bosquejo(una rama de tamaño w), si solapamos todos estos bosquejos para las k claves de un nodo obtendremos un arbol binario de profundidad w (ver figura 2).

Las claves son representadas por x_0,x_1,....,x_{k-1}; cada clave x_i se puede representar como una ruta, donde una elección de rama está determinado por un bit, siendo 1 cuando una rama derecha es escogida y 0 en el caso contrario. Cada clave puede ser diferenciada de las otras mediante la lectura de un número menor de bits del que realmente expresa su ruta, para ser precisos a lo mucho por k-1 bits.

Esto se debe principalmente a que si optamos por una estructura,que contiene solo dos claves, existirá solo un nodo de bifurcación, luego si se induce que para estructuras más grandes,existirán k-1 nodos de bifurcación por las k claves a evaluar, se dice que este valor puede ser menor ya que la presencia de dos nodos de bifurcación en el mismo nivel puede ser diferenciado por un solo bit. **Por ello el método de Shetching nos brinda la posibilidad de extraer solo los bits relevantes para la diferenciación de nuestras claves y encontrar un predecesor/sucesor en tiempo constante.**

Los bits que son importantes para diferenciar una clave de otra son justamente los que corresponden los niveles donde se encuentra un nodo de bifurcación. El esbozo perfecto de una clave se obtiene al extraer los bits relevantes según nuestro análisis, la posición de estos bits es guardada en un vector b para no perderlos de vista en futuros procedimientos,luego esbozo perfecto se convertirá en una palabra de al menos k-1 bits donde la i-esima posición del sketch perfecto será llenado con la b-iésima posición de la clave. Cabe destacar que las operaciones AC^0 nos daran el sketch perfecto de una palabra cualquiera,sin embargo usaremos otro método para obtener un resultado aproximado.

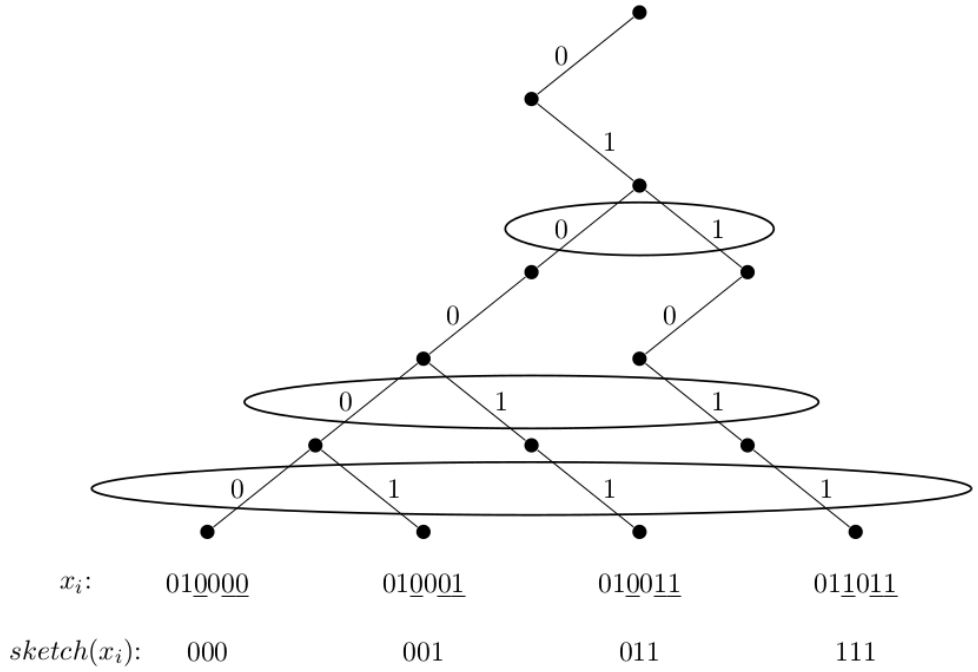

Figura 2: Se muestra la obtención del bosquejo de las claves de un nodo,generandose el arbol binario correspondiente.

