Algoritmo para la obtención de Árboles Binarios de Búsqueda Óptimos

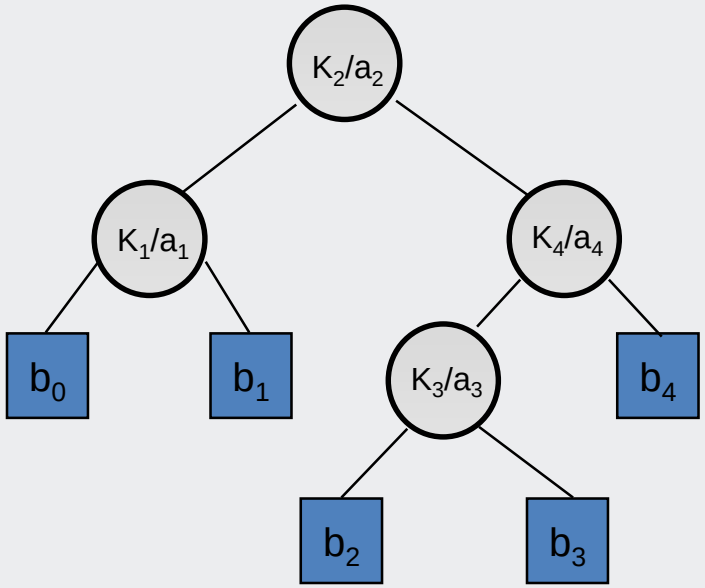
Notas del desarrollo de Niklaus Wirth de su libro Algoritmos y estructuras de datos de 1985, sección 4.6

Resumen de la clase del lunes 14 de junio de 2021

Universidad Católica del Uruguay

Árboles de búsqueda óptimos

- La longitud de trayectoria ponderada es el “costo” **P** del árbol.
- Árbol óptimo: es aquél árbol de búsqueda con costo mínimo, entre todos los posibles con determinado conjunto de claves y probabilidades.
- Para encontrar el árbol de costo mínimo no es necesario que la suma de las probabilidades p y q sea 1, sino que alcanza con utilizar conteos de frecuencia.
  - $\mathbf{a_i}$ número de veces que el argumento de búsqueda es igual a $\mathbf{K_i}$.
  - $\mathbf{b_i}$ número de veces que el argumento de búsqueda está entre $\mathbf{K_i}$ y $\mathbf{K_{i+1}}$.
  - $\mathbf{b_0}$ número de veces que el argumento de búsqueda es $\mathbf{< K_1}$.
  - $\mathbf{b_N}$ número de veces que el argumento de búsqueda es $\mathbf{> K_N}$.



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

El árbol es óptimo cuando este valor es mínimo

$$W = \sum_{i=1}^{N} a_i + \sum_{j=0}^{N} b_j$$

Este valor también es importante en el algoritmo

Árboles de búsqueda óptimos

Universidad Católica del Uruguay

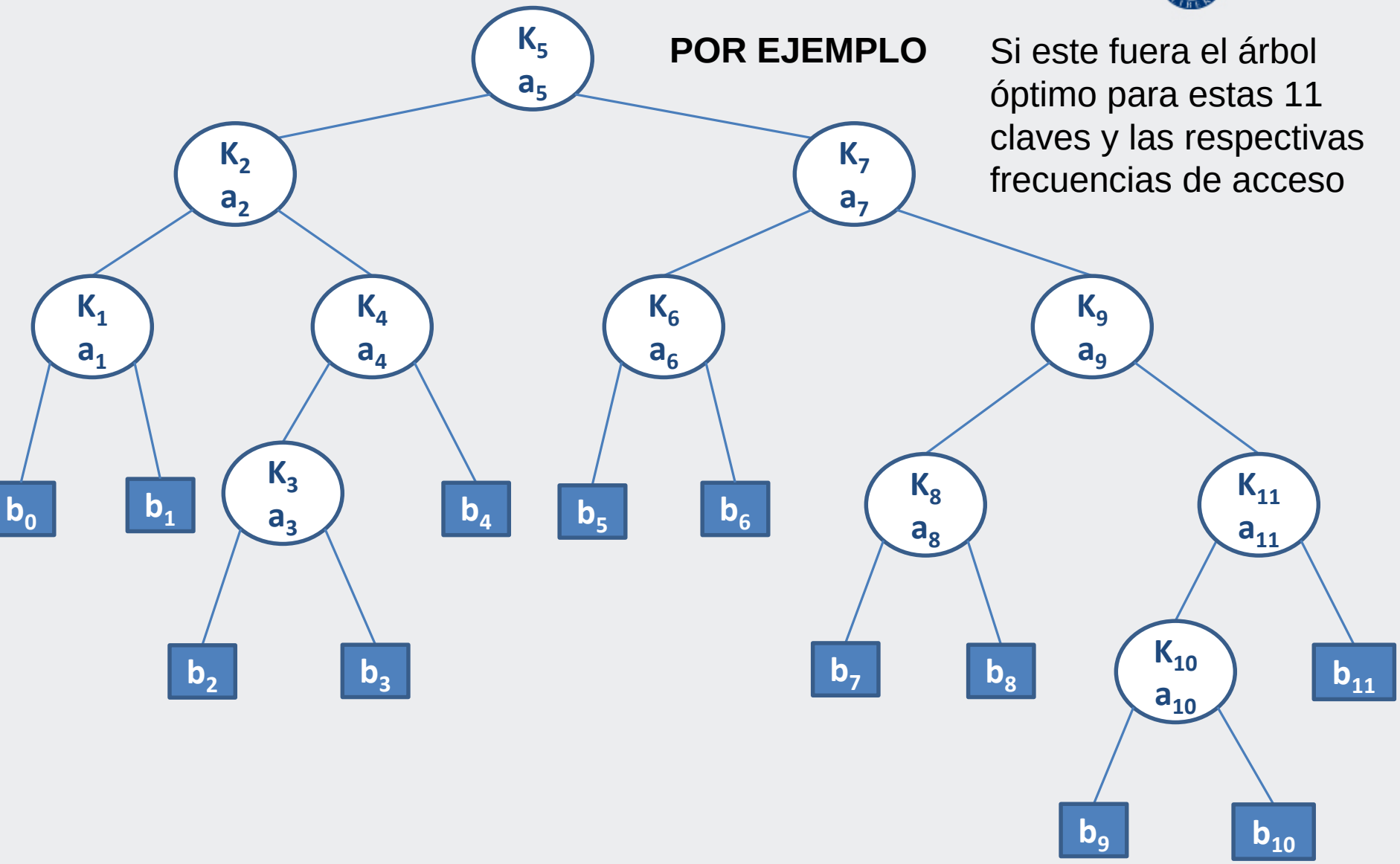
- El número de posibles configuraciones en un árbol de “N” nodos crece exponencialmente.
- Por lo tanto, un algoritmo que trate de armar todos los árboles binarios de búsqueda posibles, calcularle a cada uno su costo y quedarse con el de menor costo, tiene un orden de tiempo de ejecución exponencial, lo que lo hace impracticable.
- Debe entonces idearse otra solución. La solución encontrada se basa en un principio obvio:

Si un árbol es óptimo, sus dos subárboles también lo son.

