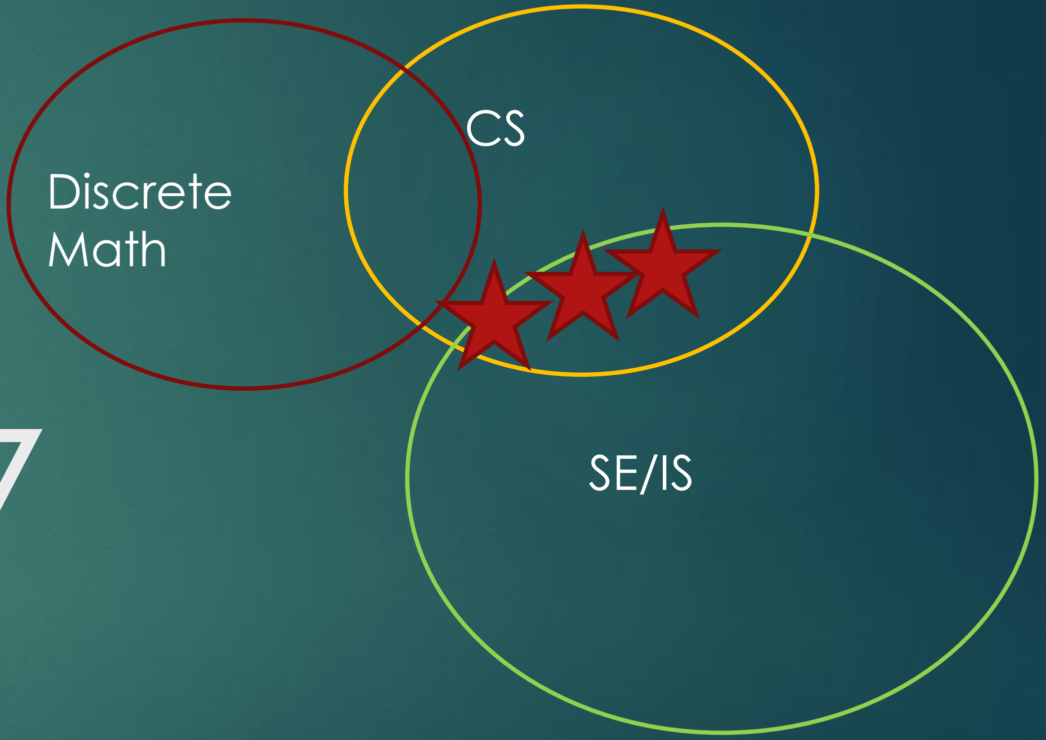
Árboles I: EIF-203-I-2017

DR. CARLOS LORÍA-SÁENZ

ESCUELA DE INFORMÁTICA, UNA



Objetivos Generales

- Conceptos y aplicaciones
- Propiedades básicas
- Representación computacional (ADT Árbol)
- Árboles orientados
- Árboles de expresiones y recorridos
- Árboles Binarios
- Árboles Binarios de Búsqueda
- Árboles de Huffman

