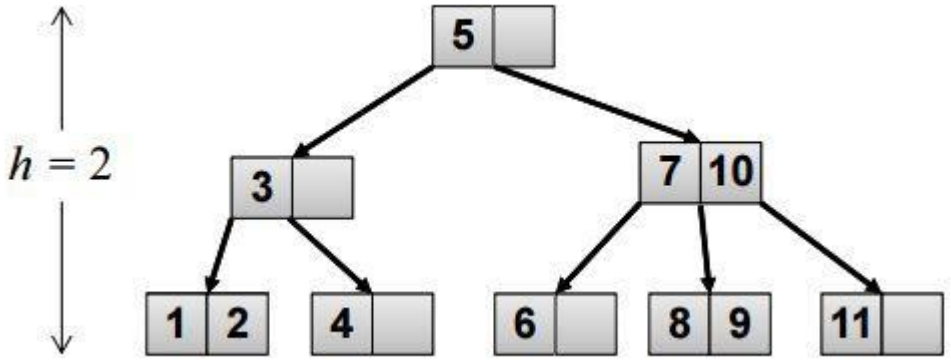
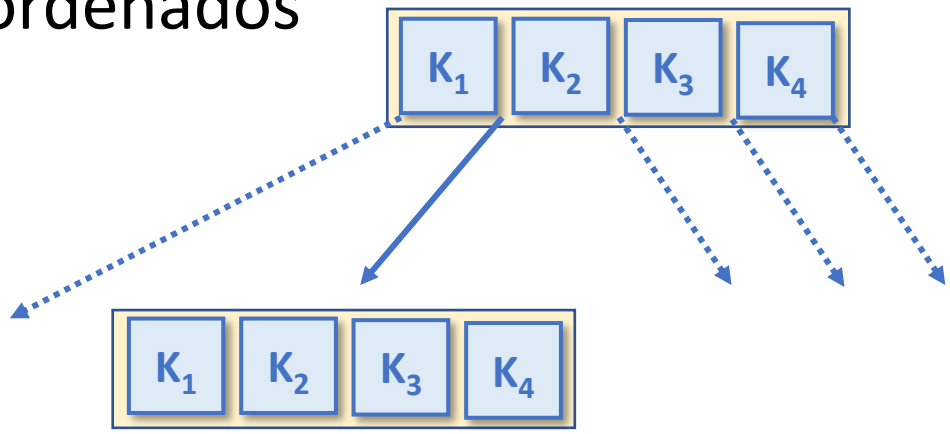
Árboles B (Bayer & McCreight)

- Objetivo
  - Modelar árboles sobre memoria secundaria (disco) capaces de almacenar cantidades masivas con acceso logarítmico
- Reducen la altura del árbol a costa de almacenar múltiples elementos por nivel
- Desarrollado por el alemán Rudolf Bayer y el suizo Edward M. McCreight en 1972

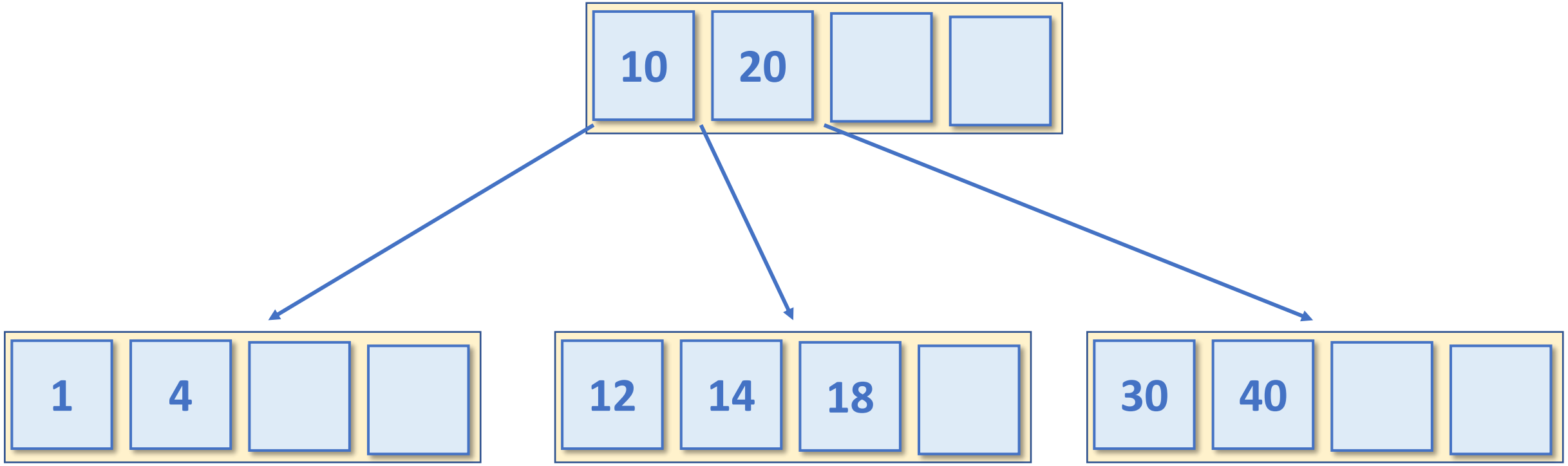


Árbol B de orden n (B-n). Definición

- Los nodos se llaman páginas
- Todas páginas las hojas se encuentran al mismo nivel
- Una página contiene m elementos o claves ordenados
- Las claves se almacenan de forma ordenada
  - La pagina raíz contiene 1 <= m <= 2n claves
  - La página no raíz contiene n <= m <=2n claves
- Toda página no hoja tiene m+1 página hijas
- Página en situación critica → m=n
  - Si el numero de claves de la página es igual al grado del árbol



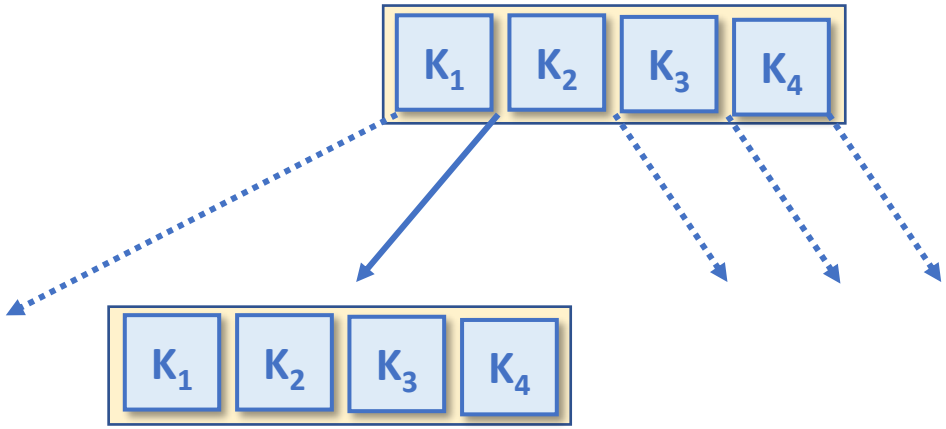
Árbol B. Ejemplo



Árbol B-2

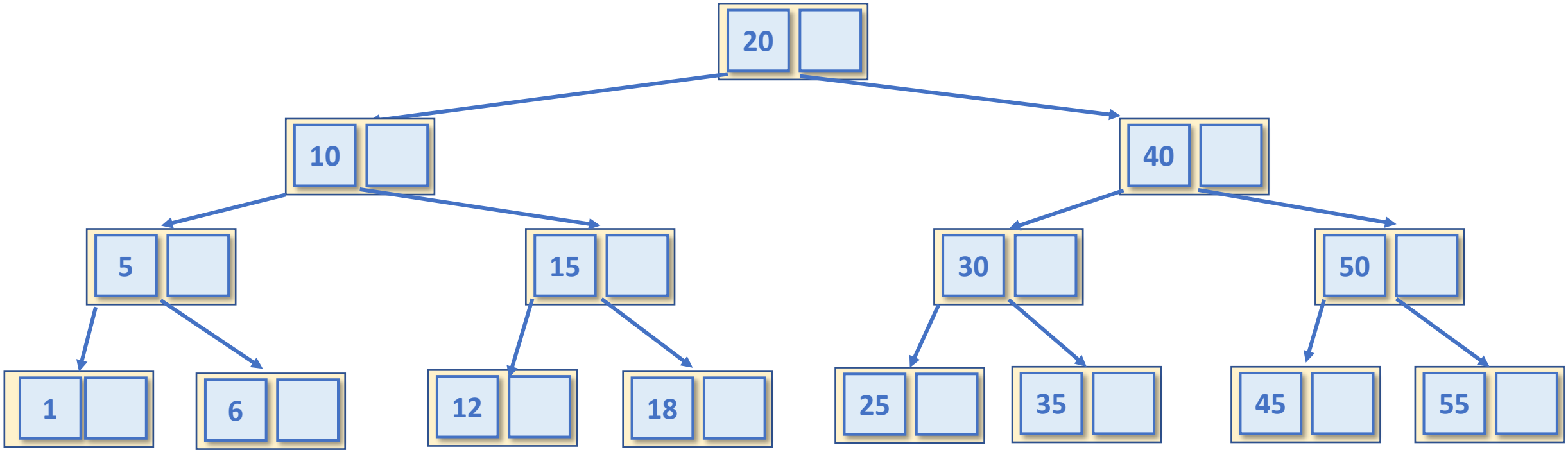
¿Cómo podría implementarse un nodo?