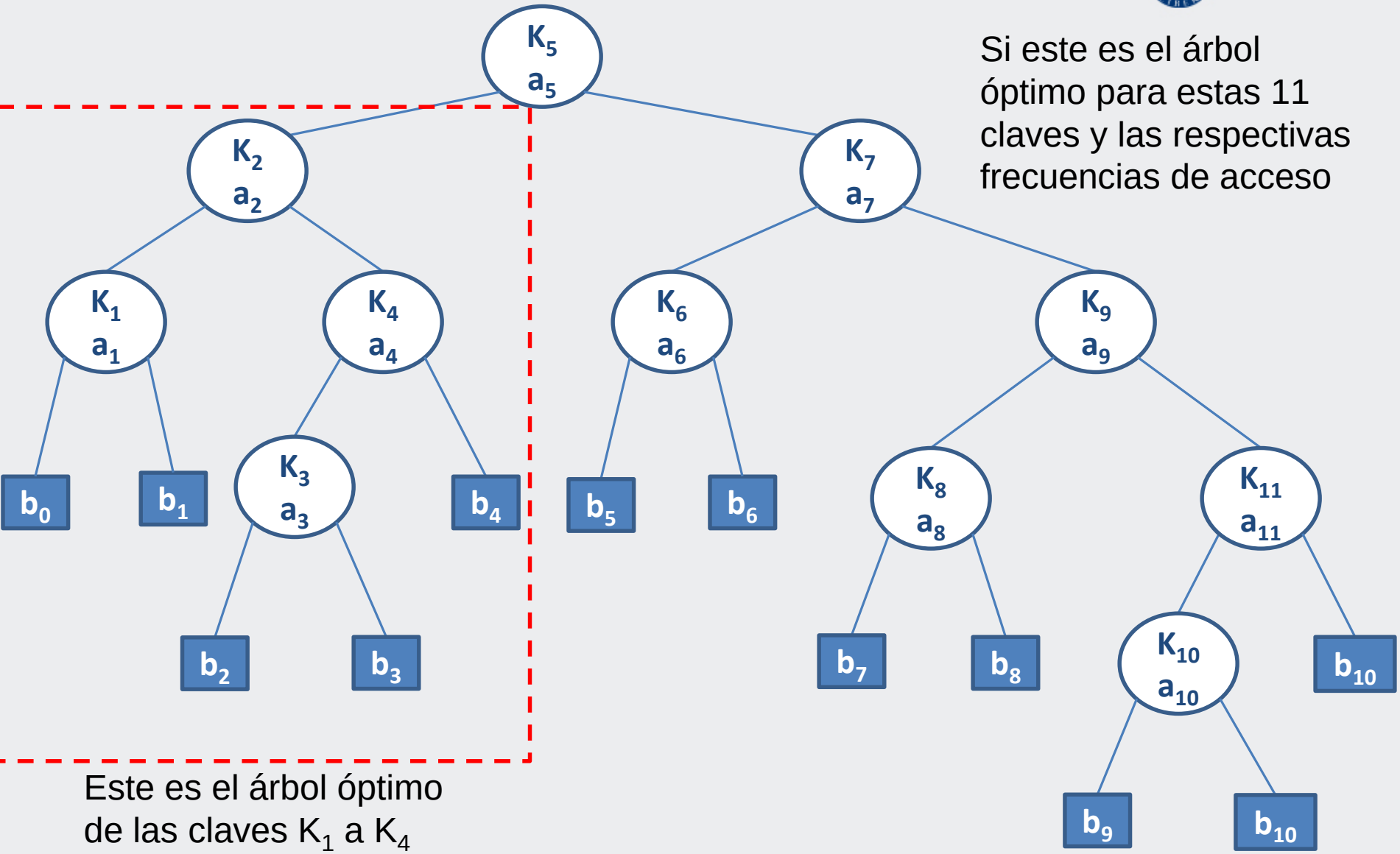
Árboles de búsqueda óptimos

POR EJEMPLO

Si este fuera el árbol óptimo para estas 11 claves y las respectivas frecuencias de acceso



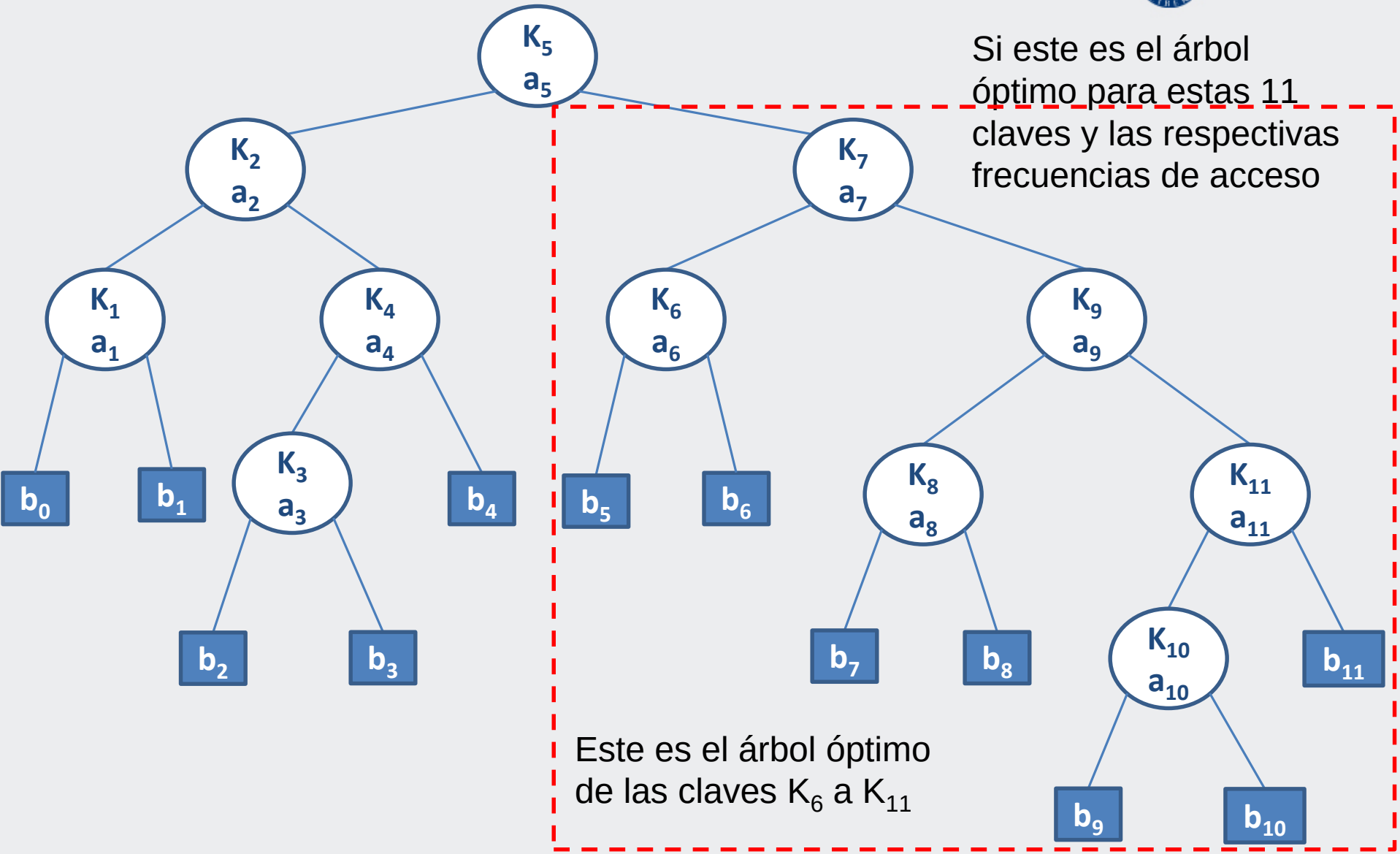
Árboles de búsqueda óptimos



Si este es el árbol óptimo para estas 11 claves y las respectivas frecuencias de acceso

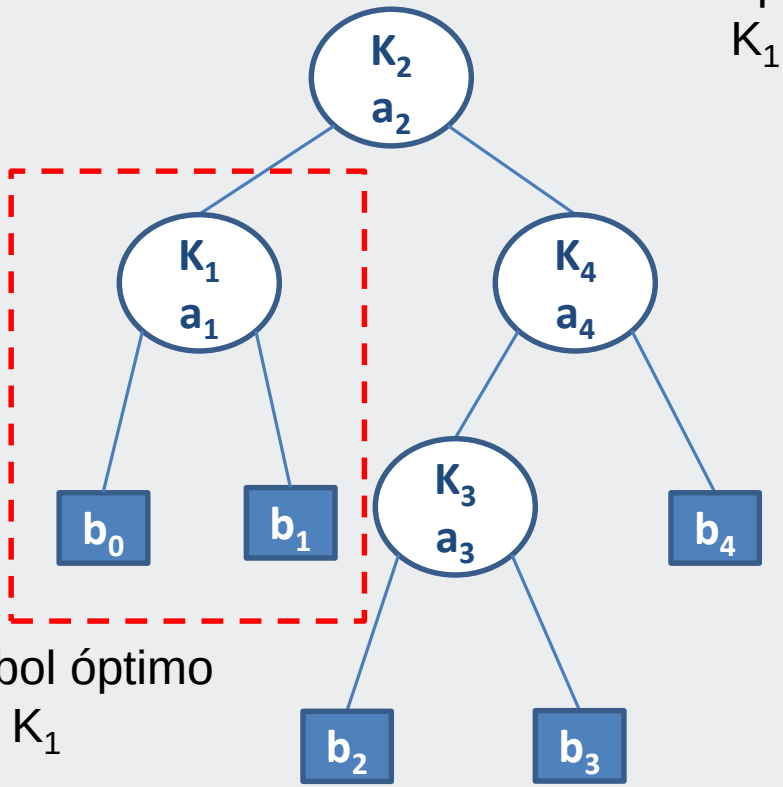
Este es el árbol óptimo de las claves K_1 a K_4

