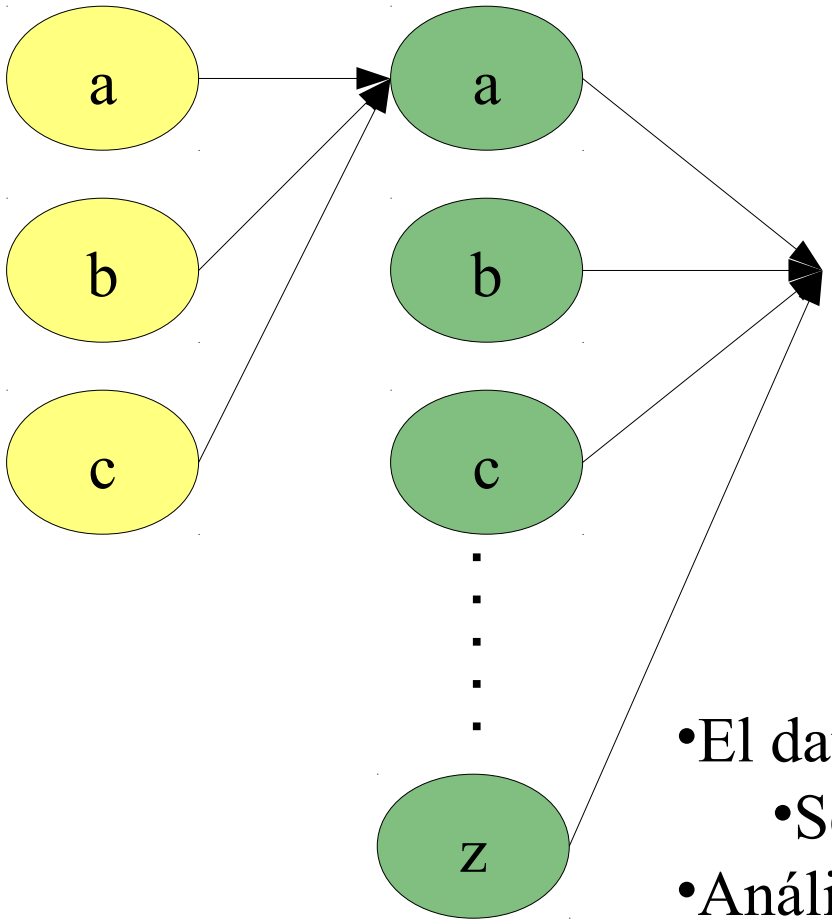
Estructuras de datos

Herramientas de programación para procesamiento de señales

Ejemplo: guía telefónica



- Largo n
- Alfabeto m (=27)
- Hipótesis n>>m
- List: O(n)
- Tree 1-level: m+n/m
- Tree 2-levels: $2*m+n/m^2$
- Tree k-levels: $k*m+n/m^k$
- k tal que $m^k=n$
- $k=\log_m(n)$

- El dato tiene estructura en su contenido
  - Secuencia de datos ordinales
- Análisis estadístico: probabilidad uniforme
- Mejora: Huffman coding
- Caso general: clustering

Índice

- Estructuras de datos abstractas
- Cola de prioridad
- Binary heap

Estructuras de datos abstractas

- Abstracción formal
  - Definición de operaciones
- Tipos:
  - Listas: ordenadas, indexadas o encadenadas (simple, doble)
  - Cola: operaciones en los extremos, FIFO, etc (stack)
  - Mapa: clave, objeto
  - Set: únicos, sin orden
- Niveles de abstracción
  - Estructura abstracta
  - Estructura concreta
  - Implementación computadora
  - Misma estructura, diferentes roles

Cola de prioridad

- Definición:
  - Cola con un valor asignado a cada elemento (prioridad)
- Operaciones:
  - insert
  - remove_top
  - change_priority
  - get_top
- Implementación:
  - Min-priority queue
  - binary heap
  - array