- Un nodo es una página o conjunto de claves
- Atributos que definen una página de un árbol B
  - Número **mínimo** de claves de página (es el orden del árbol)
  - Número **máximo** de claves de la página (el doble del mínimo)
  - **Lista de claves** el la página
  - **Lista de páginas hijas** de la página definida



Capacidad mínima de un árbol B

- Dado un árbol B de **orden n** y de **altura h**, si este almacena el número mínimo de claves por página, se corresponde con un árbol degenerado o estirado al máximo



Cálculo de la capacidad mínima (claves)

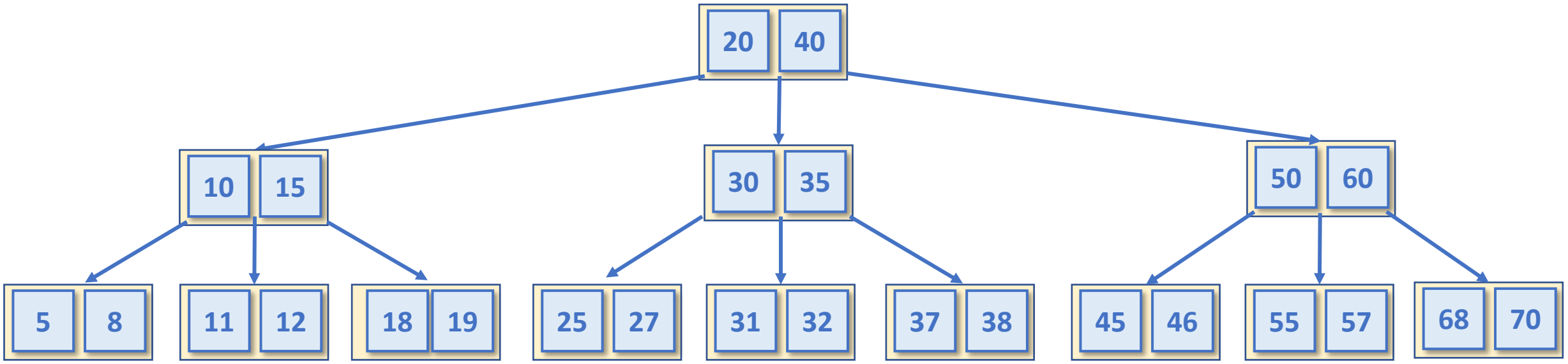
Nivel	Páginas por nivel	Valor mínimo de m	Total
1			
2			
3			
4			
…			
h			

Altura máxima de un árbol B

- La capacidad mínima por hoja implica un árbol de altura máxima
- Se obtiene de la formula anterior
  - $\mathrm{N} = 1 + 2\mathrm{n} \ * \ \sum_{\mathrm{i}=2}^{\mathrm{h}} (\mathrm{n}+1)^{\mathrm{i}-2}$
  - Donde N es el número de claves del árbol
- La altura máxima se aproxima a
  - $\mathrm{h}_{\mathrm{max}} \approx 1 + \log_{\mathrm{n}+1} {}^{(\mathrm{N}+1)}/_{2}$
  - Si la constante **n** es muy grande
    - la $\mathbf{h}_{\mathbf{max}}$ se aproxima a: $\boldsymbol{h_{max} \approx log_n}$ **N → O(log$_n$N)**
- Cuanto mayor se el orden (n) del árbol, menor será la altura del árbol

Capacidad máxima de un árbol B

- Dado un árbol B de **orden n** y de **altura h**, si este almacena el número máximo de claves por página, se corresponde con un árbol completo



Cálculo de la capacidad máxima (claves)

Nivel	Páginas por nivel	Valor máximo de m	Total
1			
2			
3			
4			
…			
h			

Altura mínima de un árbol B

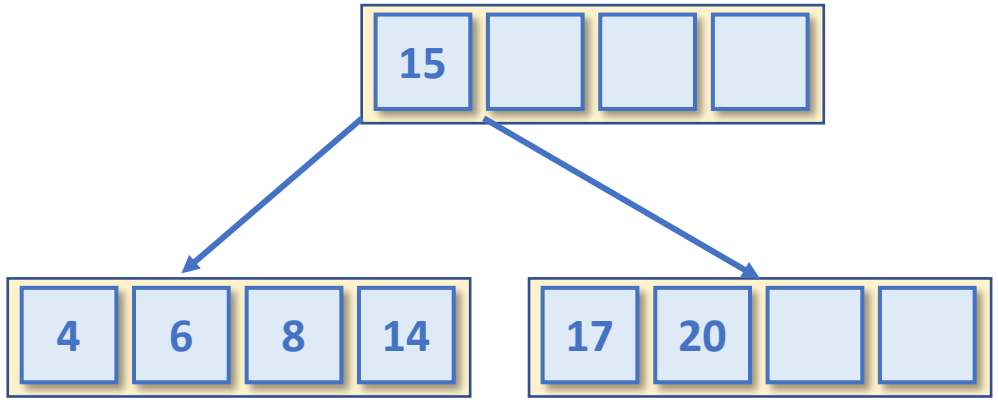
- La capacidad máxima por hoja implica un árbol de altura mínima
- Se obtiene de la formula anterior
  - $\mathrm{N} = 2\mathrm{n} * \sum_{\mathrm{i}=1}^{\mathrm{h}} (2\mathrm{n}+1)^{\mathrm{i}-1}$
  - Donde N es el número de claves del árbol
- La altura mínima se aproxima a
  - $\mathrm{h}_{\mathrm{min}} \approx \log_{\mathrm{n}+1} (\mathrm{N}+1)$
  - Si la constante **n** es muy grande
    - la $\mathbf{h}_{\mathbf{min}}$ se aproxima a: $\boldsymbol{h}_{\mathbf{min}} \approx \boldsymbol{log}_{2n}\ \mathbf{N} \rightarrow \mathbf{O}(\log_{2n}\mathbf{N})$
- Cuanto mayor se el orden (n) del árbol, menor será la altura del árbol

Árbol B. Buscar

- Buscar una clave → búsqueda secuencia o binaria
- Si la búsqueda falla esta se detendrá en una posición dentro de la página entre 0 y m. Se buscará en la página siguiente
- El proceso finaliza
  - o bien porque se encuentra la clave
  - o bien porque se encuentra un enlace null en cuyo caso la clave no existe
- Complejidad temporal
  - Caso mejor → la clave está en la raíz → O(m) = (1)
    - dado que 1<=m<=2n, m se puede considerar constante
  - Caso peor → se busca en un árbol degenerado y la clave no se encuentra
    - $O(m) * O(h) = O(\log_n N) * O(1) = O(\log_n N)$

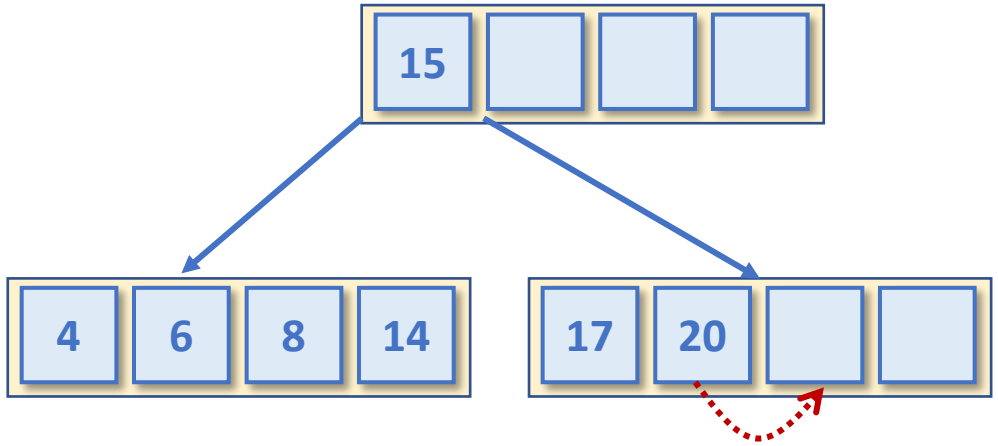
Árbol B. Insertar. Caso 1

- La página donde se quiere insertar tiene m <2n claves
- La inserción siempre se produce en la hojas
- Se abre hueco par el elemento a insertar, desplazando los elementos de clave mayor una posición a la derecha si es preciso
- Ejemplo: insertar el 19