Árboles de búsqueda óptimos



Árboles de búsqueda óptimos

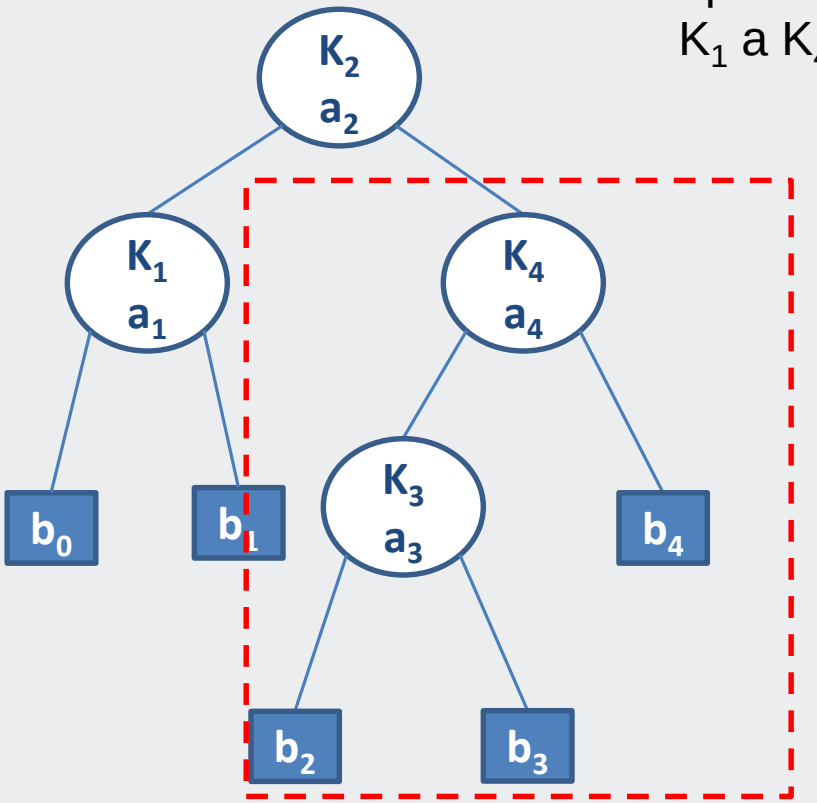
Si este es el árbol óptimo para las claves K_1 a K_4



Este es el árbol óptimo para la clave K_1

Árboles de búsqueda óptimos

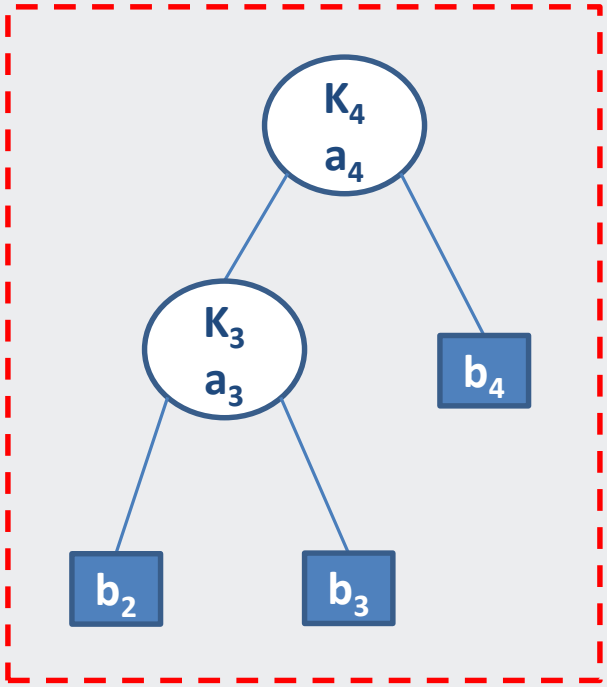
Si este es el árbol óptimo para las claves K_1 a K_4



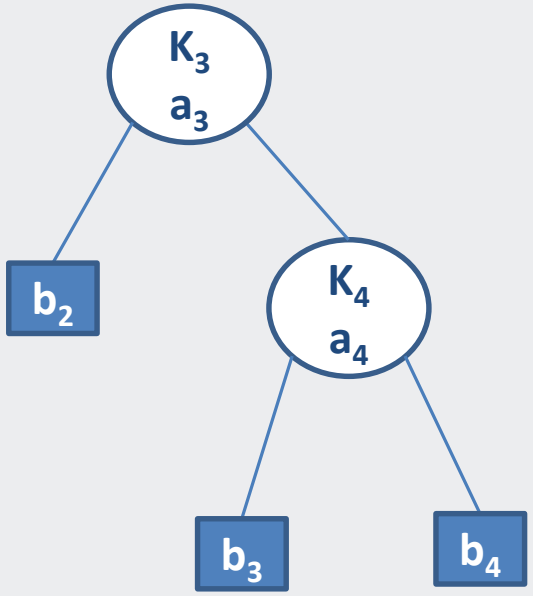
Este es el árbol óptimo para las claves K_3 y K_4