Objetivos de esta parte

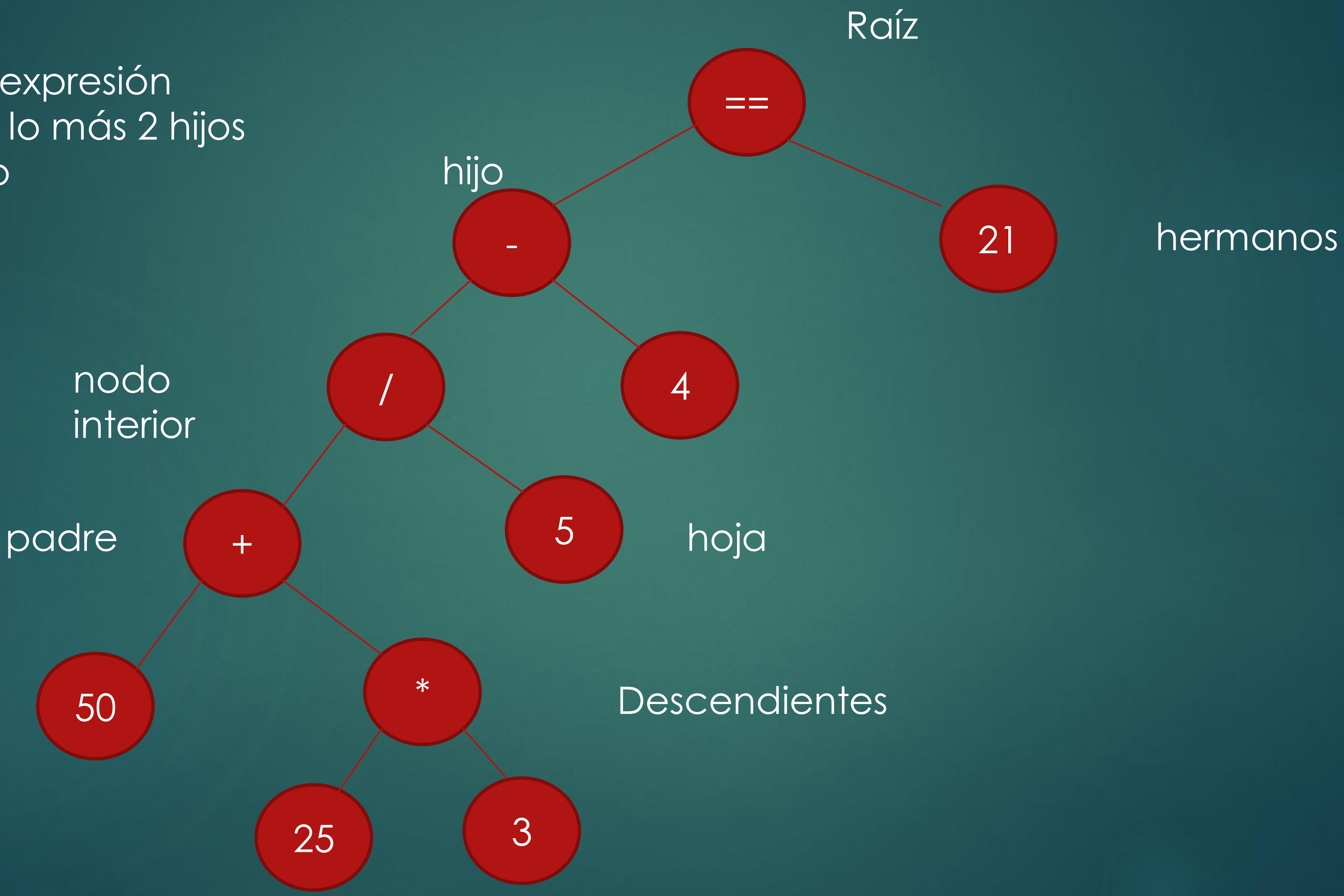
- Conceptos intuitivos de grafo y ejemplos de uso informático
- Ejemplos comunes
  - File System
  - HTML
  - Compilación/evaluación de expresiones
- Árboles orientados
- Árbol de expresiones y recorridos

