Algoritmos y Estructuras de Datos I

Universidad Católica del Uruguay

Resumen de clase

Jueves 10 de junio de 2021

EFICIENCIAS DE LAS BÚSQUEDAS

- En lista
- En árbol
- Se tiene o no se tiene la frecuencia con la que se accede a cada clave.

EL CASO DE UNA LISTA

- Supongamos que tenemos un conjunto de “n” palabras claves, desde la k_1 hasta la k_n
- Si no tenemos información sobre la frecuencia de acceso a las claves, se asume que todas tienen la misma probabilidad de ser accedidas, es decir $p_i = 1/n$ para todo i
- Por lo tanto, da lo mismo la posición “i” en que quede insertada cada una de las claves en la lista, ya que la cantidad promedio de comparaciones que deben realizarse para resolver una búsqueda exitosa es:

$$Cp = \sum_{i=1}^{n} i * p_i = \frac{1}{n} * \sum_{i=1}^{n} i = \frac{1}{n} * \frac{(n+1)n}{2} = \frac{n+1}{2}$$

- Para resolver una búsqueda infructuosa, debe llegarse hasta el final de la lista, o sea se realizarán n + 1 comparaciones.

FRECUENCIA DE ACCESO A LAS CLAVES

- Hasta ahora hemos supuesto que todas las claves tienen la misma probabilidad de ser accedidas.
- Sin embargo, en muchas aplicaciones esto no es cierto.
- Sea por ejemplo el analizador sintáctico de un lenguaje de programación. Cada vez que se lee una palabra del código fuente, el analizador debe verificar que esa palabra no sea una palabra reservada del lenguaje. Por cada palabra leída, va a buscar si esa palabra está o no en el conjunto de palabras reservadas del lenguaje.
- Es indudable que no todas las palabras aparecen con la misma frecuencia en los programas fuentes de un lenguaje.
- Por ejemplo en Java, sean “if”, que indudablemente aparece muchas veces en cualquier programa fuente, y “goto” que seguramente aparezca muy rara vez en algún programa fuente.

EL CASO DE UNA LISTA

- Supongamos que tenemos un conjunto de palabras claves cada una con su probabilidad de ser accedida. Sea p_i la probabilidad de ser accedida de la clave k_i.