Árboles de búsqueda óptimos

De los dos árboles posibles
para las claves K_3 y K_4



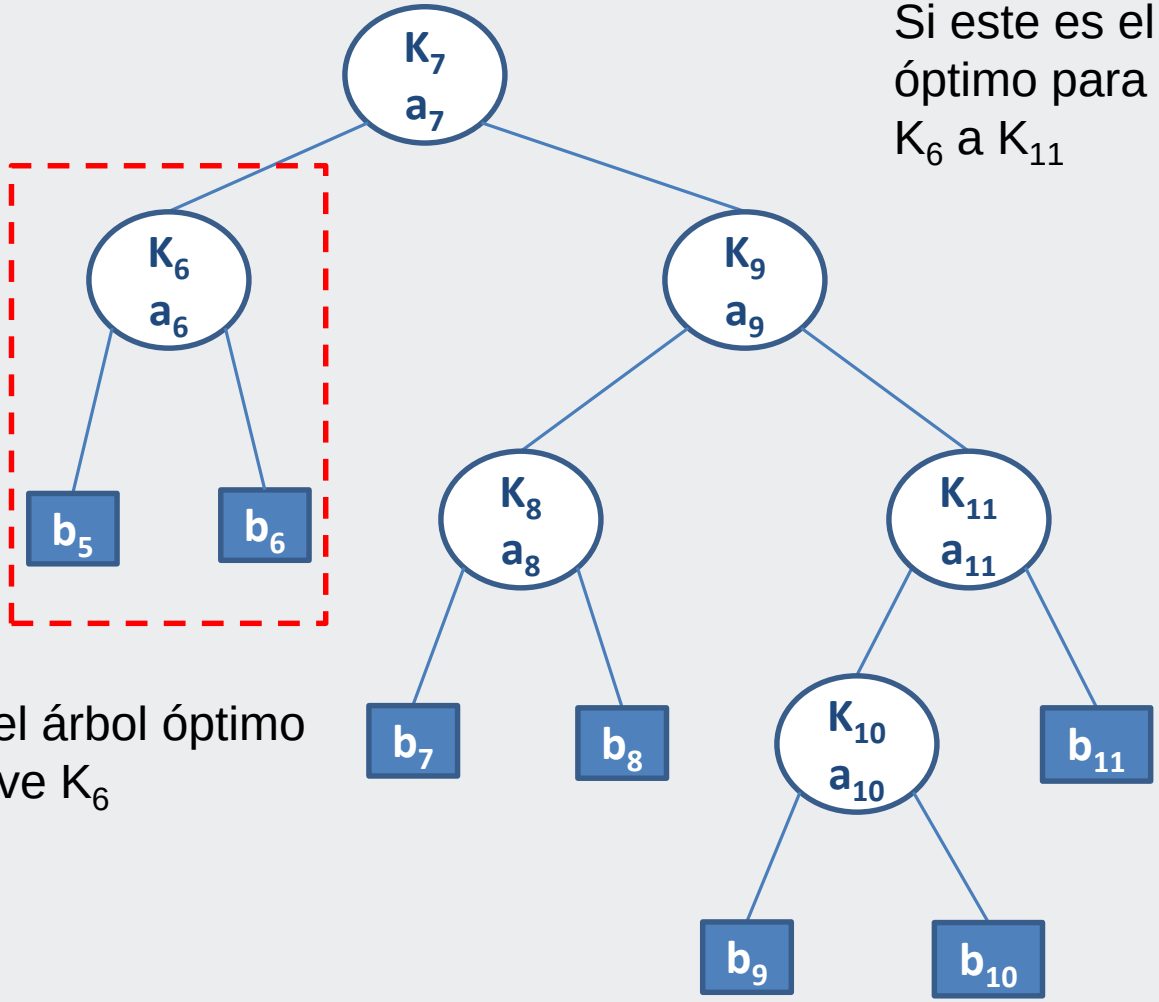
Este es el óptimo



Y no este

Árboles de búsqueda óptimos

Universidad Católica del Uruguay



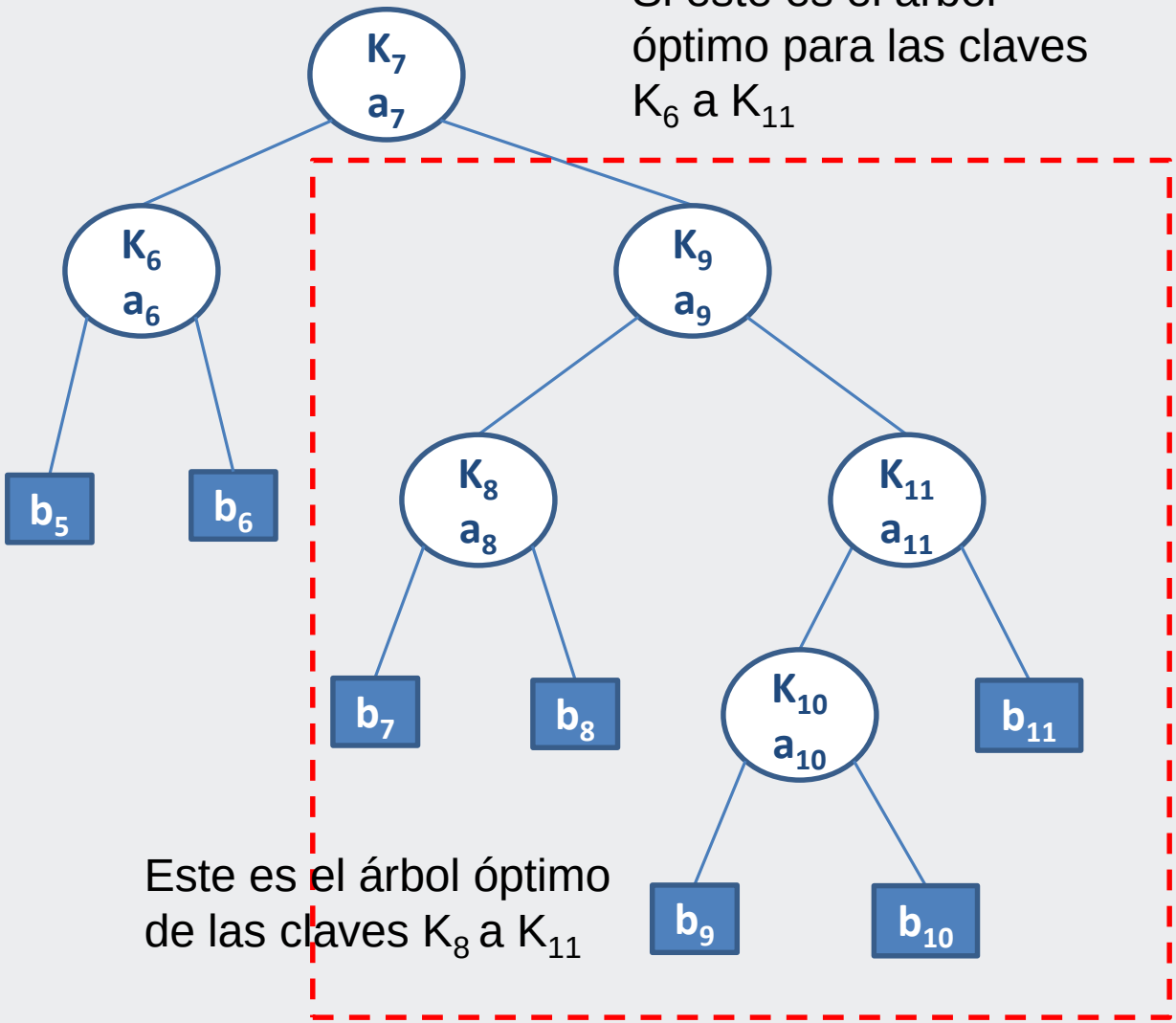
Si este es el árbol óptimo para las claves K_6 a K_{11}

Este es el árbol óptimo de la clave K_6

Árboles de búsqueda óptimos

Universidad Católica del Uruguay

Si este es el árbol óptimo para las claves K_6 a K_{11}



Este es el árbol óptimo de las claves K_8 a K_{11}

Árboles de búsqueda óptimos

Universidad Católica del Uruguay

- La estrategia entonces es hallar el árbol óptimo de “N” nodos, calculando:
  - Primero todos los N subárboles óptimos de un nodo
  - A partir de lo calculado en el paso anterior, hallar todos los N-1 subárboles óptimos de dos nodos
  - A partir de lo calculado en todos los pasos anteriores, hallar todos los N-2 subárboles óptimos de tres nodos
  - Así sucesivamente…
  - Como penúltimo paso hallar los dos subárboles óptimos de N-1 nodos
  - Y finalmente el árbol óptimo de N nodos

Árboles de búsqueda óptimos

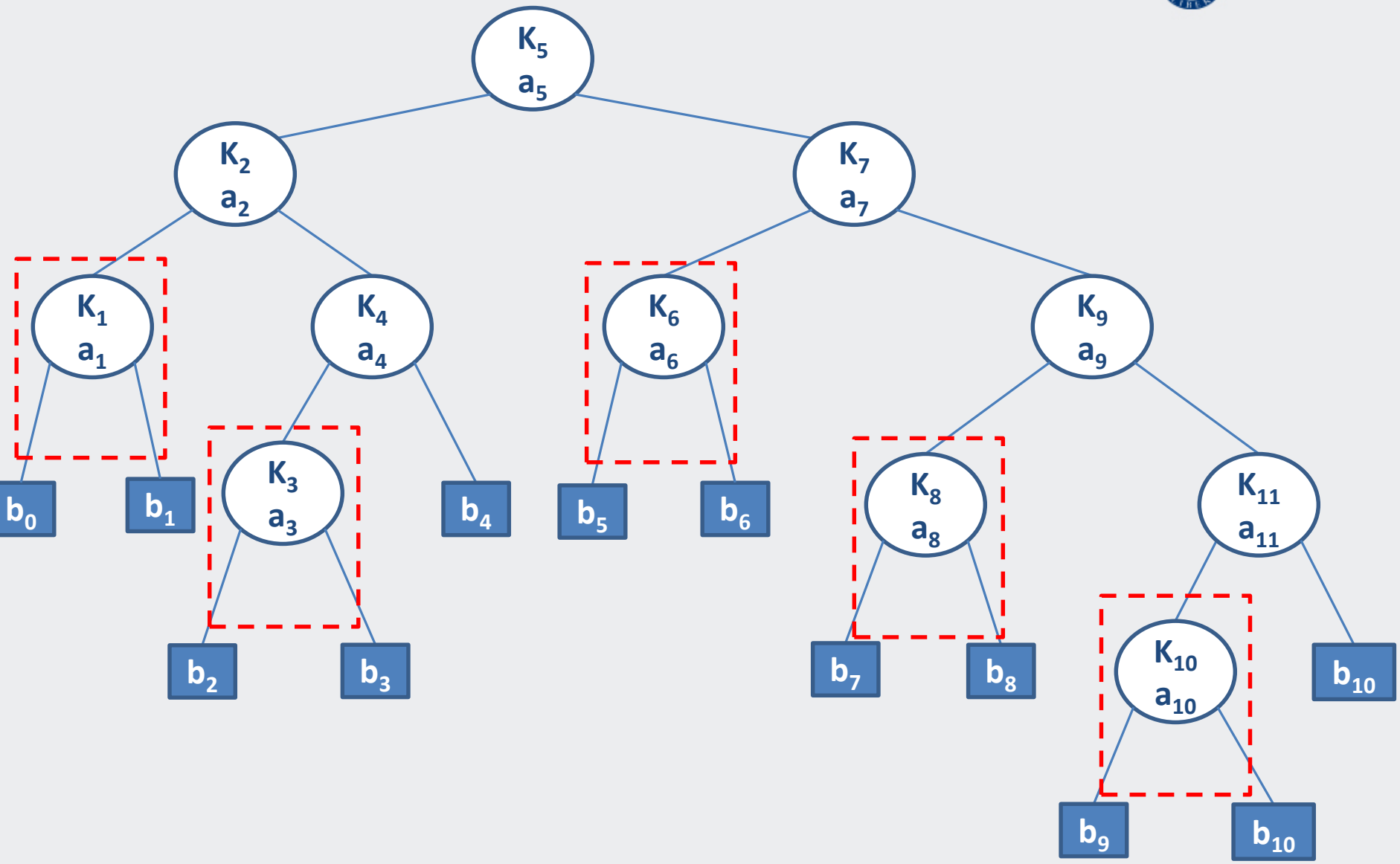
EN EL EJEMPLO

Estos son los 11 subárboles óptimos de 1 nodo

En el árbol óptimo final, aparecen cinco subárboles óptimos, los de claves 1, 3, 6, 8 y 10

NOTA: En estas placas omitimos dibujar los “nodos virtuales” b (no son nodos del árbol) para no sobrecargar las figuras