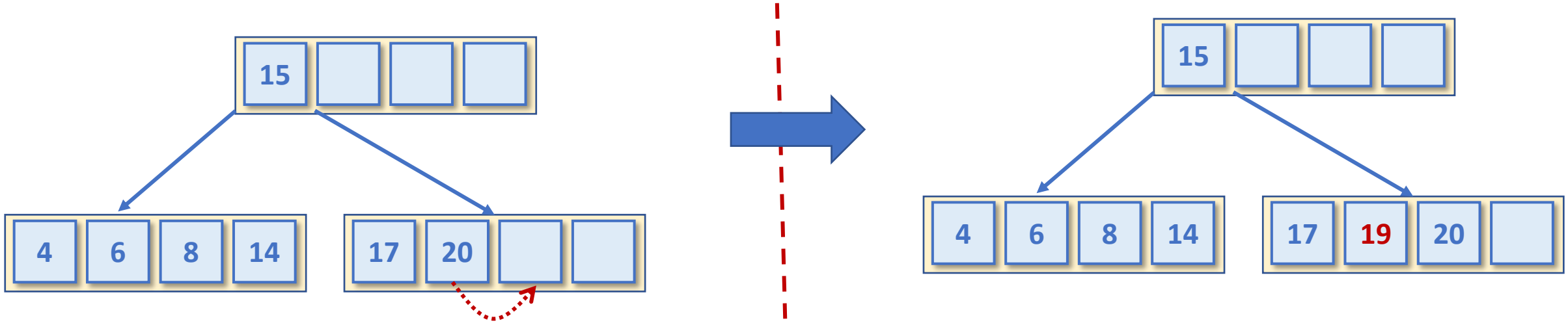
Árbol B. Insertar. Caso 1

- La página donde se quiere insertar tiene m <2n claves
- La inserción siempre se produce en la hojas
- Se abre hueco par el elemento a insertar, desplazando los elementos de clave mayor una posición a la derecha si es preciso
- Ejemplo: insertar el 19



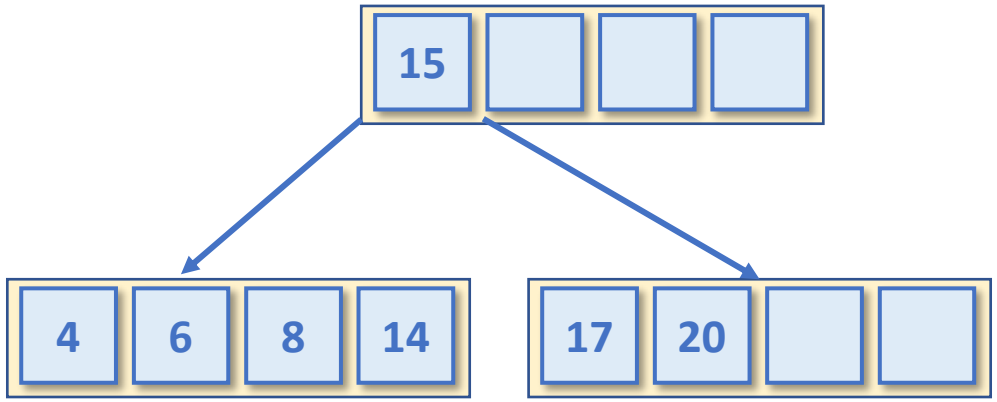
Árbol B. Insertar. Caso 1

- La página donde se quiere insertar tiene m <2n claves
- La inserción siempre se produce en la hojas
- Se abre hueco par el elemento a insertar, desplazando los elementos de clave mayor una posición a la derecha si es preciso
- Ejemplo: insertar el 19



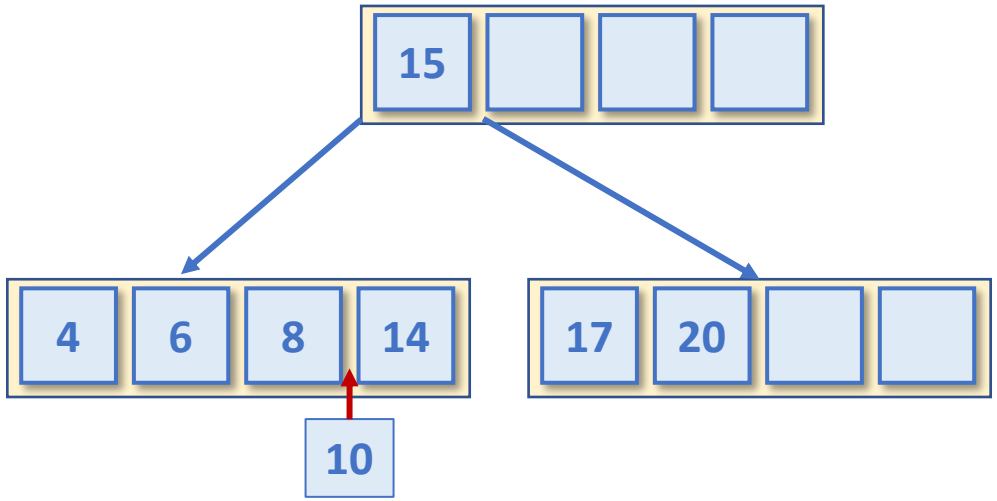
Árbol B. Insertar. Caso 2

- La página donde se quiere insertar tiene m = 2*n claves
- Procedimiento
  - Simular la inserción y dividir la hoja en dos, propagando el elemento central a la página padre
  - Cada nueva página tendrá las primeras (m+1)/2 y la últimas (m+1)/2 claves respectivamente
- Repetir el proceso de forma recursiva si es necesario
  - Desdoblar la raíz es la única forma de aumentar la altura del árbol
- Ejemplo: insertar el 10



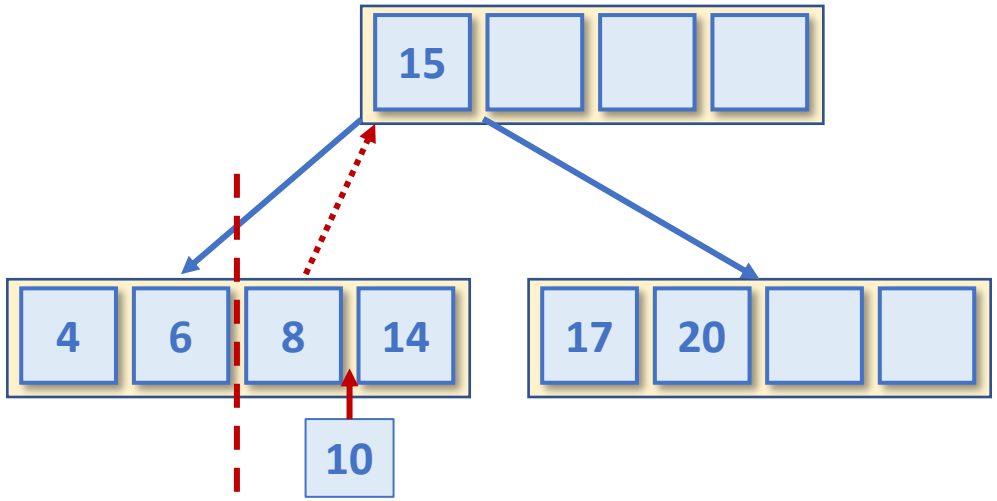
Árbol B. Insertar. Caso 2

- La página donde se quiere insertar tiene m = 2*n claves
- Procedimiento
  - Simular la inserción y dividir la hoja en dos, propagando el elemento central a la página padre
  - Cada nueva página tendrá las primeras (m+1)/2 y la últimas (m+1)/2 claves respectivamente
- Repetir el proceso de forma recursiva si es necesario
  - Desdoblar la raíz es la única forma de aumentar la altura del árbol
- Ejemplo: insertar el 10



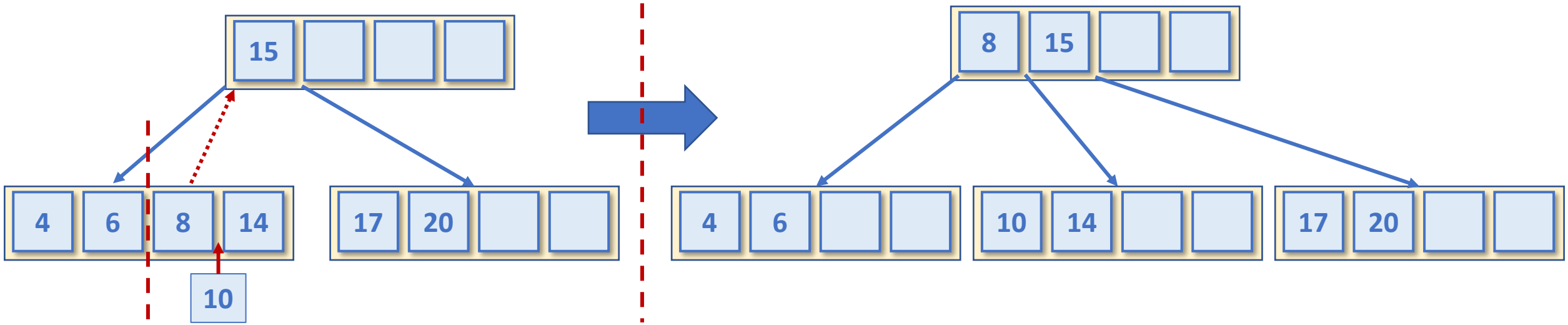
Árbol B. Insertar. Caso 2

- La página donde se quiere insertar tiene m = 2*n claves
- Procedimiento
  - Simular la inserción y dividir la hoja en dos, propagando el elemento central a la página padre
  - Cada nueva página tendrá las primeras (m+1)/2 y la últimas (m+1)/2 claves respectivamente
- Repetir el proceso de forma recursiva si es necesario
  - Desdoblar la raíz es la única forma de aumentar la altura del árbol
- Ejemplo: insertar el 10