  Sean k_1 = case, k_2 = goto y k_3 = if

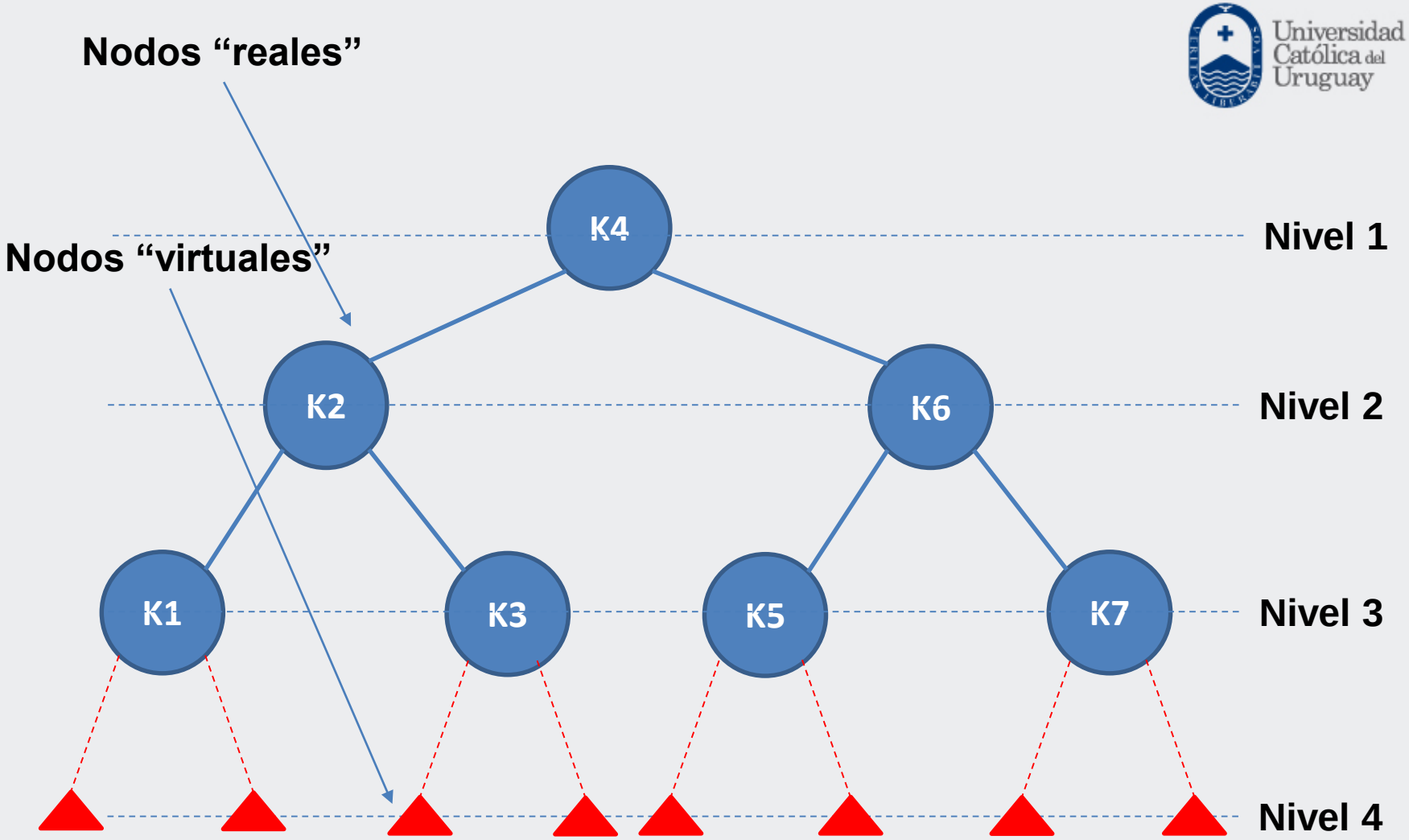
  Sean $p_1 = 0{,}1$, $p_2 = 0{,}05$ y $p_3 = 0{,}85$
- Obviamente en la posición 1 se debe poner la de mayor probabilidad, y en la posición 3 la de menor probabilidad, de forma que si la lista queda L = if, case, goto
- La cantidad de comparaciones queda