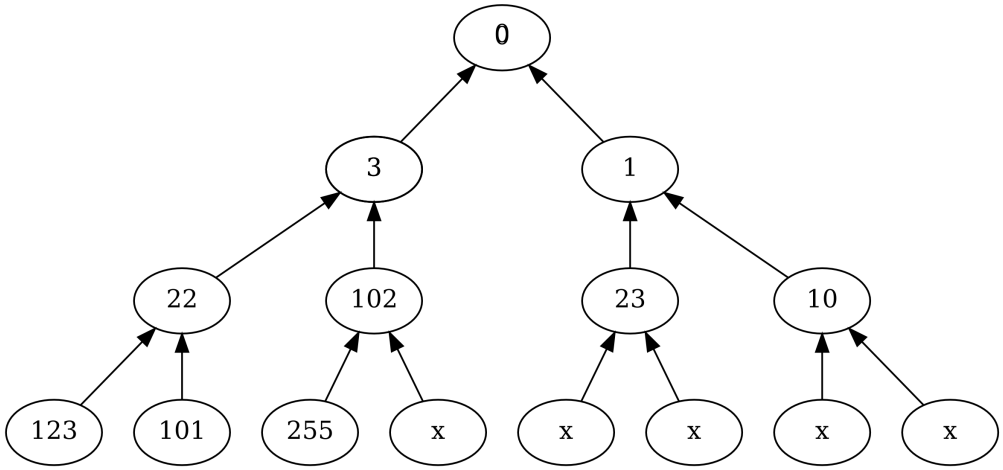
Estructuras de datos abstractas

	insert	delete	search	find max	find min	r. access	almac.
u. array	O(n)	O(n)	O(n)	O(n)	O(n)	O(1)	O(n)
u. linked list	O(1)	O(1)	O(n)	O(n)	O(n)	O(n)	O(n)
max-heap	O(log n)	O(log n)	X	O(1)	X	X	O(n)

- Hipótesis
  - Análisis de peor caso
  - Falta tiempo de construcción
  - #accessos >> construcción

Heap (1)

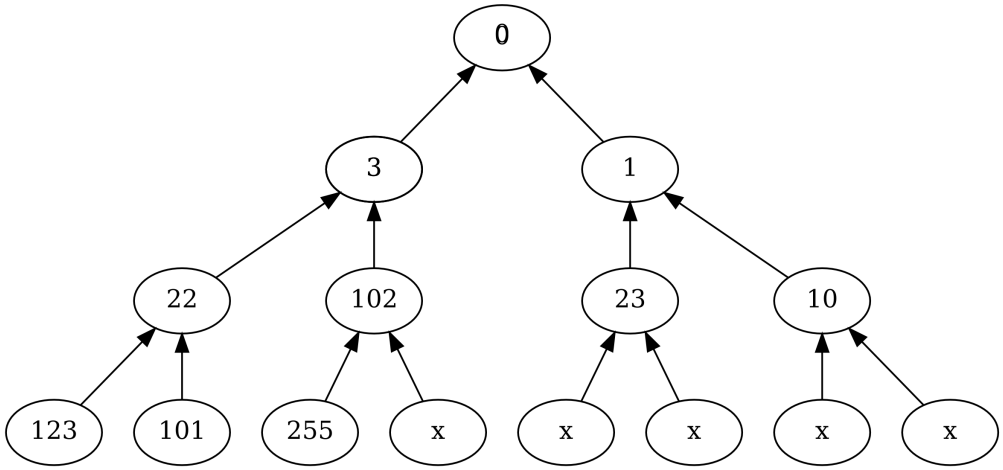
- Definición:
  - árbol (binario)
  - Shape property
  - Heap property