Árbol B. Insertar. Caso 2

- La página donde se quiere insertar tiene m = 2*n claves
- Procedimiento
  - Simular la inserción y dividir la hoja en dos, propagando el elemento central a la página padre
  - Cada nueva página tendrá las primeras (m+1)/2 y la últimas (m+1)/2 claves respectivamente
- Repetir el proceso de forma recursiva si es necesario
  - Desdoblar la raíz es la única forma de aumentar la altura del árbol
- Ejemplo: insertar el 10



Árbol B. Insertar. Complejidad temporal

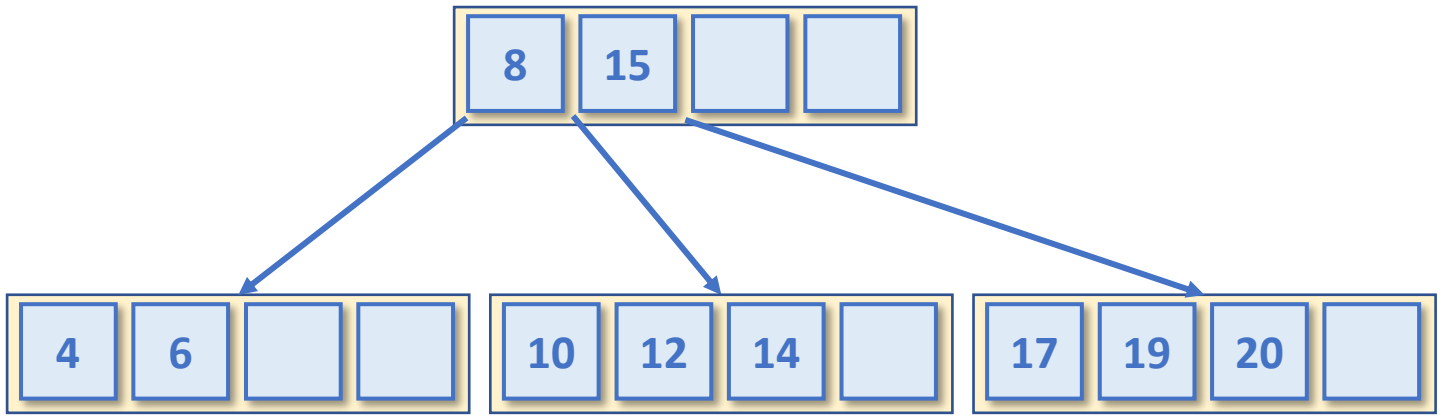
- Caso mejor
  - Hay espacio en la una página hoja en un árbol de altura mínima
  - Complejidad del árbol de altura mínima → $O(\log_{2n}(N))$
  - Complejidad de insertar en la página → $O(m)$
  - $O(\log_{2n}(N)) + O(m) = O(\log_{2n}(N))$
- Caso peor
  - Se inserta en un árbol degenerado y se desdoblan todas las páginas desde las hojas hasta la raíz
  - Complejidad del árbol degenerado → $O(\log_{n}(N))$
  - Complejidad de cada página→ $O(n)$
  - $O(\log_{n}(N)) * O(n) = O(\log_{n}(N))$

Árbol B. Insertar. Ejercicios

- Partiendo de un árbol B-2 vacío insertar la secuencia de claves: 6, 11, 5, 4, 8, 9, 12, 21, 14, 10, 19, 28, 3, 17, 32, 15, 16, 26, 27

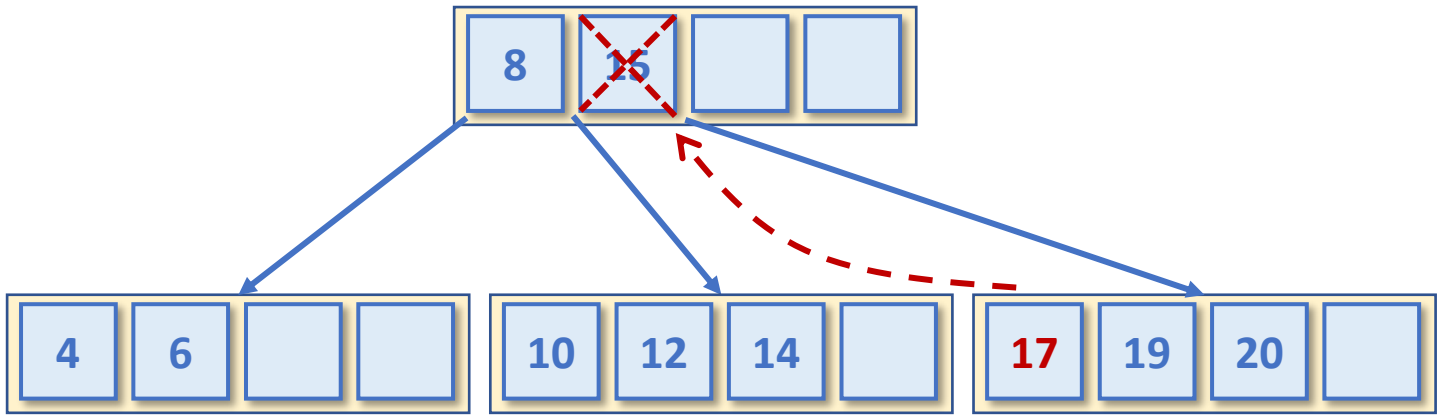
Árbol B. Eliminar. Caso 1

- El elemento se encuentra en una página que **no es una hoja**
  - Buscar el elemento extremo izquierdo del subárbol derecho
  - El elemento a buscar se encontrará en una hoja y será el menor de los mayores
  - La hoja que dona el elemento no está en situación critica
  - Se sustituye el elemento por el encontrado y se borrará de la página donante
- Ejemplo: Eliminar el 15



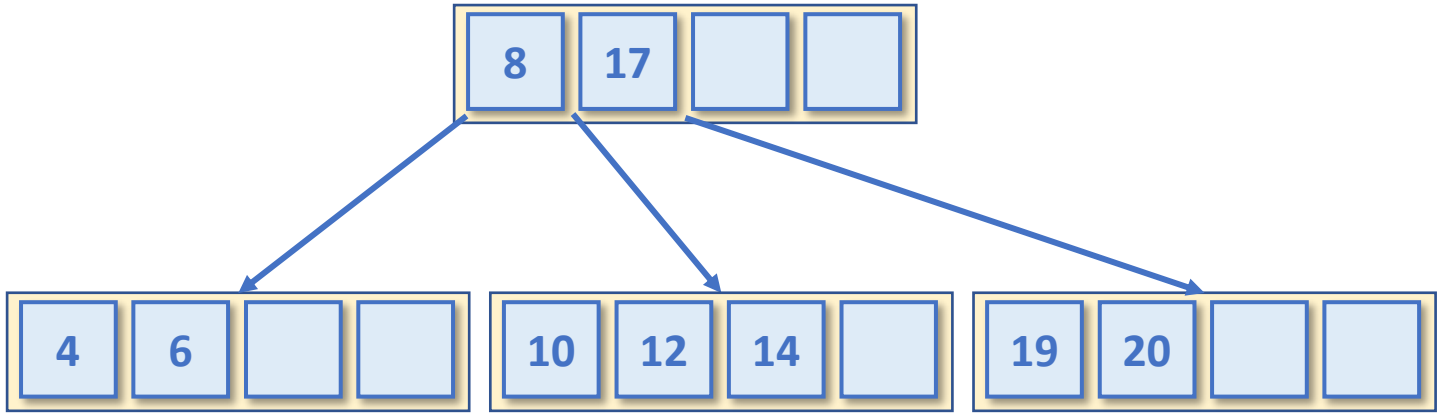
Árbol B. Eliminar. Caso 1

- El elemento se encuentra en una página que **no es una hoja**
  - Buscar el elemento extremo izquierdo del subárbol derecho
  - El elemento a buscar se encontrará en una hoja y será el menor de los mayores
  - La hoja que dona el elemento no está en situación critica
  - Se sustituye el elemento por el encontrado y se borrará de la página donante
- Ejemplo: Eliminar el 15



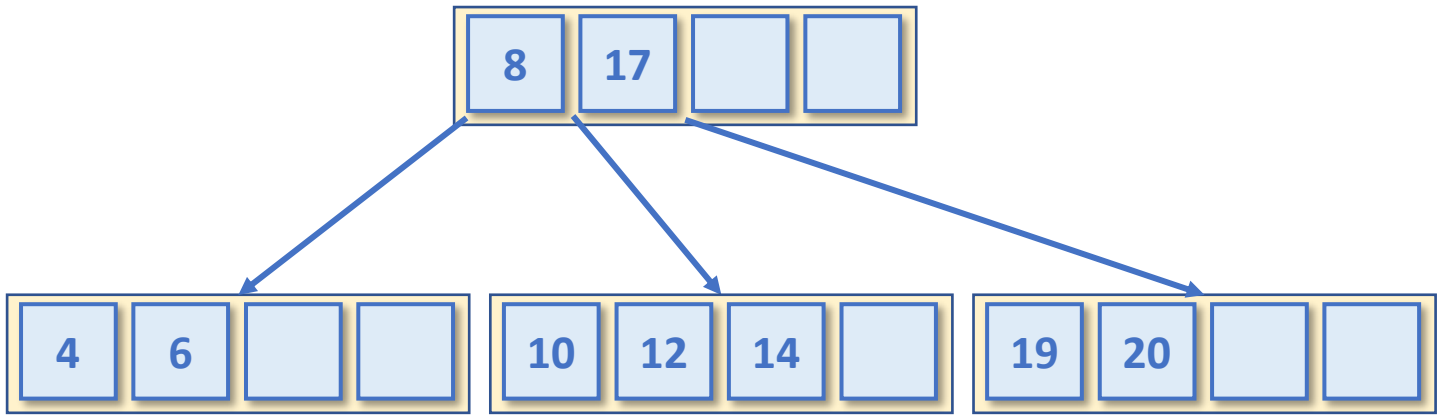
Árbol B. Eliminar. Caso 1

- El elemento se encuentra en una página que **no es una hoja**
  - Buscar el elemento extremo izquierdo del subárbol derecho
  - El elemento a buscar se encontrará en una hoja y será el menor de los mayores
  - La hoja que dona el elemento no está en situación critica
  - Se sustituye el elemento por el encontrado y se borrará de la página donante
- Ejemplo: Eliminar el 15