$$Cp = 1 * 0{,}85 + 2 * 0{,}1 + 3 * 0{,}05 = 0{,}85 + 0{,}2 + 0{,}15 = 1{,}2$$

EFICIENCIA DE LA BÚSQUEDA EN LOS ÁRBOLES

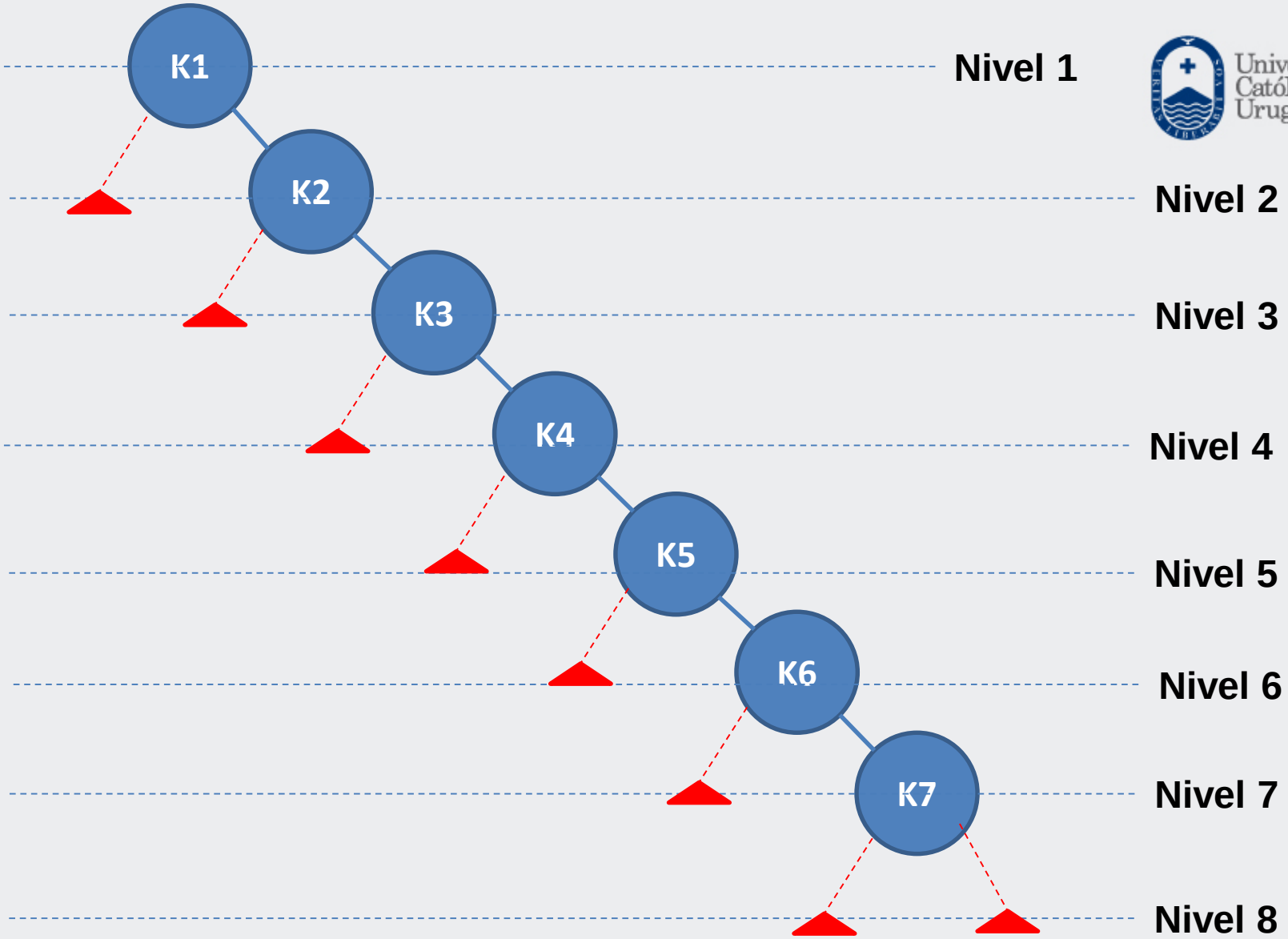
Universidad Católica del Uruguay

- En el caso de los árboles, la cantidad de comparaciones que debe realizarse para resolver la búsqueda de una clave es igual al nivel en el que se encuentra esa clave
- La LONGITUD DE TRAYECTORIA INTERNA representa el esfuerzo para resolver las búsquedas EXITOSAS
- La LONGITUD DE TRAYECTORIA EXTERNA representa el esfuerzo para resolver las búsquedas INFRUCTUOSAS



LTI = 1 + 2*2 + 4*3 = 17

LTE = 8 * 4 = 32



LTI = 1 + 2 + 3 + 4 + 5 + 6 + 7 = 28

LTE = 2 + 3 + 4 + 5 + 6 + 7 + 8 + 8 = 43

EFICIENCIA DE LA BÚSQUEDA EN LOS ÁRBOLES

Universidad Católica del Uruguay

- La LONGITUD DE TRAYECTORIA INTERNA PROMEDIO se obtiene dividiendo la LTI por el tamaño.
- La LONGITUD DE TRAYECTORIA EXTERNA PROMEDIO se obtiene dividiendo la LTE por el tamaño más uno.
- Si se conocen las frecuencias de las búsquedas, la eficiencia estará dada por la LONGITUD DE TRAYECTORIA PONDERADA