Ejemplo: árbol de expresión

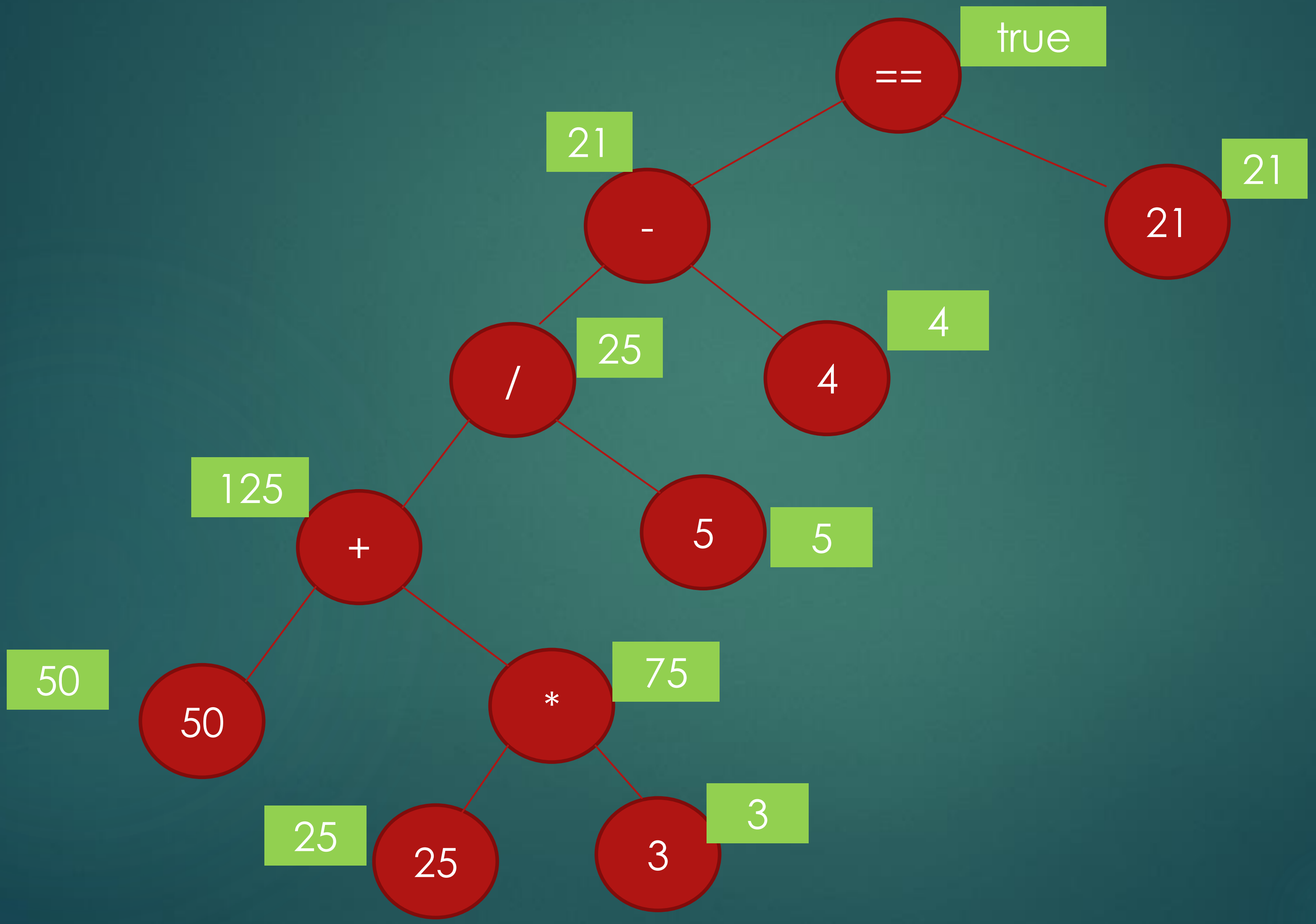
- Considere evaluar $(50 + 25 * 3)/5 - 4 == 21$
- Para evaluar hay que "*leer toda expresión*"
- Ineficiente. Problemas con la precedencia de operadores
- Las computadoras no evalúan así
- Se quiere evaluar de izquierda a derecha
- <u>Solución</u>: Un árbol y un recorrido
- Una estructura jerárquica los de abajo se evalúan primero