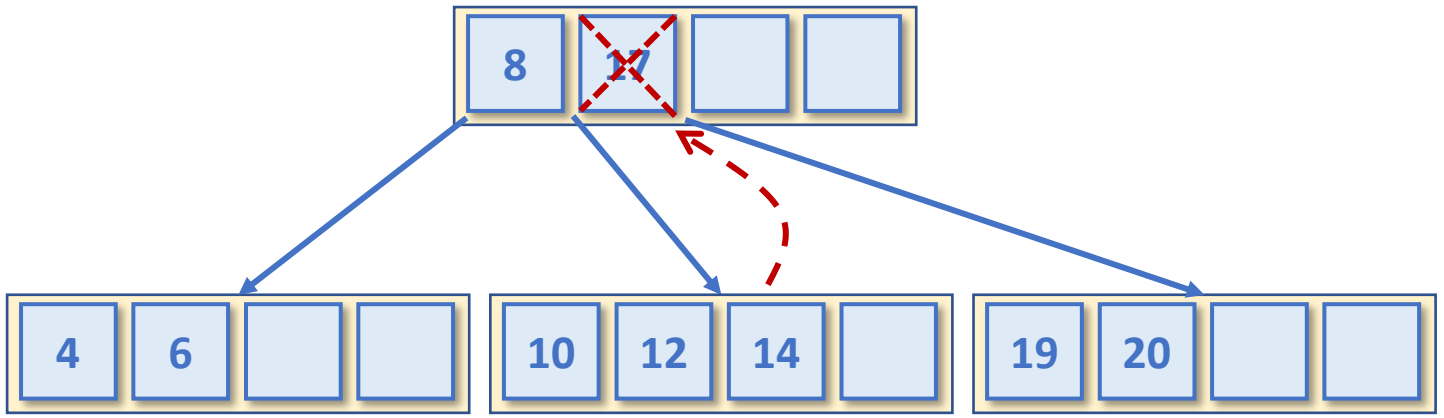
Árbol B. Eliminar. Caso 2

- El elemento se encuentra en una página que **no es una hoja**
  - Buscar el elemento extremo izquierdo del subárbol derecho
  - El elemento a buscar se encontrará en una hoja y será el menor de los mayores
  - La hoja que dona el elemento está en situación critica
  - Se sustituye por antecesor (elemento extremo derecho subárbol izquierdo).
- Ejemplo: Eliminar el 17



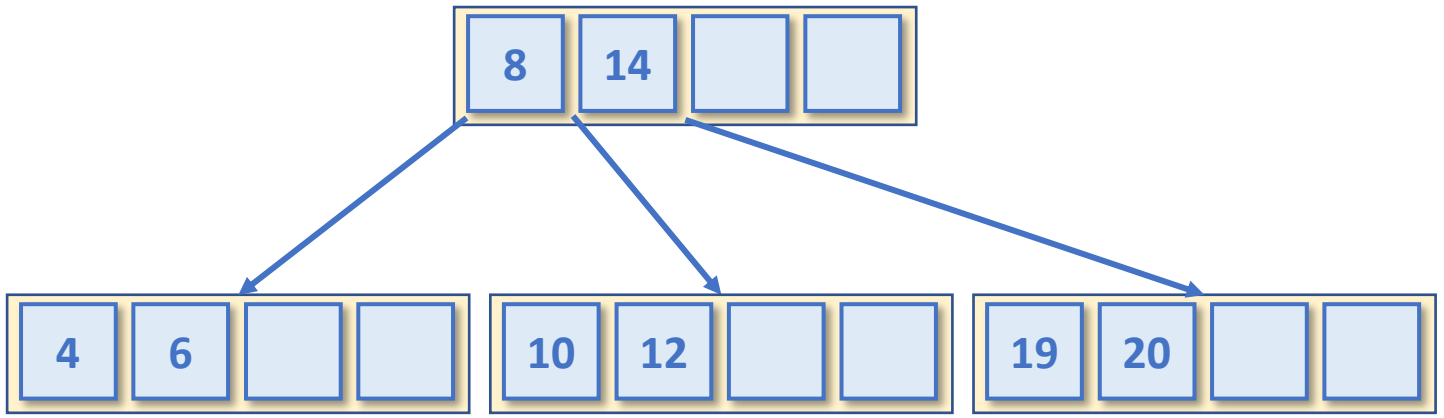
Árbol B. Eliminar. Caso 2

- El elemento se encuentra en una página que **no es una hoja**
  - Buscar el elemento extremo izquierdo del subárbol derecho
  - El elemento a buscar se encontrará en una hoja y será el menor de los mayores
  - La hoja que dona el elemento está en situación critica
  - Se sustituye por antecesor (elemento extremo derecho subárbol izquierdo).
- Ejemplo: Eliminar el 17



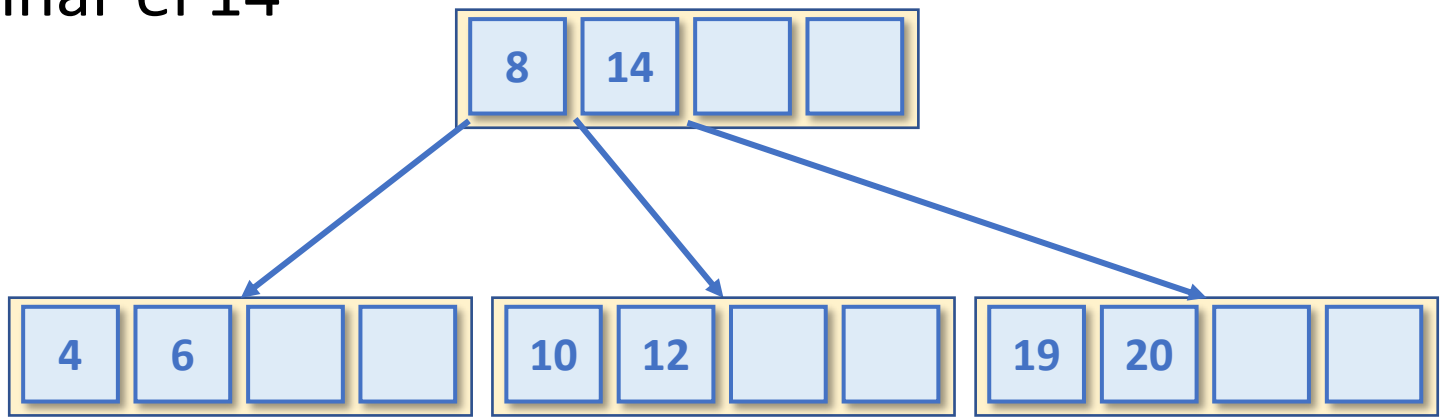
Árbol B. Eliminar. Caso 2

- El elemento se encuentra en una página que **no es una hoja**
  - Buscar el elemento extremo izquierdo del subárbol derecho
  - El elemento a buscar se encontrará en una hoja y será el menor de los mayores
  - La hoja que dona el elemento está en situación critica
  - Se sustituye por antecesor (elemento extremo derecho subárbol izquierdo)
- Ejemplo: Eliminar el 17