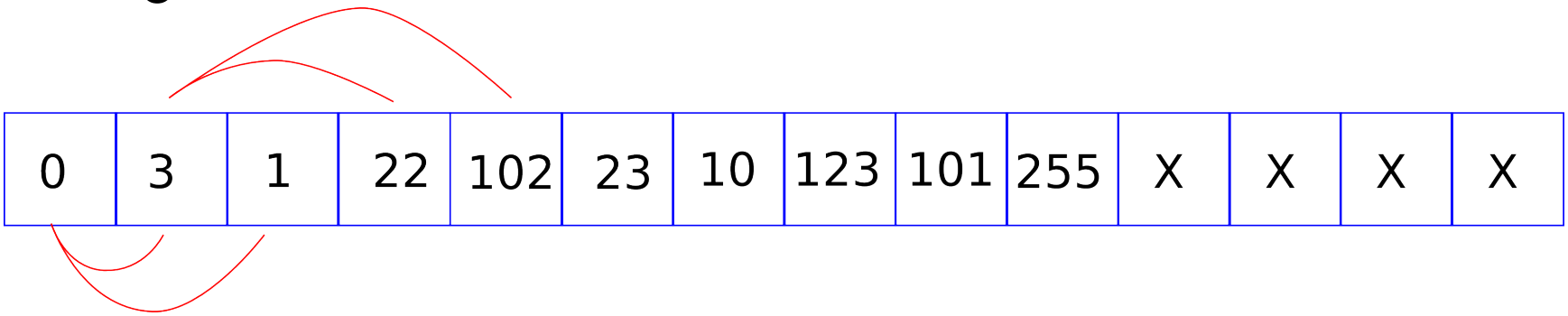


- Datos:
  - Cualquier estructura (binario, d-tree, forest)
  - Comparación: predicado binario
  - Tipos: max-min

Heap (2)

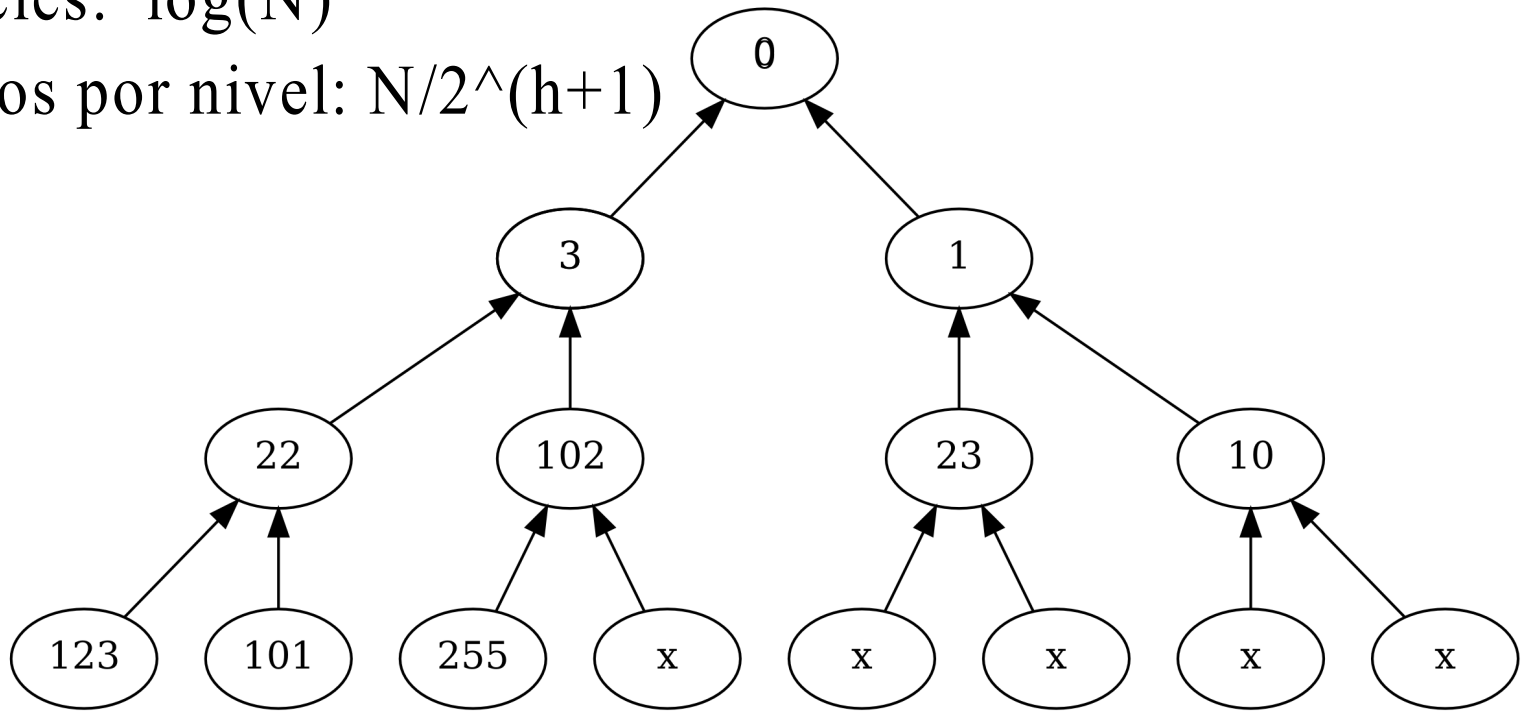
- Almacenamiento:
  - array de largo fijo
- Operaciones de árbol:
  - Parent: floor((i-1)/2)
  - Left child: 2i+1
  - Right child: 2i+2