$$(50 + 25 * 3)/5 - 4 == 21$$

- Árbol de expresión
- Binario: a lo más 2 hijos por nodo



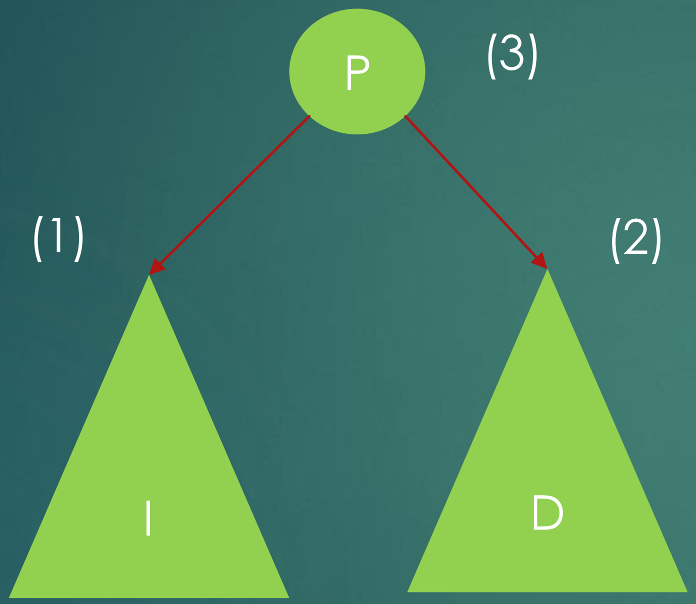
Evaluación



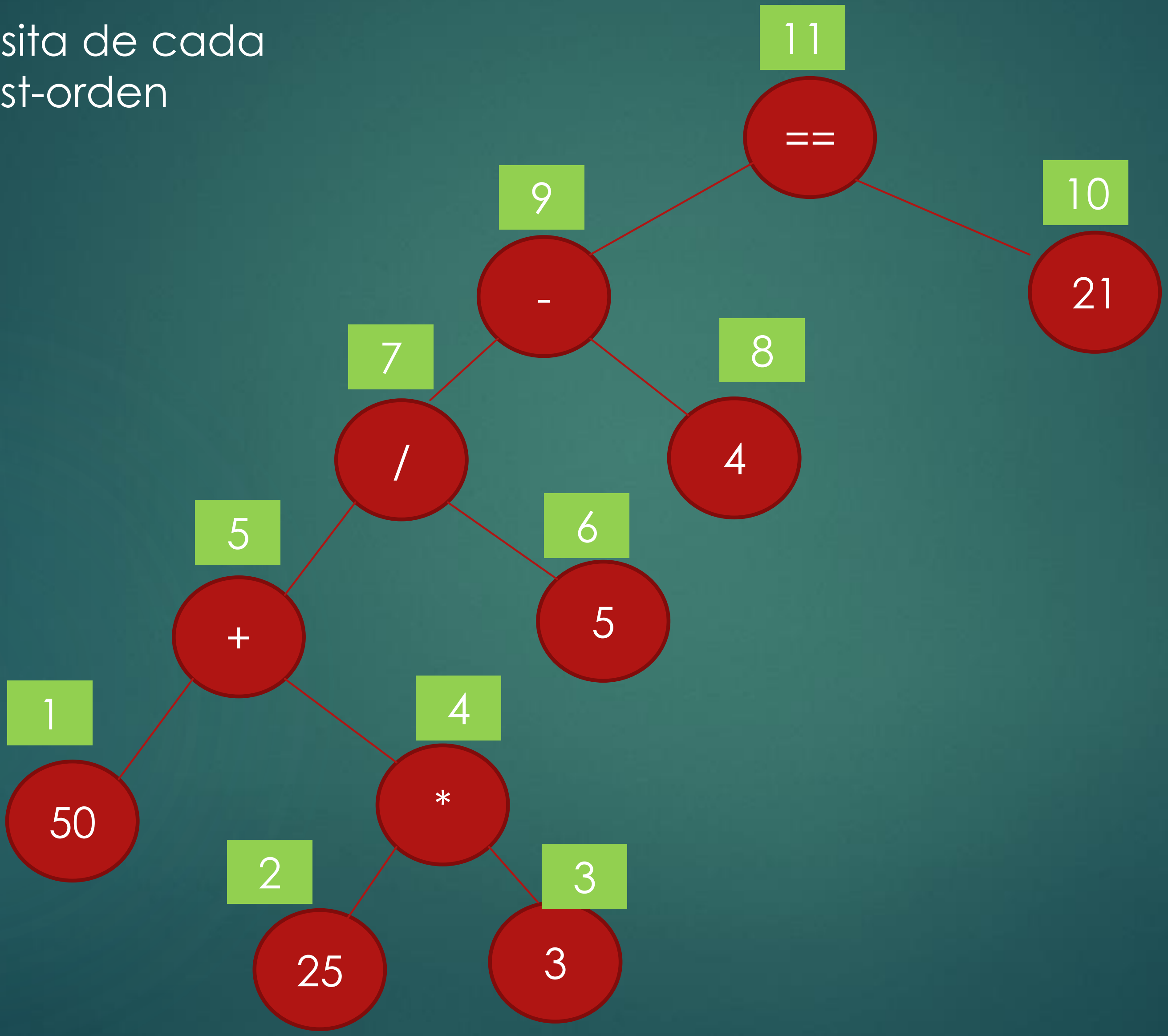
Recorrido post-Orden

- Dado un árbol T
- Si T es consta de una hoja P procese el valor de la hoja
- Si no es hoja:
  - Procesar el hijo izquierdo I
  - Procesar el hijo derecho D
  - Procesar el nodo actual P
- En resumen: $I - D - P$