Árboles de búsqueda óptimos



Árboles de búsqueda óptimos

EN EL EJEMPLO

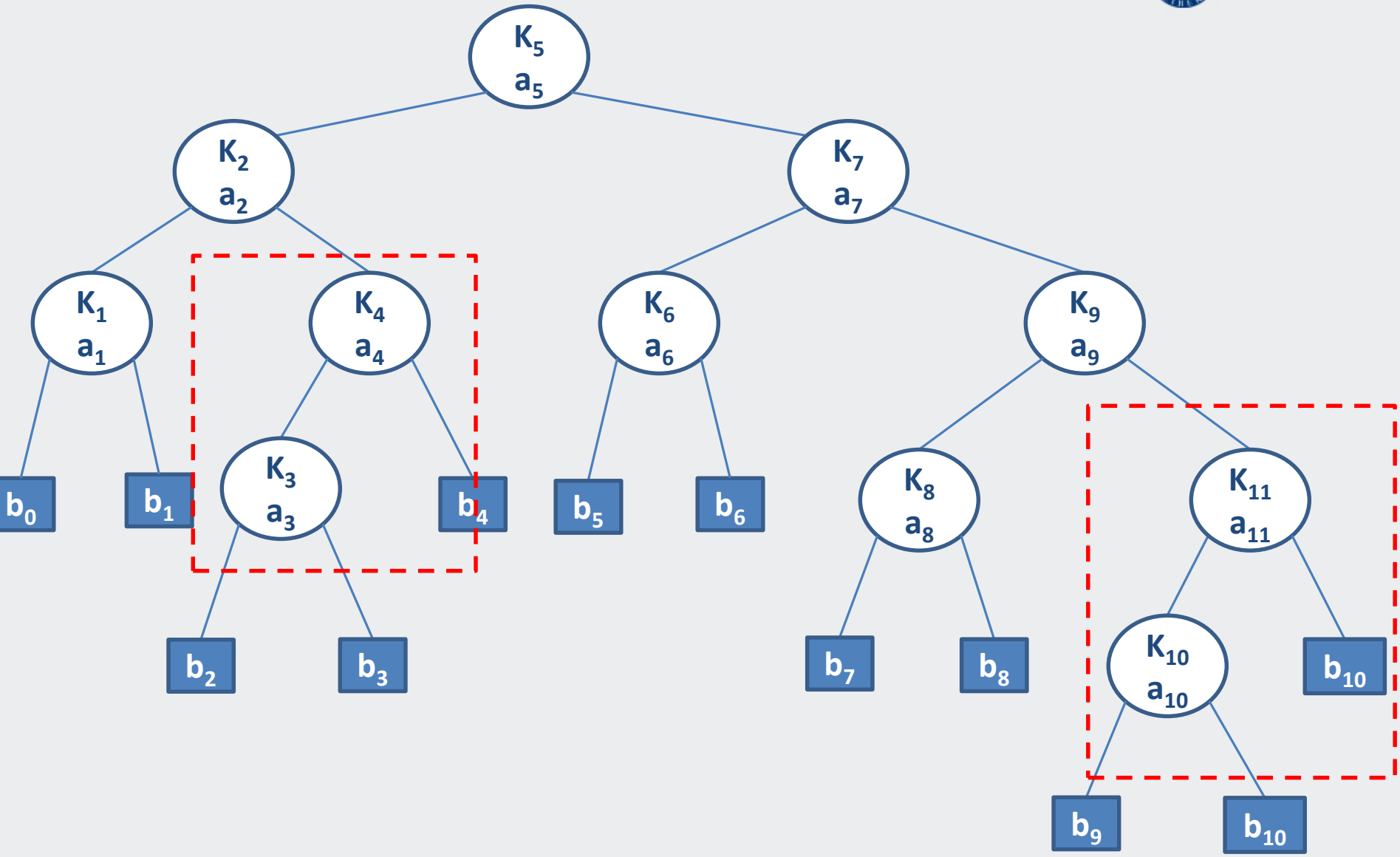
Estos son los 11 subárboles óptimos de 1 nodo

En el árbol óptimo final, aparecen cinco subárboles óptimos, los de claves 1, 3, 6, 8 y 10

Sobre esta base, calculamos ahora los 10 subárboles óptimos de 2 nodos

Es decir, de los dos posibles en cada caso, nos quedamos con el óptimo
En el árbol óptimo de ejemplo, al final aparecen dos de los subárboles de dos nodos, el de las claves 3 y 4, y el de las claves 10 y 11

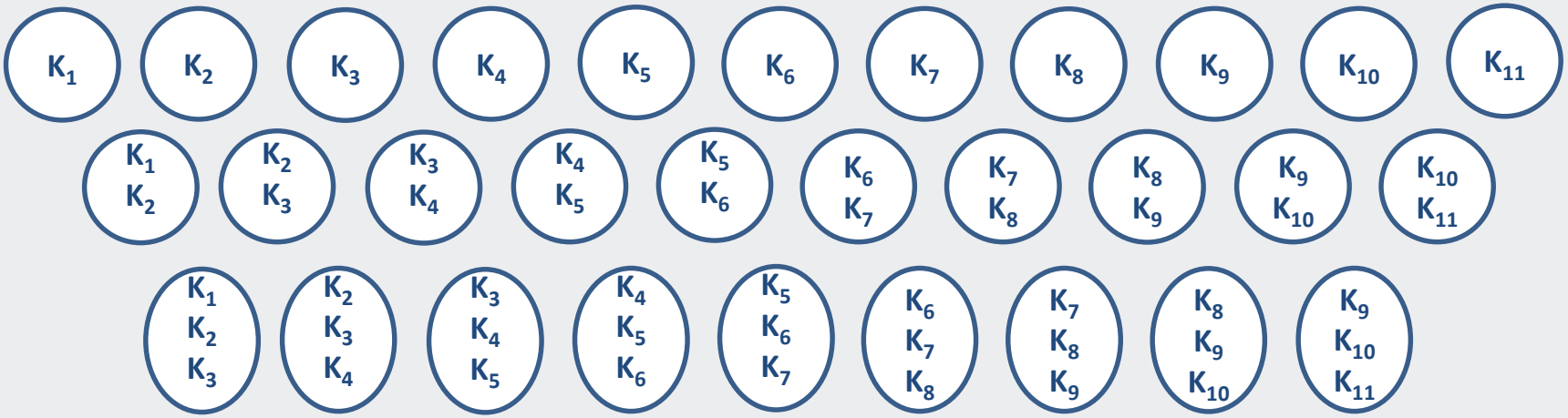
Árboles de búsqueda óptimos



Árboles de búsqueda óptimos

EN EL EJEMPLO

Sobre la base de los dos pasos anteriores base, calculamos ahora los 9 subárboles óptimos de tres nodos



Como ya están resueltos los pasos anteriores, es muy fácil obtener cada uno de estos subárboles

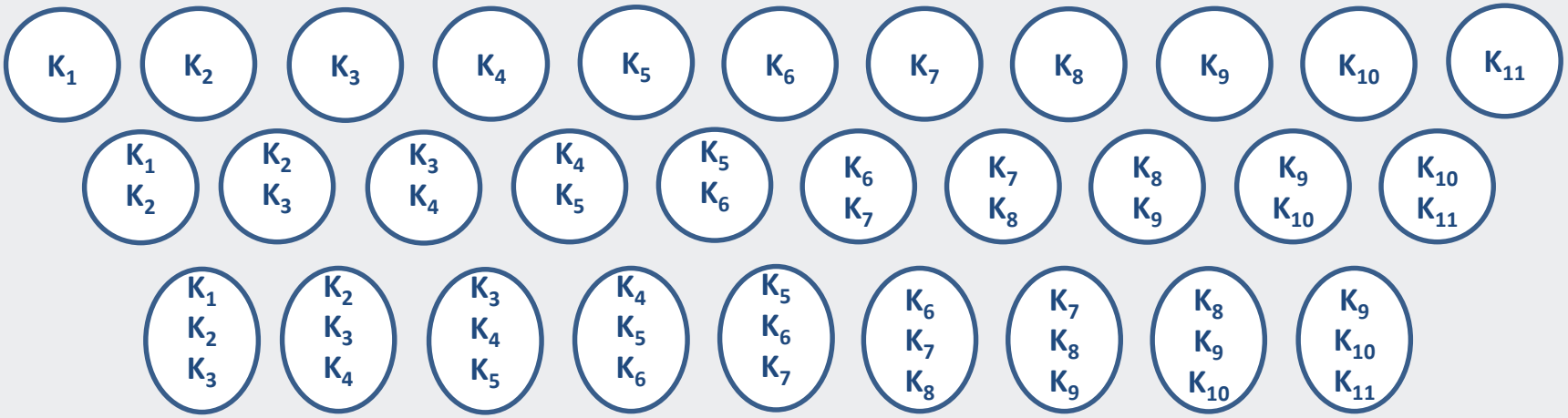
Por ejemplo, para el de las claves 5, 6 y 7