Heap (3)

- Árbol binario balanceado
  - Hojas: N/2
  - Internos: N/2
  - Niveles: log(N)
  - Nodos por nivel: N/2^(h+1)

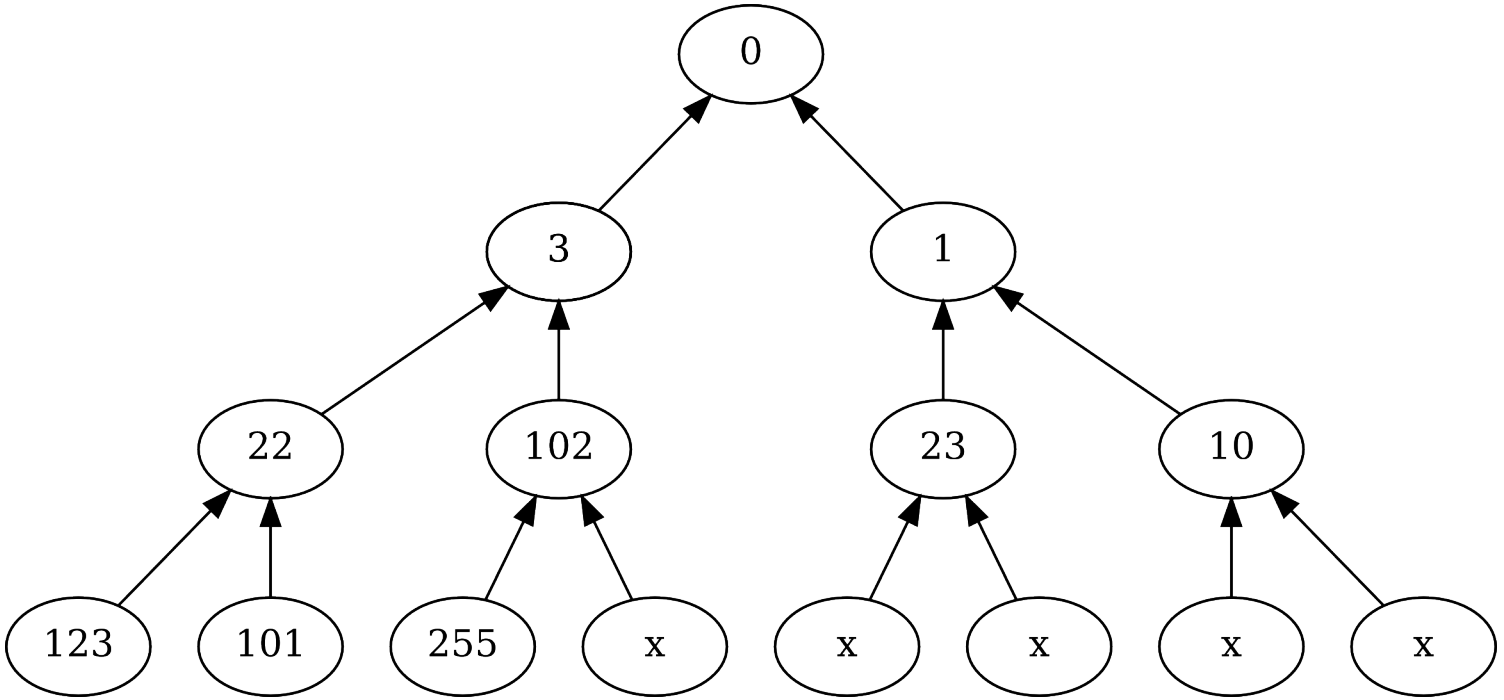


Heap (4)

- Operaciones:
  - publicas:
    - isEmpty()
    - insert()
    - removeTop()
    - changePriority()
    - isFull()
  - internas:
    - moveUp()
    - moveDown()
    - left(), right(), parent()

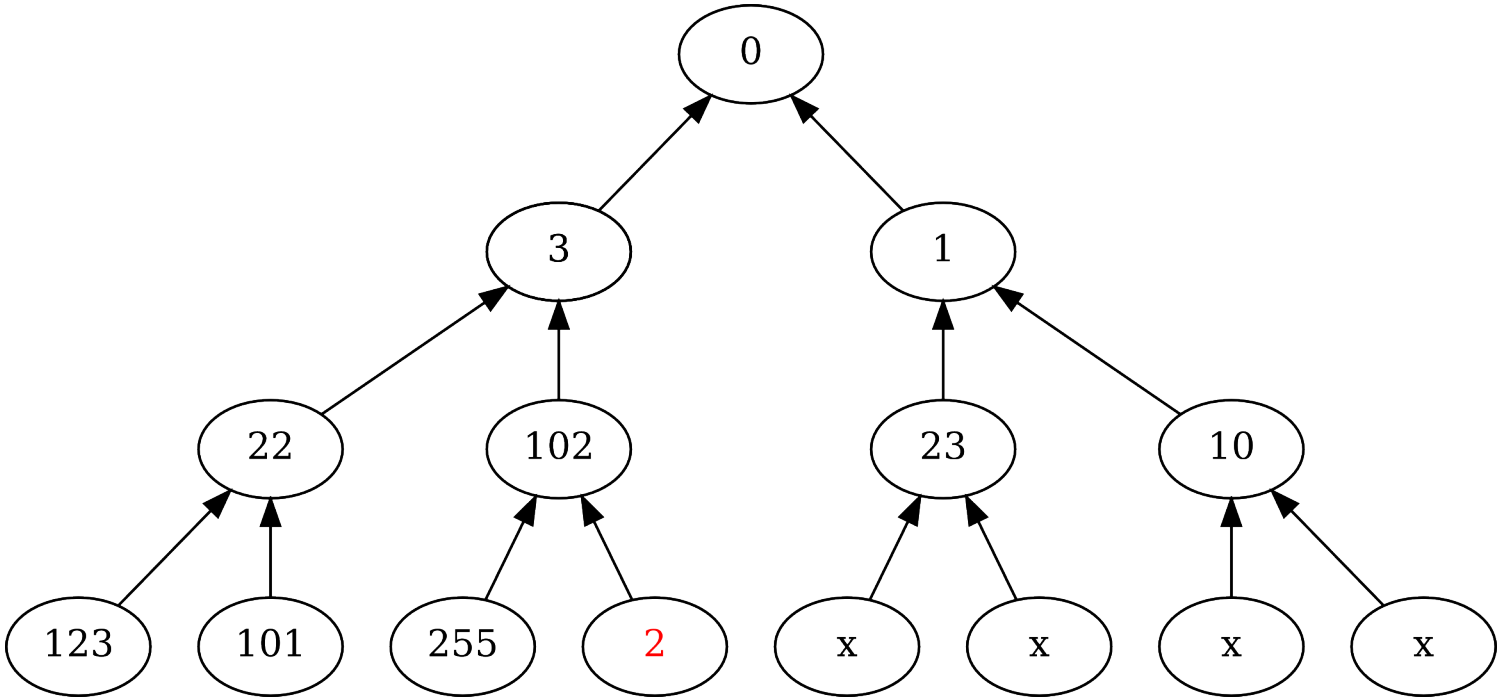
Heap (5)