Post-orden



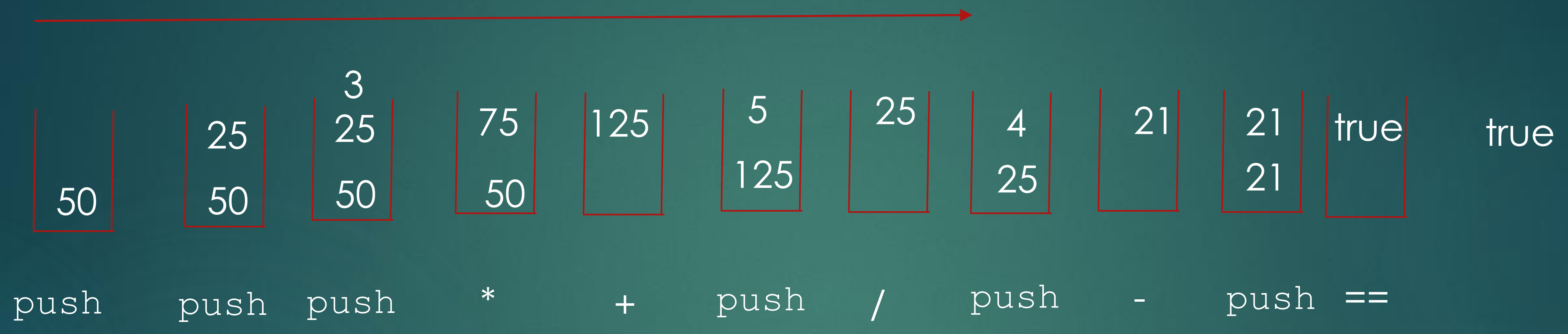
Orden de visita de cada nodo en post-orden



Notación post-fija (polaca inversa)

- Es la que se obtiene de recorrer en post-orden un árbol de expresión:
- En el ejemplo anterior: $50\ 25\ 3\ *\ +5\ /\ 4-\ 21\ ==$
- Fácil de ejecutar usando una pila

$$50\ 25\ 3\ *\ +\ 5\ /\ 4\ -\ 21\ ==$$

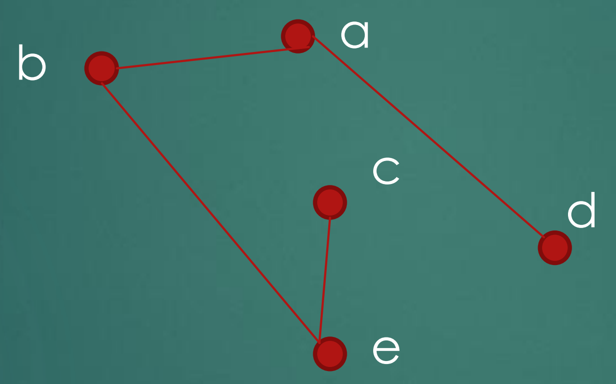

50
25
50
3
25
50
75
50
125
5
125
25
4
25
21
21
21
true
true
push
push
push
*
+
push
/
push
-
push
==

Tres recorridos

- En-orden: `I-P-D`
- Pre-orden: `P-I-D`
- Post-orden: `I-D-P`
- Las otras combinaciones se descartan asumiendo la izquierda primera que la derecha siempre

Árbol Libre

- Grafo simple tal que entre dos vértices distintos existe siempre existe un camino que los conecta