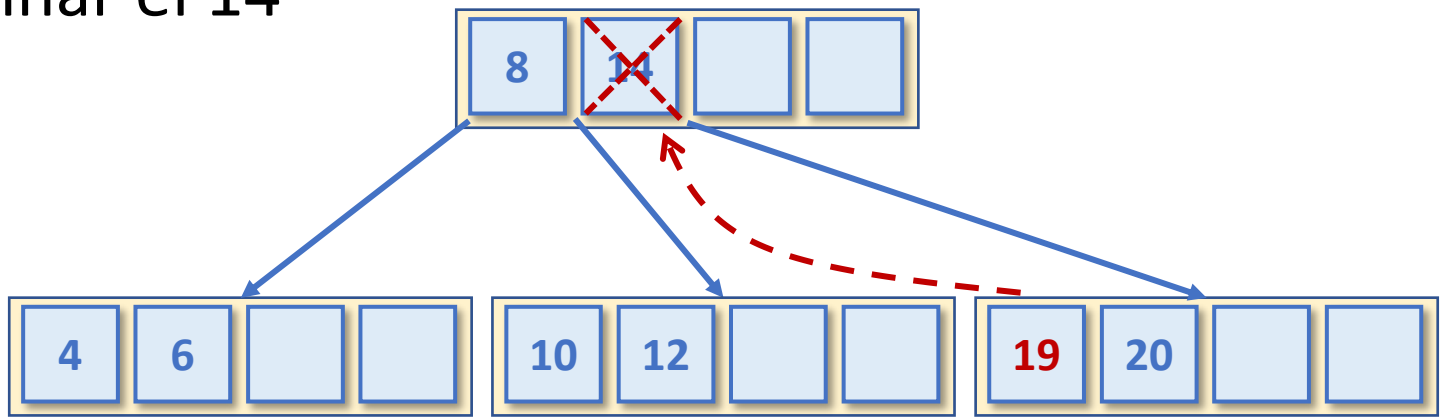
Árbol B. Eliminar. Caso 3

- El elemento se encuentra en una página que **no es una hoja**
  - Buscar el elemento extremo izquierdo del subárbol derecho y la página esta en situación crítica (página sucesora)
  - Busca el elemento extremo derecho subárbol izquierdo y la página está en situación crítica (página antecesora)
  - Se reemplaza por el elemento de la página sucesora aunque este en situación crítica
- Ejemplo: Eliminar el 14



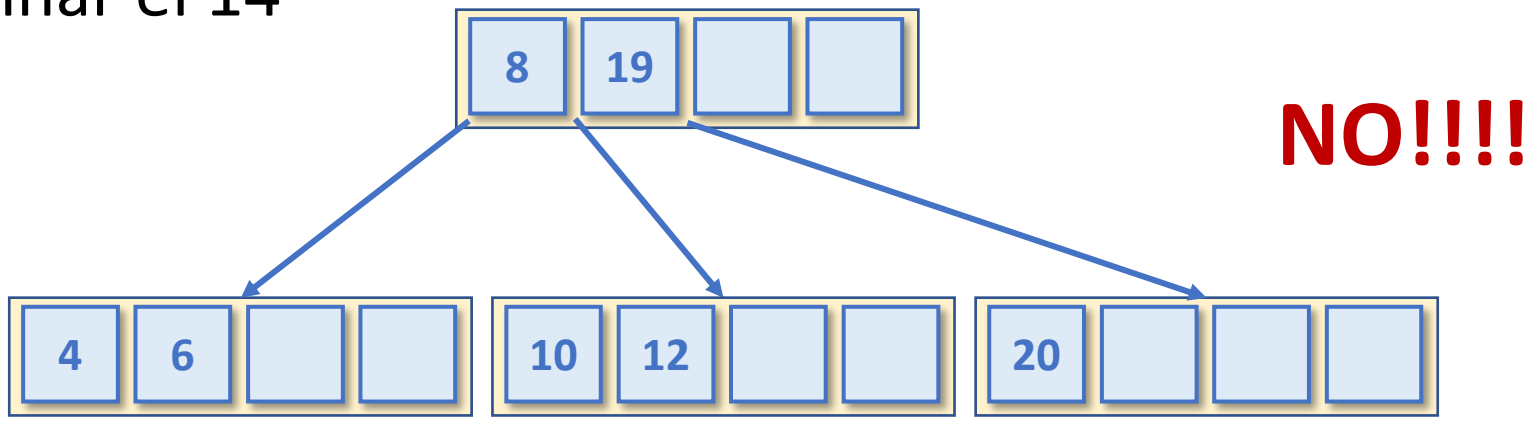
Árbol B. Eliminar. Caso 3

- El elemento se encuentra en una página que **no es una hoja**
  - Buscar el elemento extremo izquierdo del subárbol derecho y la página esta en situación crítica (página sucesora)
  - Busca el elemento extremo derecho subárbol izquierdo y la página está en situación crítica (página antecesora)
  - Se reemplaza por el elemento de la página sucesora aunque este en situación crítica
- Ejemplo: Eliminar el 14



Árbol B. Eliminar. Caso 3

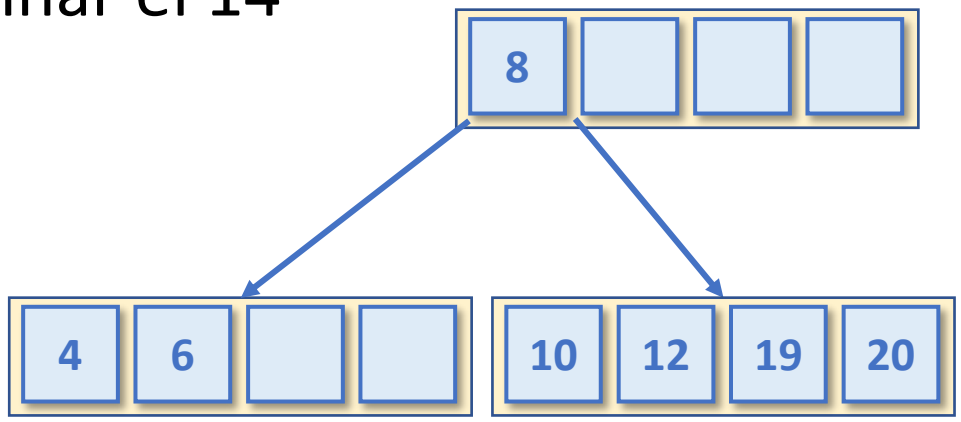
- El elemento se encuentra en una página que **no es una hoja**
  - Buscar el elemento extremo izquierdo del subárbol derecho y la página esta en situación crítica (página sucesora)
  - Busca el elemento extremo derecho subárbol izquierdo y la página está en situación crítica (página antecesora)
  - Se reemplaza por el elemento de la página sucesora aunque este en situación crítica
- Ejemplo: Eliminar el 14



NO!!!!

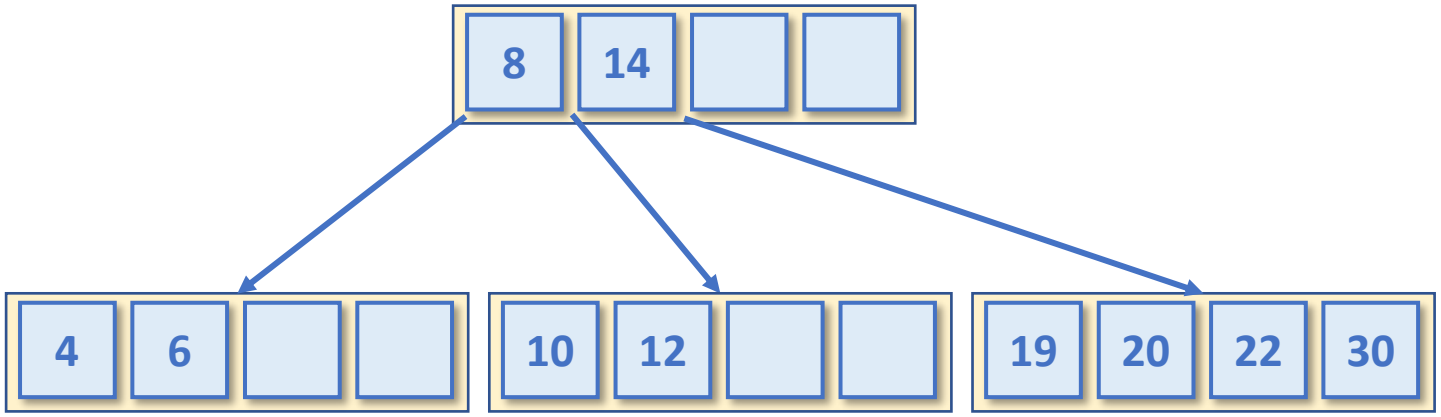
Árbol B. Eliminar. Caso 3

- El elemento se encuentra en una página que **no es una hoja**
  - Buscar el elemento extremo izquierdo del subárbol derecho y la página esta en situación crítica (página sucesora)
  - Busca el elemento extremo derecho subárbol izquierdo y la página está en situación crítica (página antecesora)
  - Se reemplaza por el elemento de la página sucesora aunque este en situación crítica
- Ejemplo: Eliminar el 14