En el árbol óptimo de ejemplo no aparecen subárboles de tres nodos

Árboles de búsqueda óptimos

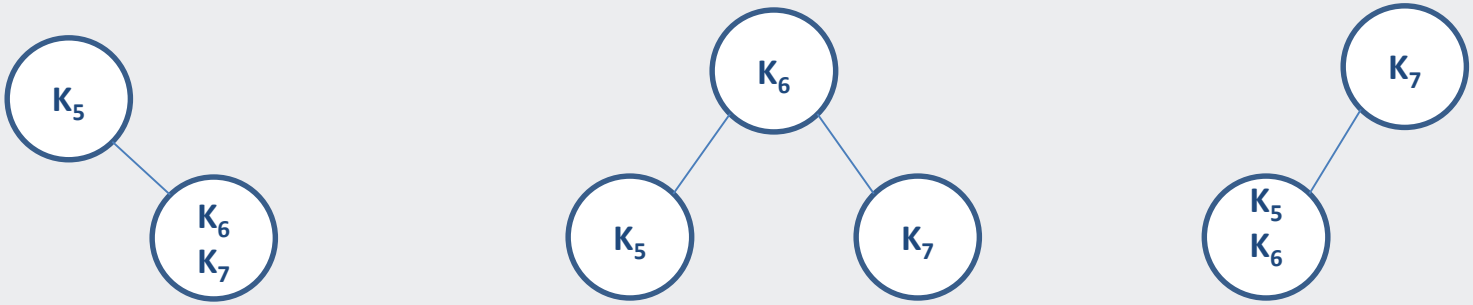
EN EL EJEMPLO

Sobre la base de los dos pasos anteriores base, calculamos ahora los 9 subárboles óptimos de tres nodos



Como ya están resueltos los pasos anteriores, es muy fácil obtener cada uno de estos subárboles

Por ejemplo, para el de las claves 5, 6 y 7, son estos los tres posibles

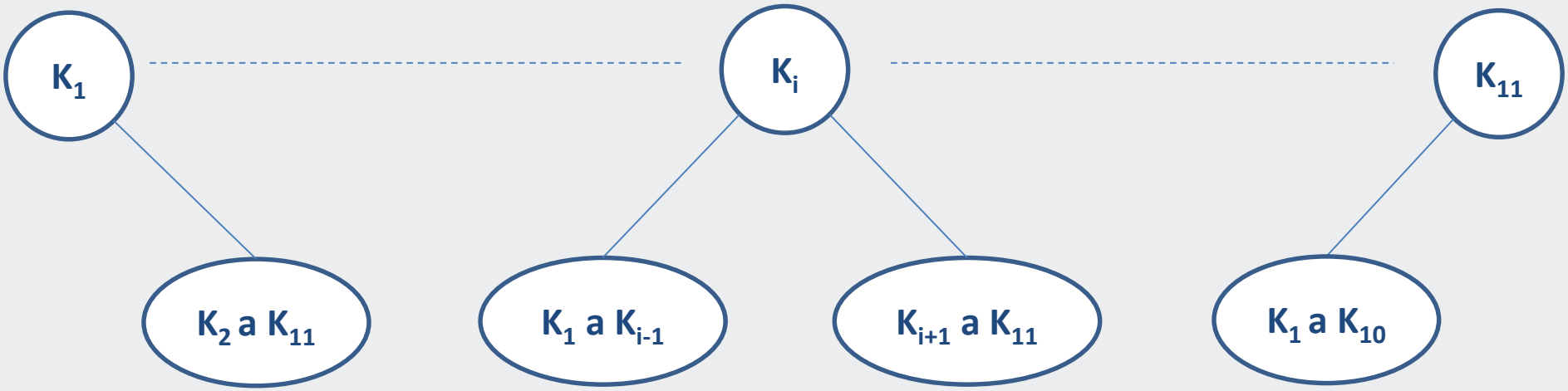


Se elige el de menor costo de los tres

Árboles de búsqueda óptimos

EN EL EJEMPLO

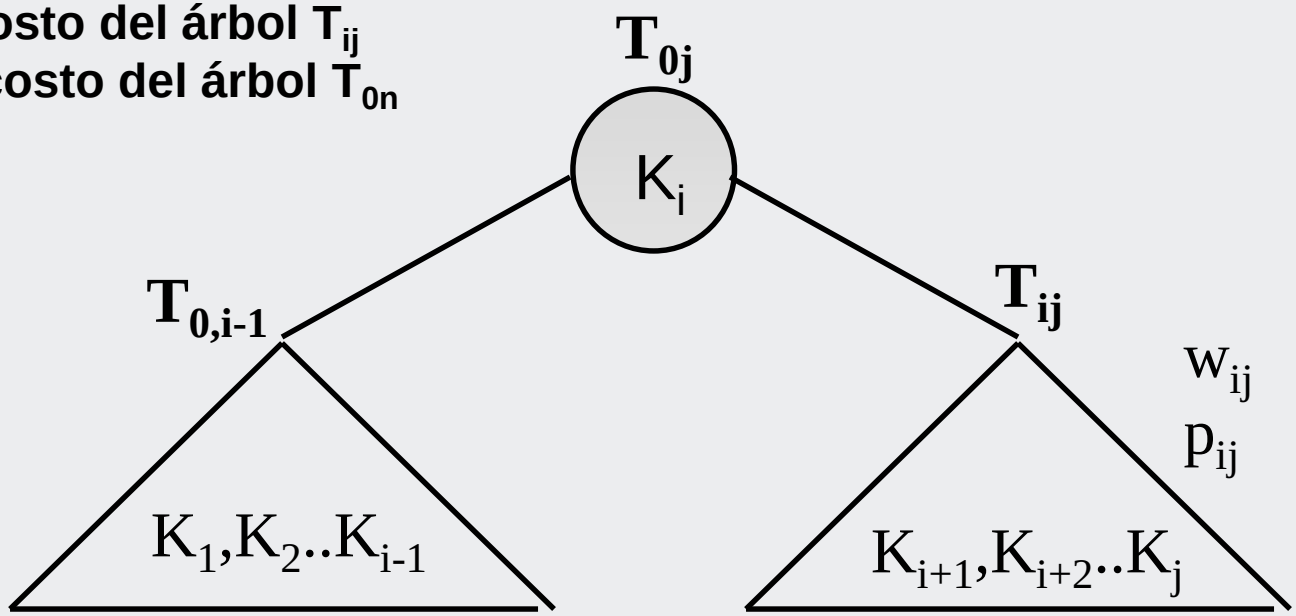
Y así sucesivamente hasta llegar al árbol de 11 nodos, hay que elegir entre 11 posibilidades:



Se elige el de menor costo de los once, y ya está calculado el árbol óptimo

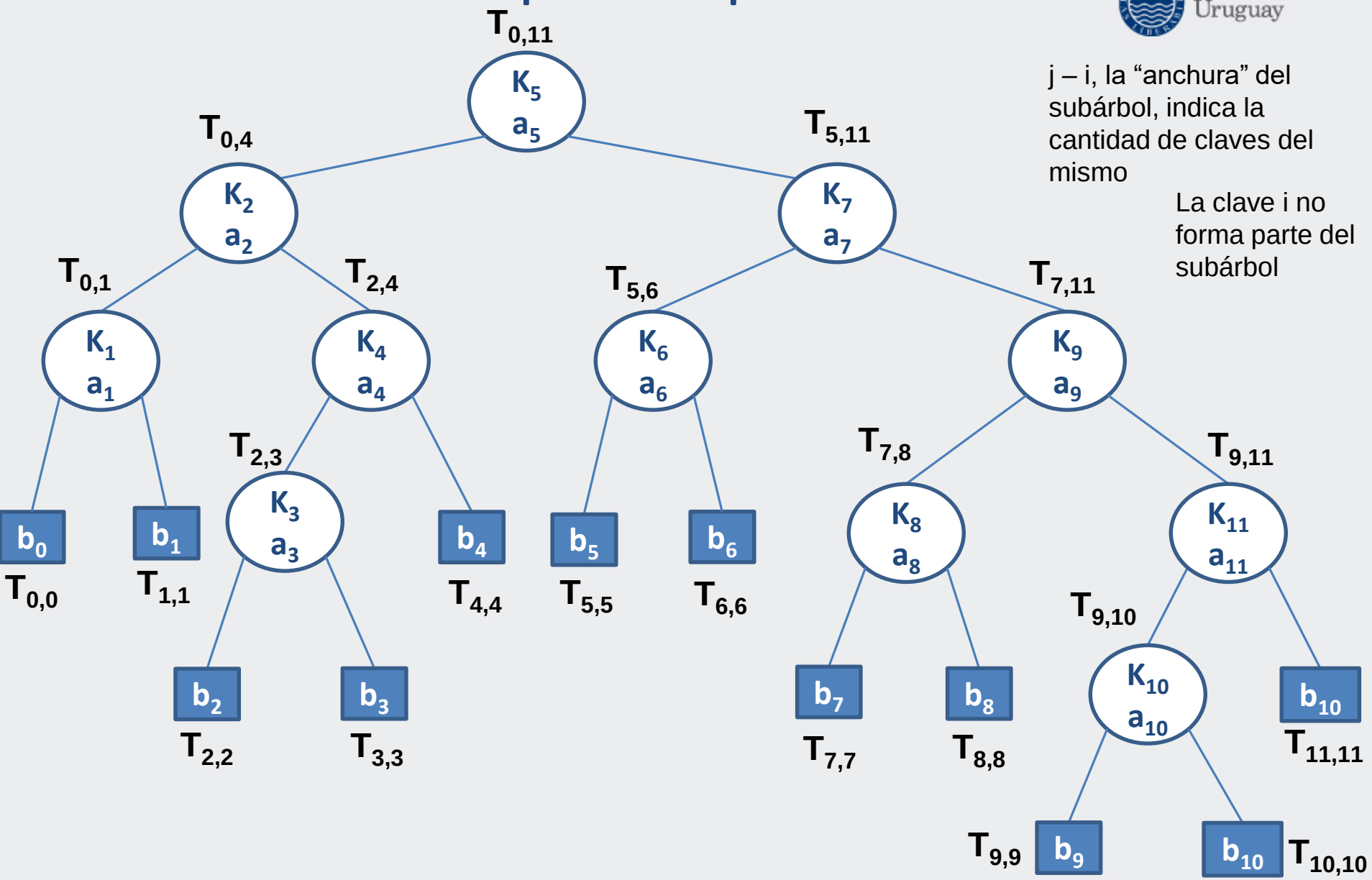
Árboles de búsqueda óptimos

- T_{ij} es el subárbol óptimo que contiene las claves K_{i+1} hasta K_j
- T_{0n} es el árbol óptimo que contiene las claves de la K_1 a la K_n
- W_{ij} es el peso del T_{ij}
- W_{0n} es el peso del árbol T_{0n}
- P_{ij} es el costo del árbol T_{ij}
- P_{0n} es el costo del árbol T_{0n}



$$w_{ij} = \sum_{k=i+1}^{j} a_k + \sum_{l=i}^{j} b_l \qquad p_{ij} = \sum_{k=i+1}^{j} a_k * h_k + \sum_{l=i}^{j} b_l * h_l'$$

Árboles de búsqueda óptimos



j – i, la “anchura” del subárbol, indica la cantidad de claves del mismo

La clave i no forma parte del subárbol

Árboles de búsqueda óptimos

Universidad Católica del Uruguay

– Definiendo a W como peso del árbol (número total de búsquedas)

$$W = \sum_{i=1}^{N} a_i + \sum_{j=0}^{N} b_j$$

– Se puede demostrar que

$$\mathbf{P = P_L + W + P_R}$$

– P longitud de trayectoria ponderada total del árbol.
– P_L longitud de trayectoria ponderada del subárbol izquierdo.
– P_R longitud de trayectoria ponderada del subárbol derecho.
– W peso del árbol.

Árboles de búsqueda óptimos

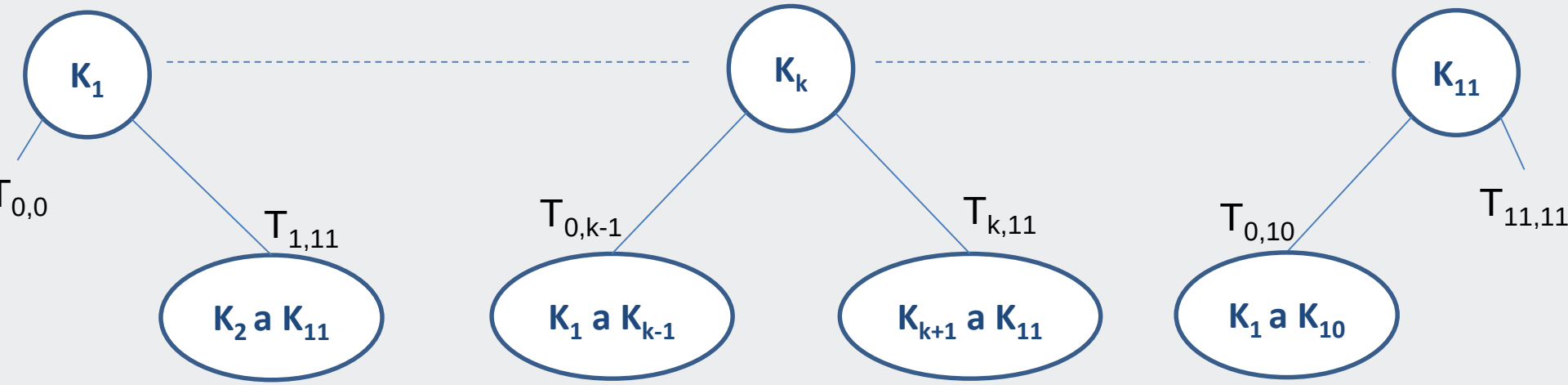
- Evidentemente, $P = p_{0,N}$ y $W = w_{0,N}$
- Estas cantidades se definen mediante las siguientes relaciones de recurrencia:

  - (1) $w_{ii} = b_i$ $\quad 0 <= i <= N$
  - (2) $w_{ij} = w_{ij-1} + a_j + b_j$ $\quad 0 <= i < j <= N$
  - (3) $p_{ii} = w_{ii}$ $\quad 0 <= i <= N$
  - (4) $p_{ij} = w_{ij} + \min_{[i < k <= j]}(p_{i,k-1} + p_{kj})$ $\quad 0 <= i < j <= N$

    - el valor de k que satisfaga ese mínimo, indica que la clave K_k es la raíz del subárbol T_{ij}

Árboles de búsqueda óptimos

EN EL EJEMPLO, para hallar $P_{0,11}$, el costo del árbol $T_{0,11}$



$$P_{0,11} = w_{0,11} + \min_{[0 < k <= 11]}(p_{0,k-1} + p_{k,11})$$

El valor de k (cualquiera entre 1 y 11) que satisfaga ese mínimo, indica que la clave K_k es la raíz del árbol $T_{0,11}$

Árboles de búsqueda óptimos

- ¿Cómo se guardan los valores de todos los subárboles óptimos de 1 nodo, 2 nodos, etcétera, hasta **n** nodos?
- Con matrices de tamaño n+1 por n+1
- En una matriz **W**, la posición W[i,j] contendrá el **peso** del subárbol $T_{i,j}$
- En una matriz **P**, la posición P[i,j] contendrá el **costo** del subárbol $T_{i,j}$
- En una matriz **R**, la posición R[i,j] contendrá la **raíz** del subárbol Ti,j

Árboles de búsqueda óptimos – estructuras de datos usadas

Universidad Católica del Uruguay

- Datos de entrada:
  - Vector **K** de tamaño n+1 para las claves (la posición 0 no se usa)
  - Vector **A** de tamaño n+1 para las frecuencias **exitosas** (la posición 0 no se usa)
  - Vector **B** de tamaño n+1 para las frecuencias **no exitosas**
- Para guardar los resultados intermedios y la salida del algoritmo:
  - Matriz **W** de tamaño n+1 por n+1 para guardar los **pesos**
  - Matriz **P** de tamaño n+1 por n+1 para guardar los **costos**
  - Matriz **R** de tamaño n+1 por n+1 para guardar las **raíces**
    - Nota: en las tres matrices solamente se usa la parte triangular superior.

Árboles de búsqueda óptimos – fases del algoritmo

Universidad Católica del Uruguay

- **Fase 1**: A partir de los vectores de entrada **A** y **B**, cálculo de los valores de las matrices **W**, **P** y **R**
  - Inicialización de las diagonales de las matrices W y P
    - P[i,i] = W[i,i] = B[i] para i = 0, 1, 2, …. , N
  - Completado de la matriz **W**
    - W[i,j] = W[i, j-1] + A[j] + B[j] $0 <= i < j <= N$
  - Completado simultáneo de las matrices **P** y **R**
    - P[i,j] = W[i,j] + mínimo $_{i<k=<j}$ (P[i,k-1] + P[k,j]) $0 <= i < j <= N$
    - R[i,j] = k
- **Fase 2**: A partir del vector de entrada **K** y de la matriz calculada **R**, armado del árbol binario óptimo
  - Se insertan en un árbol binario vacío las claves del vector **K**, en el orden establecido por la matriz **R**.