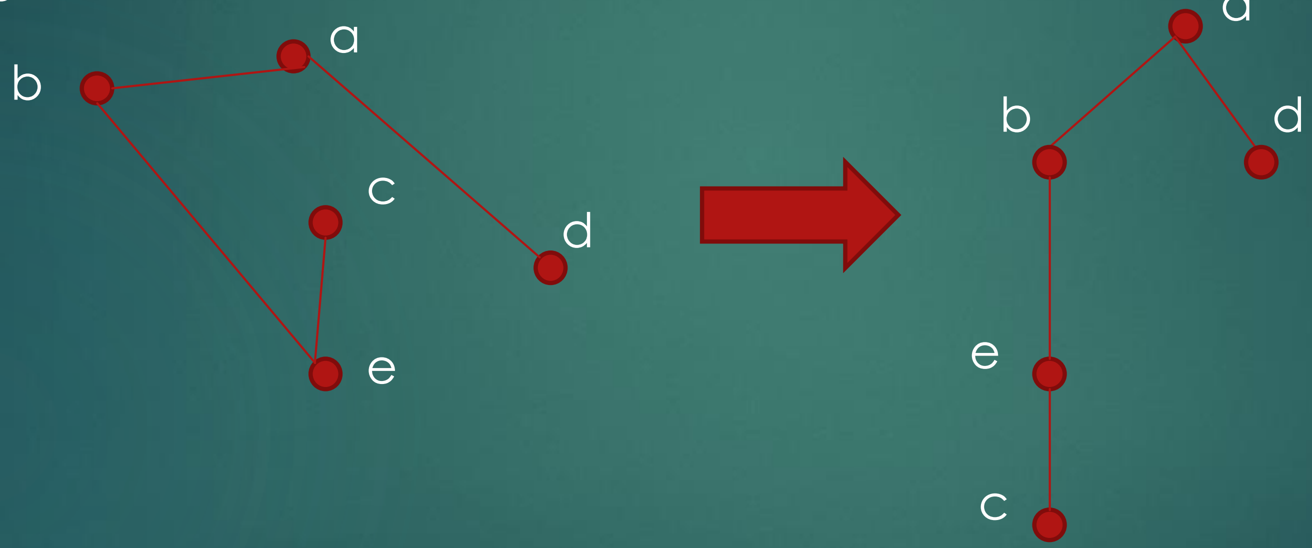
Árbol

- Mínima forma de conectar n nodos
- Tiene $n-1$ arcos
- No puede tener ciclos (probar)

Árbol con raíz y orientado

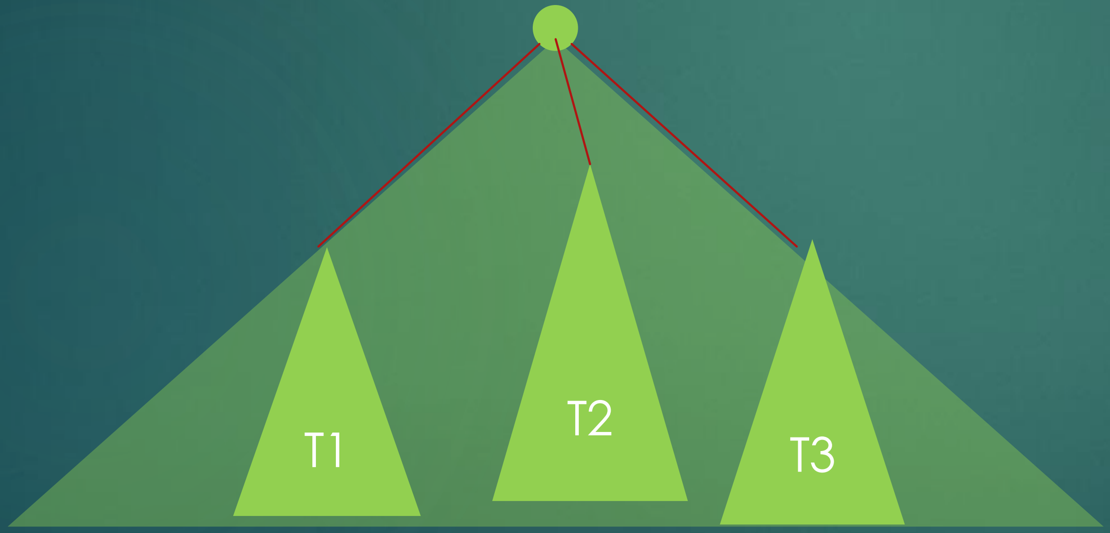
- Se fija una orientación (arriba-abajo-izquierda-derecha)
- Se escoge un nodo cualquiera, la raíz
- Sus sucesores se ordenan de izquierda a derecha y cuelgan de la raíz: se llaman hijos
- Lo mismo se repite con cada hijo
- Los nodos finales (sin hijos) se llaman hojas. Los otros interiores

Árbol orientado



Sub-árbol: árbol es una estructura recursiva

- Si T es un árbol orientado y n un nodo entonces cada grafo que tenga a n como raíz es también un árbol.
- Se llama sub-árbol