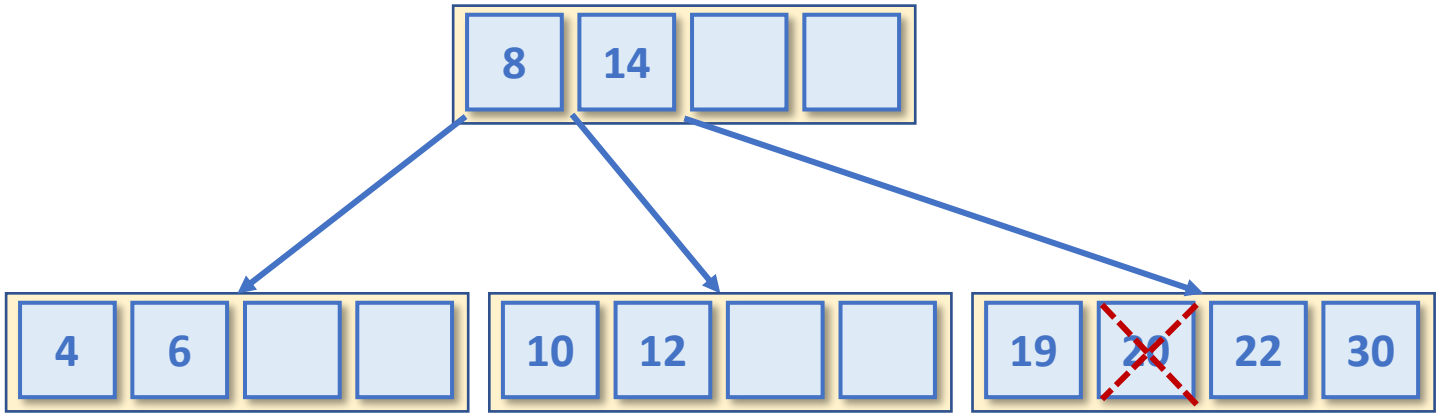
Árbol B. Eliminar. Caso 4

- El elemento se encuentra en una página que **es una hoja**
  - La página tiene m>n claves
  - Se elimina la clave y se reestructura la página se es necesario
- Ejemplo: Eliminar el 20



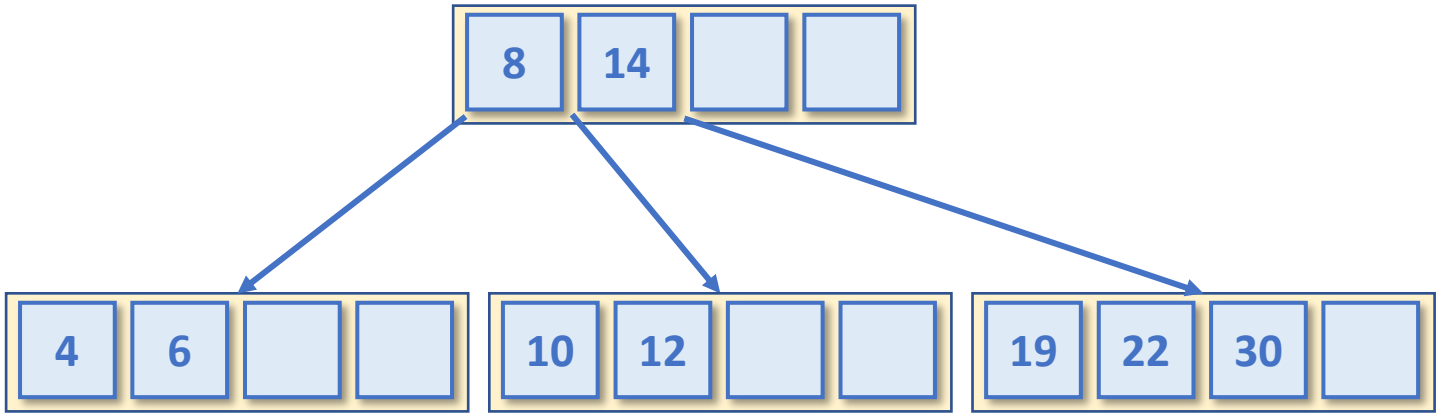
Árbol B. Eliminar. Caso 4

- El elemento se encuentra en una página que **es una hoja**
  - La página tiene m>n claves
  - Se elimina la clave y se reestructura la página se es necesario
- Ejemplo: Eliminar el 20



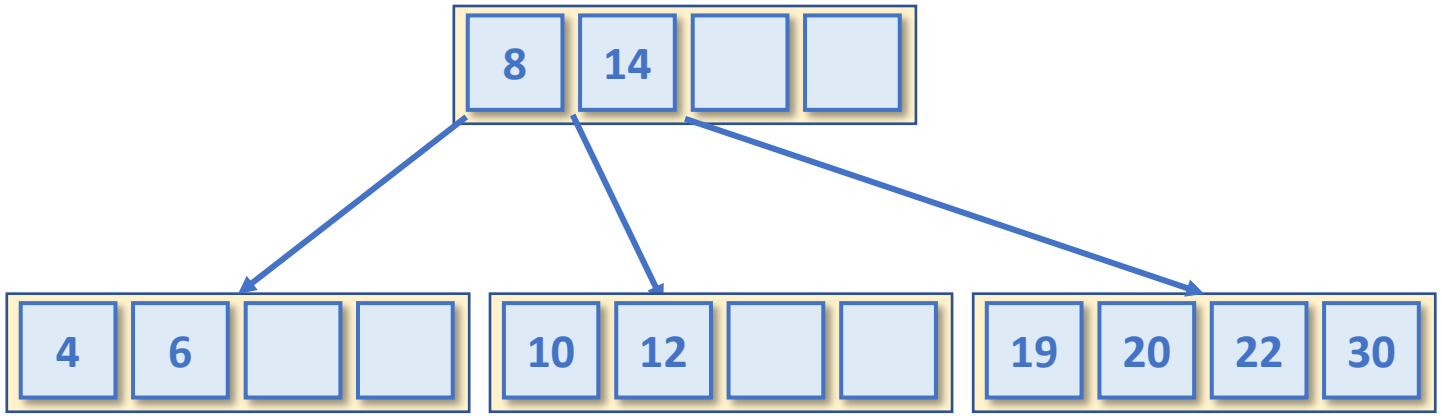
Árbol B. Eliminar. Caso 4

- El elemento se encuentra en una página que **es una hoja**
  - La página tiene m>n claves
  - Se elimina la clave y se reestructura la página se es necesario
- Ejemplo: Eliminar el 20



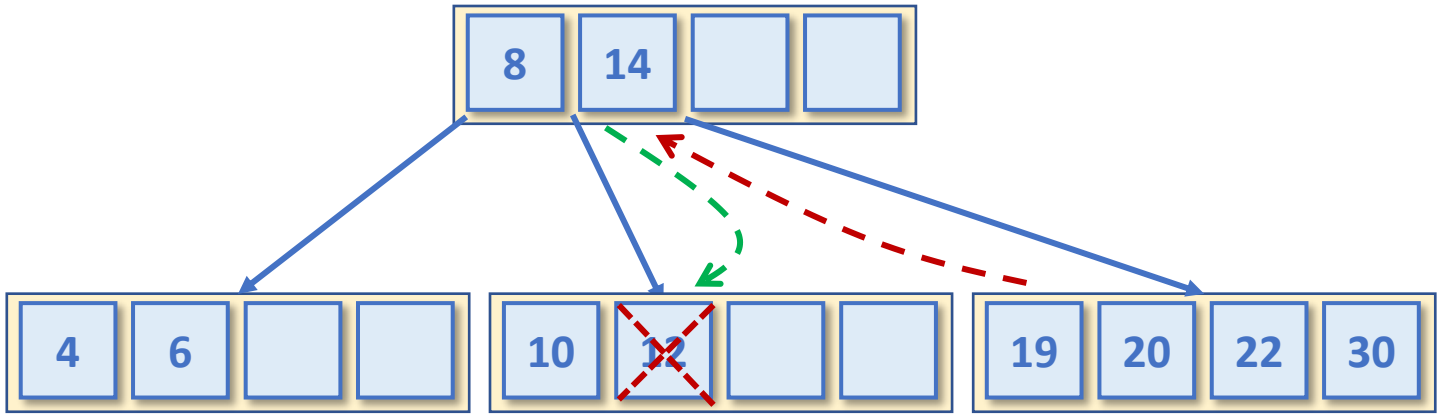
Árbol B. Eliminar. Caso 5

- El elemento se encuentra en una página que **es una hoja**
  - La página tiene m=n claves. Está en situación crítica
  - Se busca entre las páginas hoja adyacentes alguna que tenga m>n claves y se le pide la cesión de una de ellas. Primero se consulta la derecha y luego la izquierda
  - El elemento cedido se eleva a la página padre y este cede la clave inmediatamente anterior o posterior al cedido a la página donde se elimina la clave
- Ejemplo: Eliminar el 12



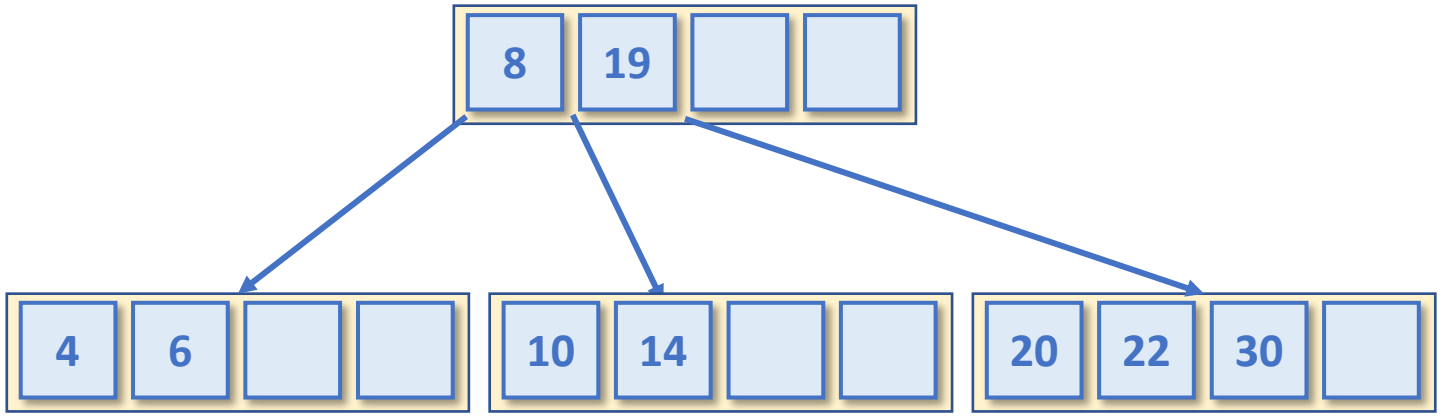
Árbol B. Eliminar. Caso 5

- El elemento se encuentra en una página que **es una hoja**
  - La página tiene m=n claves. Está en situación crítica
  - Se busca entre las páginas hoja adyacentes alguna que tenga m>n claves y se le pide la cesión de una de ellas. Primero se consulta la derecha y luego la izquierda
  - El elemento cedido se eleva a la página padre y este cede la clave inmediatamente anterior o posterior al cedido a la página donde se elimina la clave
- Ejemplo: Eliminar el 12