SECCIÓN 4

Desketchifying (Desboquejar)

La búsqueda de un predecesor/sucesor en el bosquejo solo funciona correctamente para una consulta q que es igual a alguna de las claves pero es errónea para una consulta diferente (ver figura 3). Esto es debido a que la consulta q, no fue tomada en cuenta cuando se obtuvo la función sketch() donde solo se trabajo con las claves, por lo que procederemos a corregir el error, mediante un proceso llamado Desketchifying. El proceso Desketchifying consiste en: 1. Buscar una cadena de bits y, la cual es igual al prefijo común más largo entre las rutas de la nueva consulta q y una clave(la más cercana) pertenecientes a un mismo subárbol. Cabe destacar que la cadena 'y' nos muestra la ruta hasta llegar al nodo de bifurcación que no se tomó en cuenta para la consulta q. 2. Para hallar el predecesor/sucesor de q, hallamos el mayor/menor elemento 'e' (más a la derecha/izquierda) de la rama izquierda/derecha desde el punto de bifurcación hallado. e=y011..1/e=y1000..0. 3. Finalmente hallamos sketch(e), el cual coincide con el sketch del predecesor/sucesor de q.!El error ha sido corregido!

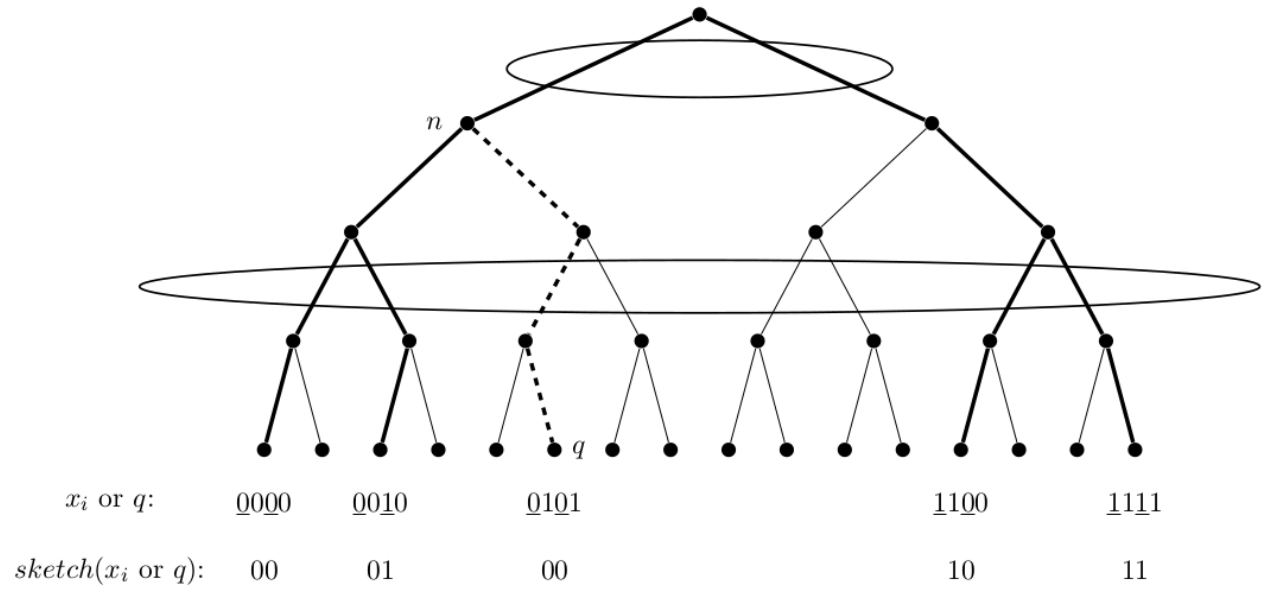

Figura 3: Se muestra la obtención de sketch de nuestras claves xi y una consulta q arbitraria.

Referencias

[1] Demaine, Erik. [MIT OpenCourseWare]. (2013, Diciembre 18). 12. Fusion Trees [Archivo de video]. Recuperado de http://www.youtube.com/watch?v=6nyGCbxD848

[2] Demaine, Erik. [MIT OpenCourseWare]. (2012, Abril 03). Lecture 12, Fusion Trees. Advanced Data Structures [Archivo de texto]. 1- 8 pp. Recuperado de http://ocw.mit.edu