Niveles, altura

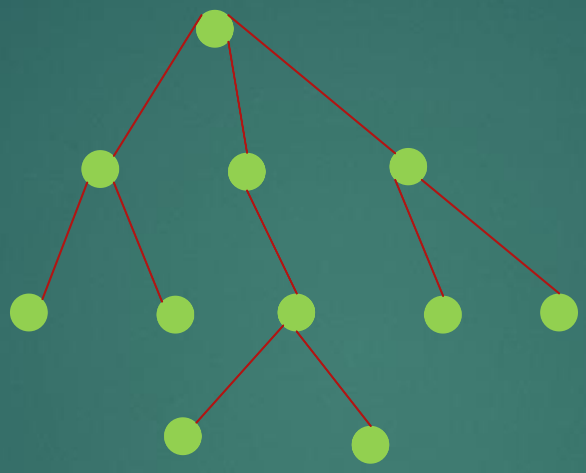
Nivel

0

1

2

3



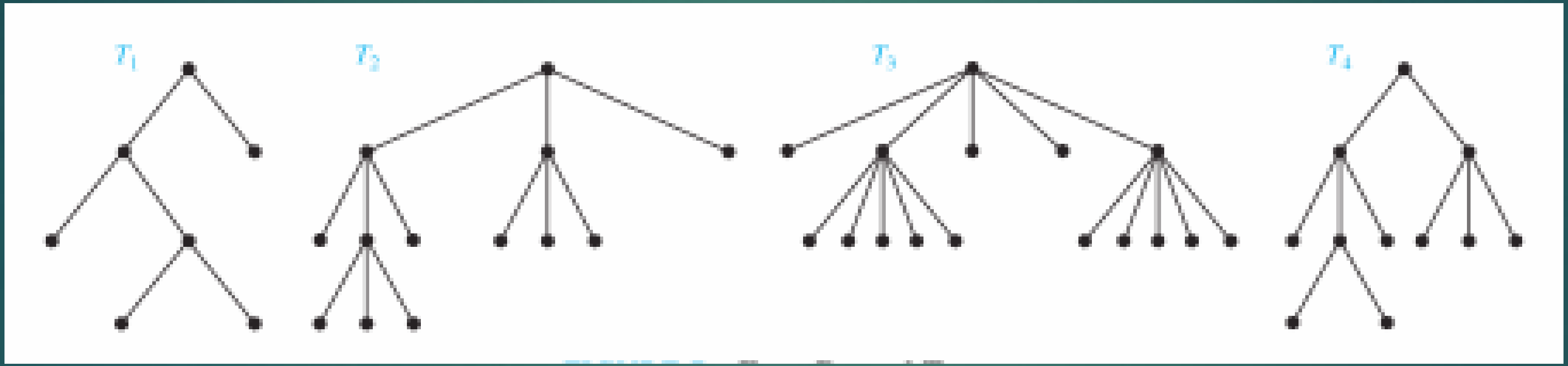
Altura: máximo nivel alcanzable. Camino más largo desde la raíz a una hoja

Árbol m-ario

- Un árbol orientado se dice ser m-ario si cada nodo interno tiene a lo más m hijos
- Se dice *lleno* si cada nodo interior tiene exactamente m hijos
- Se dice "completo" si está lleno en cada nivel, excepto tal vez, en el último nivel y los nodos están acomodados lo más a la izquierda posible
- Cuando $m = 2$ se llama árbol binario

Ejemplos

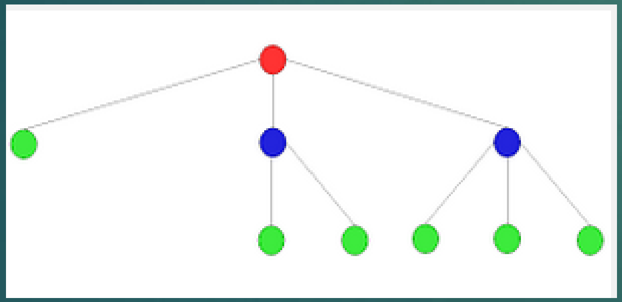
- Indique la “*ariedad*” ¿Cuáles son llenos? ¿Completos? Complete los que no lo están