LONGITUD DE TRAYECTORIA PONDERADA

$$P = \sum_{i=1}^{N} a_i * h_i + \sum_{j=0}^{N} b_j * h'_j$$

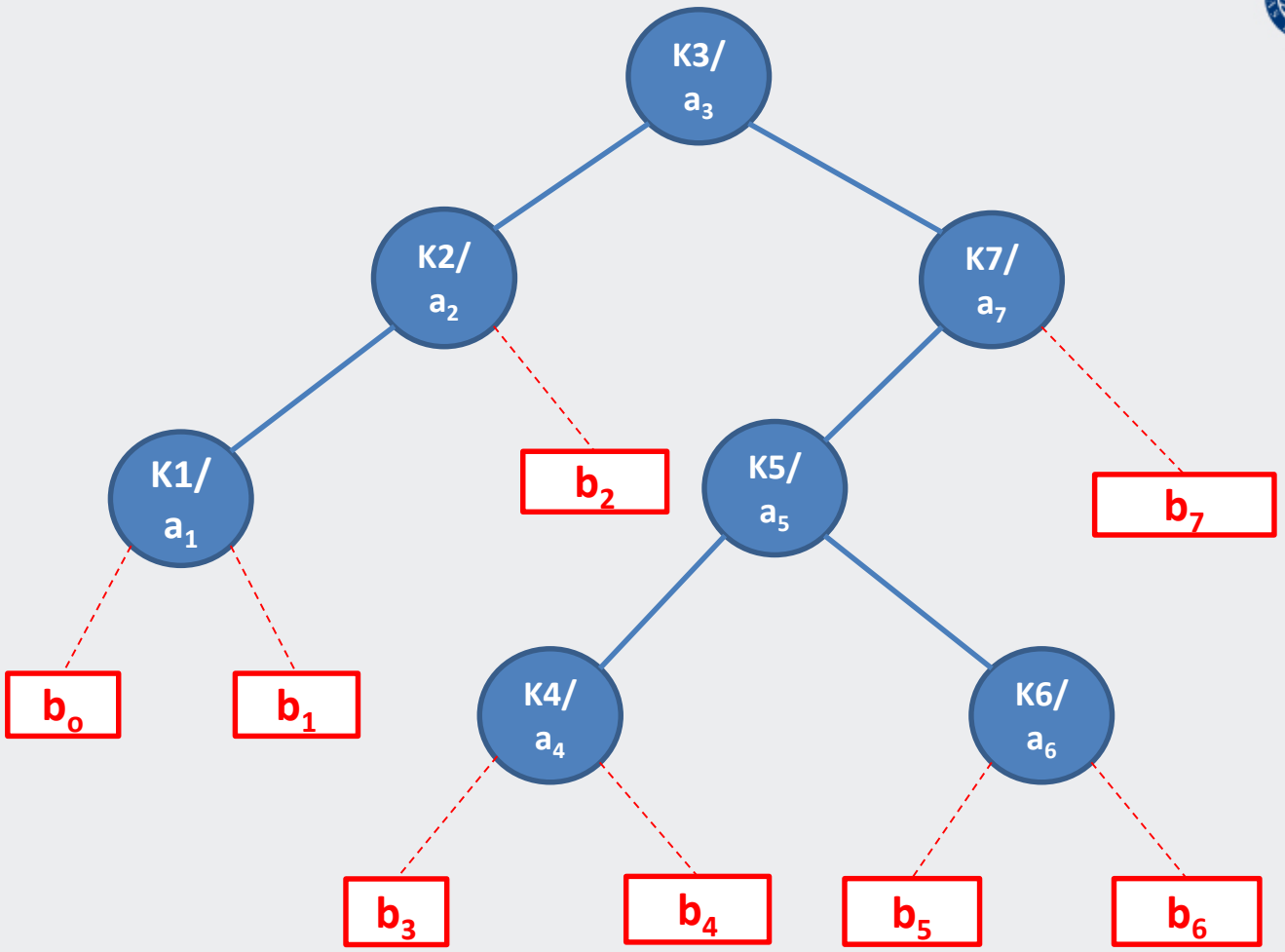
h_i es el nivel en el que se encuentra la clave de frecuencia de acceso a_i

h'_j es el nivel en el que se encuentra el nodo virtual de frecuencia de acceso b_j

EFICIENCIA DE LA BÚSQUEDA EN LOS ÁRBOLES CONOCIENDO LAS FRECUENCIAS

Universidad Católica del Uruguay

- a_i es la cantidad de veces con la que se accede a la clave K_i
- b_o es la cantidad de veces que la búsqueda fue infructuosa, y el argumento de búsqueda fue menor que K_1
- b_i es la cantidad de veces en las que la búsqueda fue infructuosa y el argumento de búsqueda estuvo entre K_i y K_{i+1} para i entre 1 y n-1
- b_n es la cantidad de veces que la búsqueda fue infructuosa, y el argumento de búsqueda fue menor que K_n


K3/
a_3
K2/
a_2
K7/
a_7
K1/
a_1
b_2
K5/
a_5
b_7
b_o
b_1
K4/
a_4
K6/
a_6
b_3
b_4
b_5
b_6

ÁRBOL BINARIO DE BÚSQUEDA ÓPTIMO

Universidad Católica del Uruguay

- De todos los árboles binarios de búsqueda que se pueden formar con “n” claves, aquél que tenga la menor longitud de trayectoria ponderada será el óptimo.