Árboles de búsqueda óptimos – fase 1 del algoritmo

Inicialización de matrices h = j – i = 0, árboles de ancho cero

W

	0	1	2	3	4	5	6	7	8
0	B[0]								
1		B[1]							
2			B[2]						
3				B[3]					
4					B[4]				
5						B[5]			
6							B[6]		
7								B[7]	
8									B[8]

P

	0	1	2	3	4	5	6	7	8
0	B[0]								
1		B[1]							
2			B[2]						
3				B[3]					
4					B[4]				
5						B[5]			
6							B[6]		
7								B[7]	
8									B[8]

P[i,i] = W[i,i] = B[i] para i = 0, 1, 2, …. , N

Árboles de búsqueda óptimos – fase 1 del algoritmo

Universidad Católica del Uruguay

Completado de la matriz W

	0	1	2	3	4	5	6	7	8
0	**B[0]**								
1		**B[1]**							
2			**B[2]**						
3				**B[3]**					
4					**B[4]**				
5						**B[5]**			
6							**B[6]**		
7								**B[7]**	
8									**B[8]**

$W[i,j] = W[i, j-1] + A[j] + B[j] \qquad 0 <= i < j <= N$

Árboles de búsqueda óptimos – fases del algoritmo

Universidad Católica del Uruguay

- **TA2 EJERCICIO 1**
- **Paso 1**

- **COMPLETAR**
- **h = 0 y la matriz W**

Árboles de búsqueda óptimos – fase 1 del algoritmo

Universidad Católica del Uruguay

Completado simultáneo de las matrices P y R

Cuando **h = 1,** árboles de un sólo nodo

```
Para i:= 0 hasta N-1 do
   j := i+1;
   P[i,j] := W[i,j] + P[i,i] + P[j,j];
   R[i,j] := j;
Fin para;
```

Árboles de búsqueda óptimos – fases del algoritmo

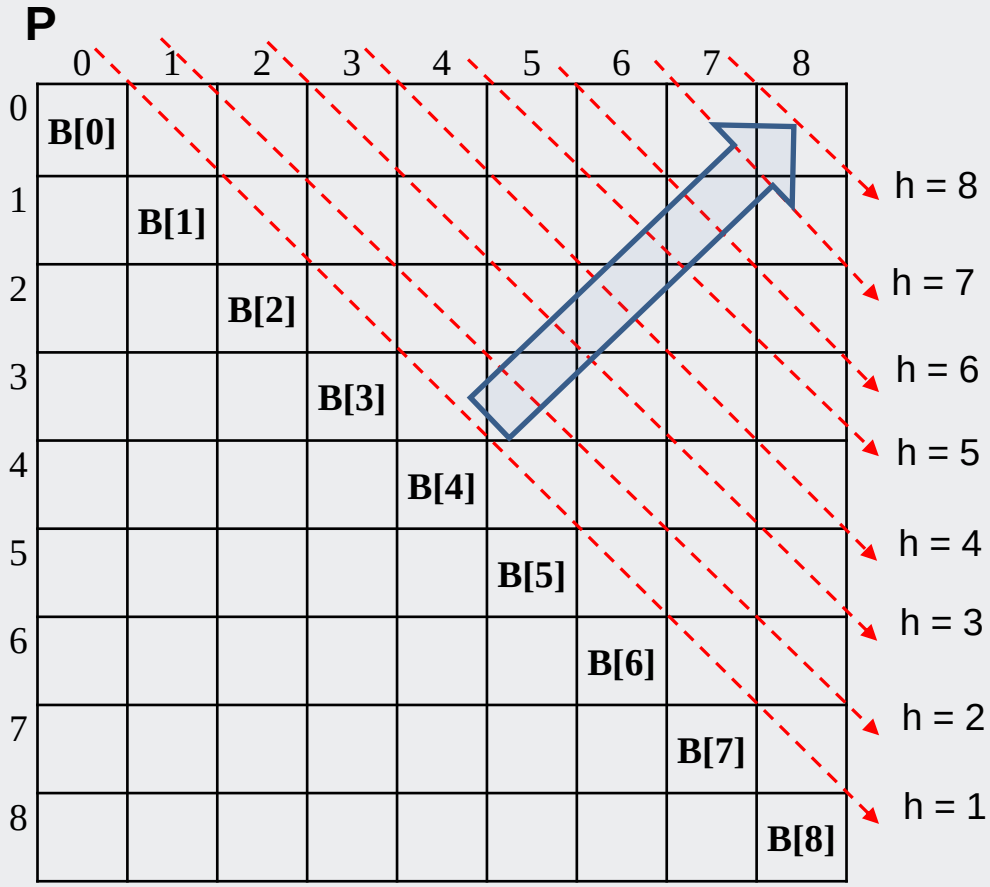
- **TA2 EJERCICIO 1**
- **PASO 2**

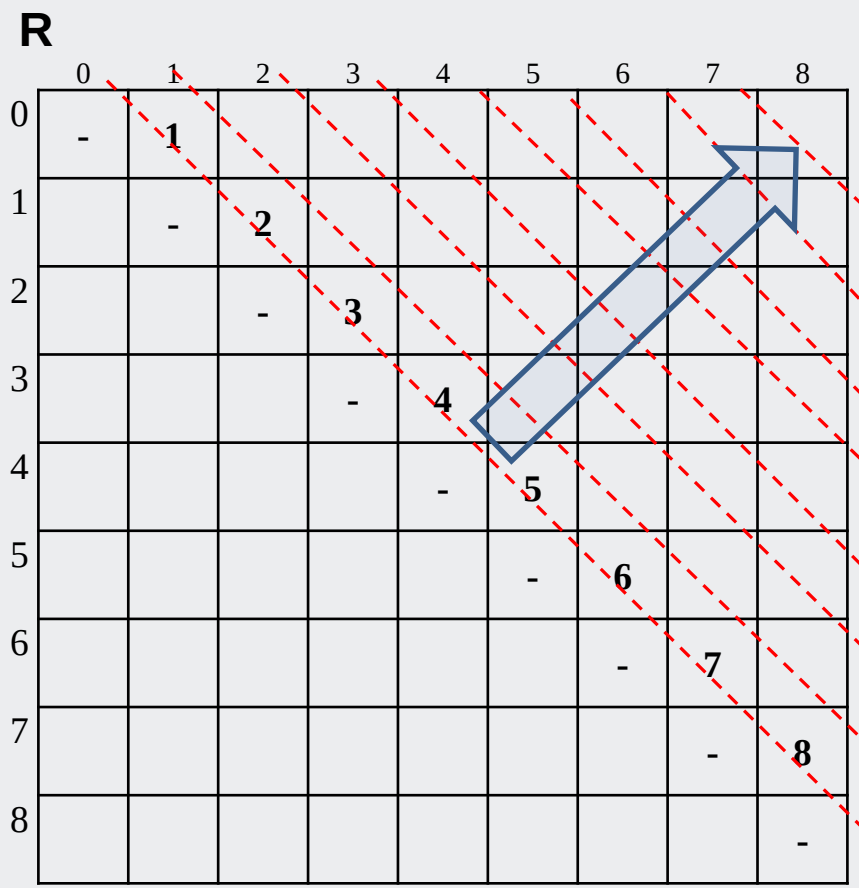
- **COMPLETAR**
- **h = 1**

Árboles de búsqueda óptimos – fase 1 del algoritmo

Universidad Católica del Uruguay

Completado simultaneo de las matrices P y R





Ancho del árbol T_{ij} h = j - i

Árboles de búsqueda óptimos – fase 1 del algoritmo

Universidad Católica del Uruguay

Completado simultáneo de las matrices P y R

Para los casos **h > 1** usamos un bucle entre 2 y N; el caso **h = N** cubre todo el árbol

```
Para h := 2 hasta N do
    Para i := 0 hasta n-h do
        j := i + h;
        encontrar k y min = MIN [i < k <= j]( P[i,k-1] + P[k,j] )
        P[i,j] := min + W[i,j];
        R[i,j] := k;
     fin para;
fin para;
```

Árboles de búsqueda óptimos – fases del algoritmo

- **TA2 EJERCICIO 2**
- **PASO 3**

- **COMPLETAR h desde 2 hasta N,**
- **Y para cada par de valores i, j**
  - **Desde k = i + 1 hasta j,**
    - **Ir quedándose con el menor valor de la suma y el valor de k que lo consigue.**

Árboles de búsqueda óptimos – fase 1 del algoritmo

Fin de la fase 1

Al finalizar esta fase

- En la posición P[0,N] se encuentra el costo del árbol $T_{0,N}$
- En la posición R[0,N] se encuentra la raíz del árbol $T_{0,N}$ (el índice en el vector de claves) , sea el valor k
- La raíz de subárbol izquierdo está en R[0,k-1]
- La raíz del subárbol derecho está en R[k,N]
- Y así sucesivamente para cada subárbol hasta armar todo el árbol

Árboles de búsqueda óptimos – fase 2 del algoritmo

A partir del vector de claves y de la matriz R, armar el árbol

```
armarArbol (TipoVector claves; TipoMatriz laMatrizR;
                        TipoIndiceMatriz i,j)
COM
    Si i < j entonces
        raíz ← laMatrizR[i,j]
        insertar(claves[raíz])
        armarArbol(claves, laMatrizR, i, raíz-1)
        armarArbol(claves, laMatrizR, raíz, j)
    Fin si
FIN
```

Árboles de búsqueda óptimos – fase 2 del algoritmo

Matriz R

	0	1	2	3	4
0		**1**	**1**	**2**	**3**
1			**2**	**3**	**3**
2				**3**	**4**
3					**4**
4