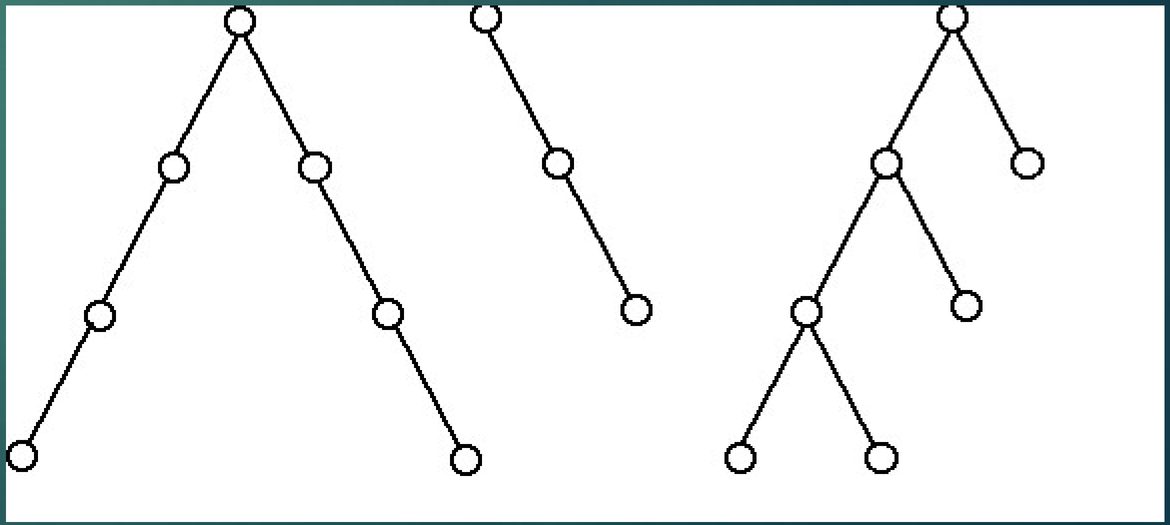
Árbol binario balanceado

- Un árbol binario está balanceado si la diferencia entre la altura de sus hijos no excede 1.

Balanceado (no lleno ni completo)



3 árboles Desbalanceados



Conteos en binarios llenos

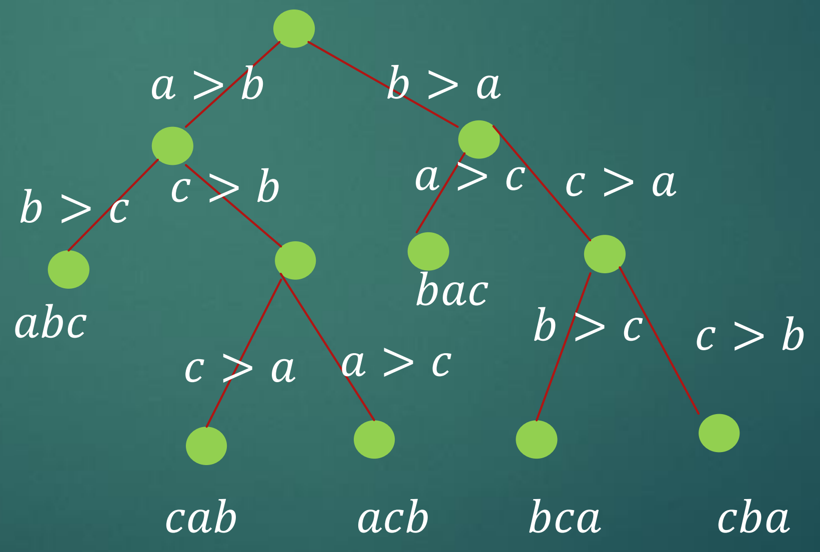
- Sea T binario no vacío y lleno con $n(T)$ nodos. Sean $i(T)$ el número de nodos interiores y $l(T)$ el número de hojas. Se cumplen:
  1. $l(T) = i(T) + 1$
  2. $n(T) = 2i(T) + 1$ o equivalentemente $i(T) = (n(T) - 1)/2$
  3. $n(T) = 2l(T) - 1$ o equivalentemente $l(T) = \frac{n(T)+1}{2}$
- Ejercicio: Probar

Contando en binarios

- Un árbol binario T tiene a los más 2^h hojas donde h es su altura
- Entonces $h \geq \ log(\ l)$ en el caso general
- Si T está lleno y es balanceado se da la igualdad:
  - Es decir, en ese caso: $h = \log(l)$ o equivalentemente $2^h = l$
- Ejercicio: Probar
- Nota: Este resultado vale para árboles m-arios: $h \geq \ log_m(l)$. La igualdad se da si el árbol está lleno y balanceado.

Ejemplo: comparaciones para ordenar n objetos

- Dada una lista de n números, cuántas comparaciones se necesitan para ordenarlos (peor caso) si sólo se permiten comparaciones entre dos números como única operación.
- Por ejemplo $S = \{a, b, c\}$.
- 3! posibles hojas



Ejemplo: continuado

- Dados n objetos se producen $l = n!$ hojas.
- La altura h de ese árbol binario es el peor caso en comparaciones
- $h \geq \log(n!) = nlog(n)$.
- Y $h - 1$ sería el mejor caso (se ahorra la última comparación). Note que está balanceado
- **Conclusión**: No se puede ordenar por comparaciones haciendo menos de $nlog(n) - 1$ de ellas. Es decir cualquier ordenamiento por comparaciones es $\Omega(nlog(n))$