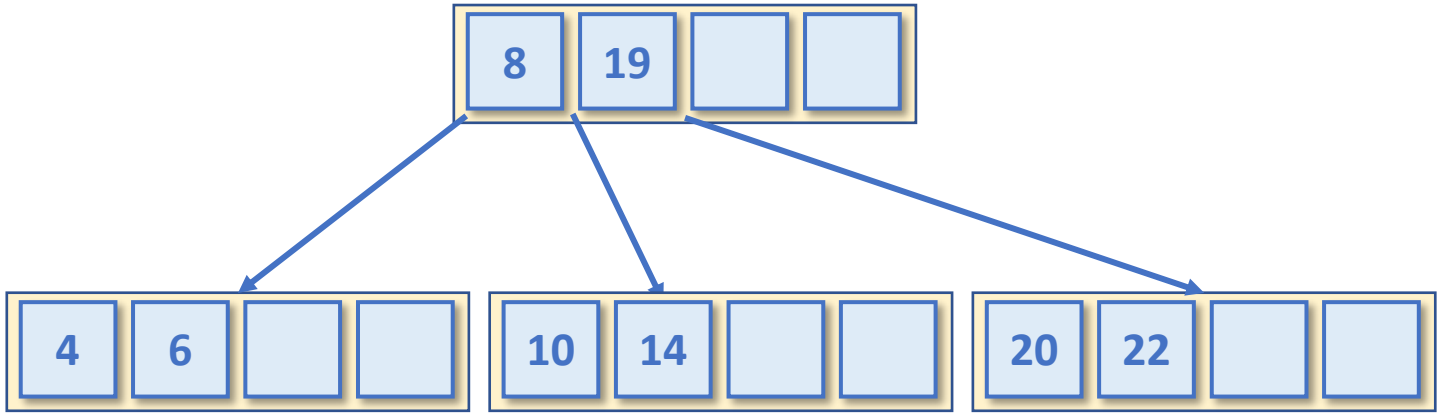
Árbol B. Eliminar. Caso 5

- El elemento se encuentra en una página que **es una hoja**
  - La página tiene m=n claves. Está en situación crítica
  - Se busca entre las páginas hoja adyacentes alguna que tenga m>n claves y se le pide la cesión de una de ellas. Primero se consulta la derecha y luego la izquierda
  - El elemento cedido se eleva a la página padre y este cede la clave inmediatamente anterior o posterior al cedido a la página donde se elimina la clave
- Ejemplo: Eliminar el 12



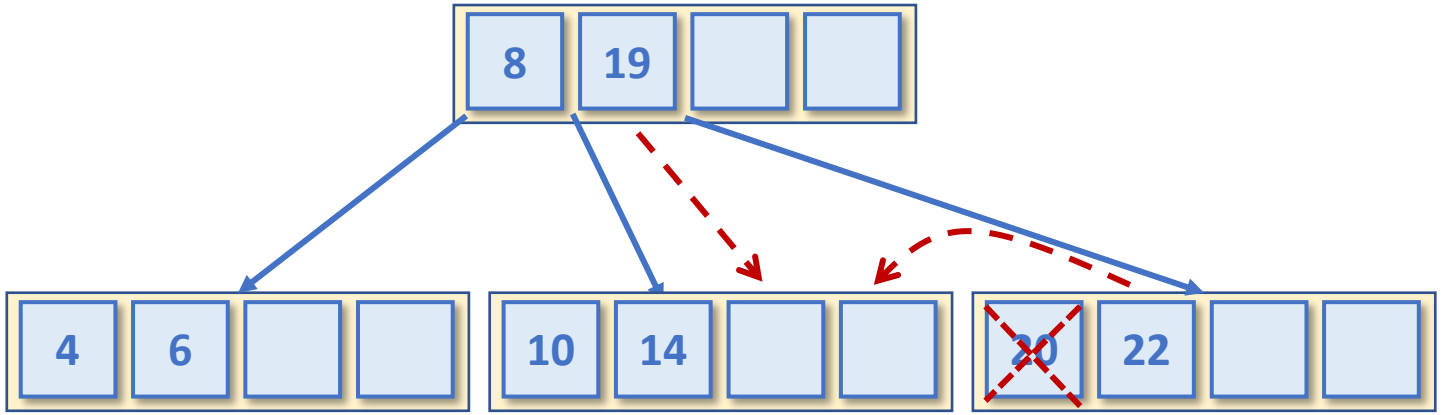
Árbol B. Eliminar. Caso 6

- El elemento se encuentra en una página que **es una hoja**
  - La página tiene m=n claves. Está en situación crítica
  - Las dos página adyacentes se encuentran en situación crítica
  - La página se fusiona con la vecina derecha y si esta no existe con la izquierda
  - La clave del padre que implica la generación de dicha página también pasa a formar parte de la nueva fusión
  - El borrado de la clave del padre fuerza un borrado recursivo ascendente que puede alcanzar a la raíz, en cuyo caso disminuye la altura del árbol
- Ejemplo: Eliminar el 20



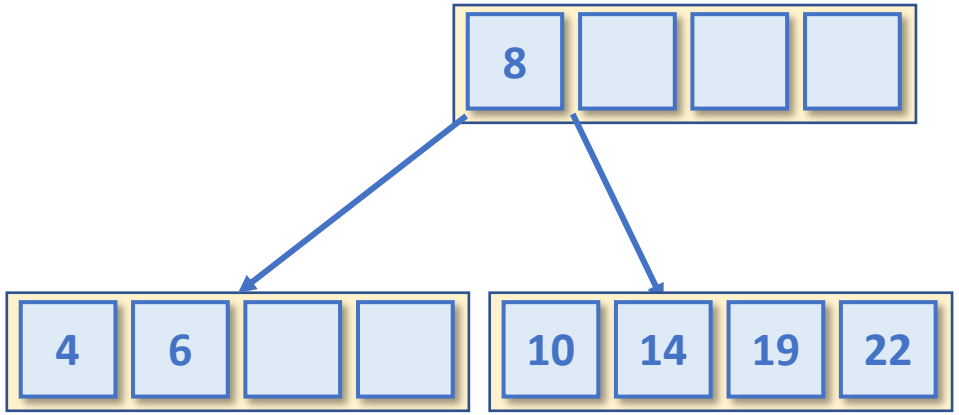
Árbol B. Eliminar. Caso 6

- El elemento se encuentra en una página que **es una hoja**
  - La página tiene m=n claves. Está en situación crítica
  - Las dos página adyacentes se encuentran en situación crítica
  - La página se fusiona con la vecina derecha y si esta no existe con la izquierda
  - La clave del padre que implica la generación de dicha página también pasa a formar parte de la nueva fusión
  - El borrado de la clave del padre fuerza un borrado recursivo ascendente que puede alcanzar a la raíz, en cuyo caso disminuye la altura del árbol
- Ejemplo: Eliminar el 20



Árbol B. Eliminar. Caso 6

- El elemento se encuentra en una página que **es una hoja**
  - La página tiene m=n claves. Está en situación crítica
  - Las dos página adyacentes se encuentran en situación crítica
  - La página se fusiona con la vecina derecha y si esta no existe con la izquierda
  - La clave del padre que implica la generación de dicha página también pasa a formar parte de la nueva fusión
  - El borrado de la clave del padre fuerza un borrado recursivo ascendente que puede alcanzar a la raíz, en cuyo caso disminuye la altura del árbol
- Ejemplo: Eliminar el 20



Árbol B. Eliminar. Complejidad temporal

- Caso mejor
  - Árbol de altura mínima
  - Complejidad del árbol de altura mínima → $O(\log_{2n}(N))$
  - Complejidad de insertar en la página → $O(m)$
  - $O(\log_{2n}(N)) + O(m) = O(\log_{2n}(N))$
- Caso peor
  - Árbol de altura máxima
  - Complejidad del árbol degenerado → $O(\log_{n}(N))$
  - Complejidad de cada página→ $O(n)$
  - $O(\log_{n}(N)) * O(n) = O(\log_{n}(N))$

Árbol B. Eliminar. Ejercicios

- Partiendo de un árbol B-2 creado en el ejercicio anterior, borrar la siguiente secuencia de claves: 11, 15, 6, 16, 10, 12, 28, 27