- Análisis de una operación: moveUp()



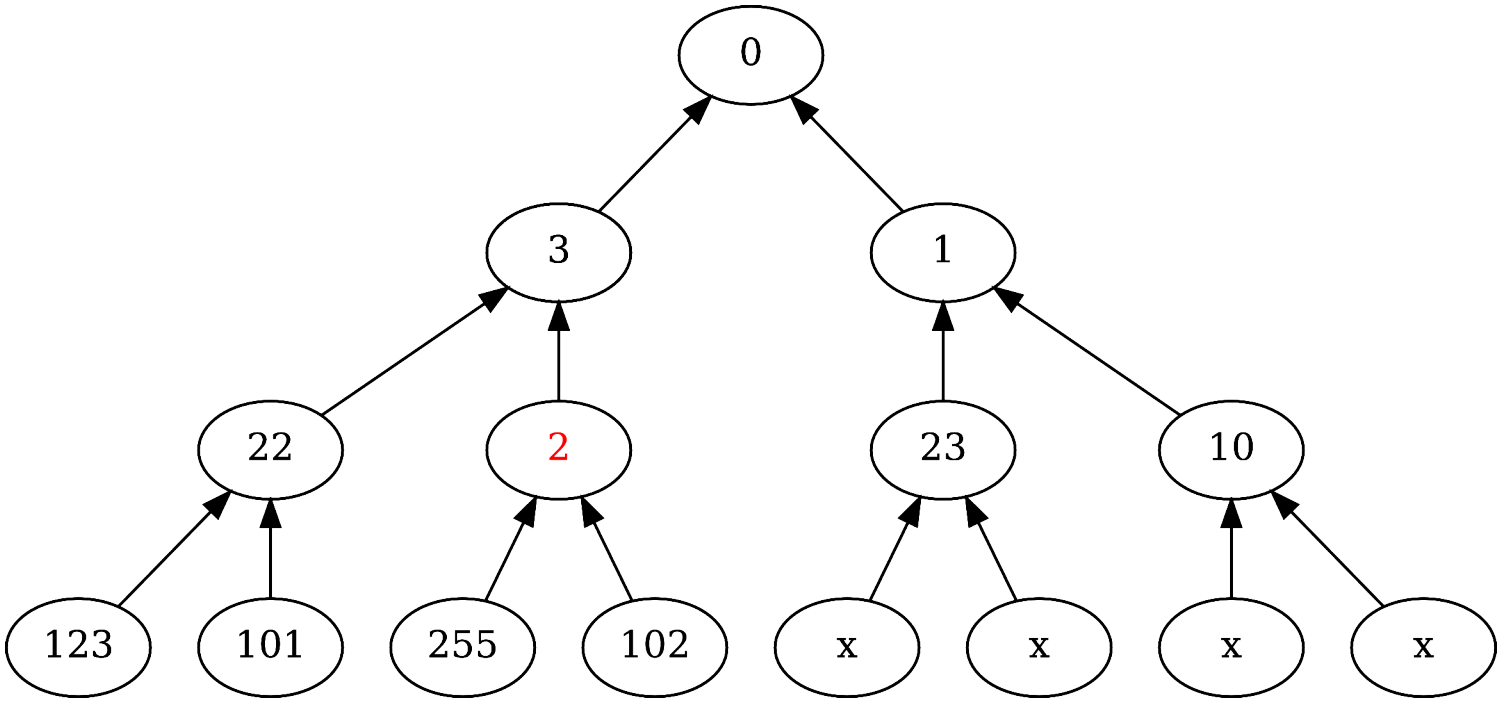
Heap (6)

- Análisis de una operación: moveUp()



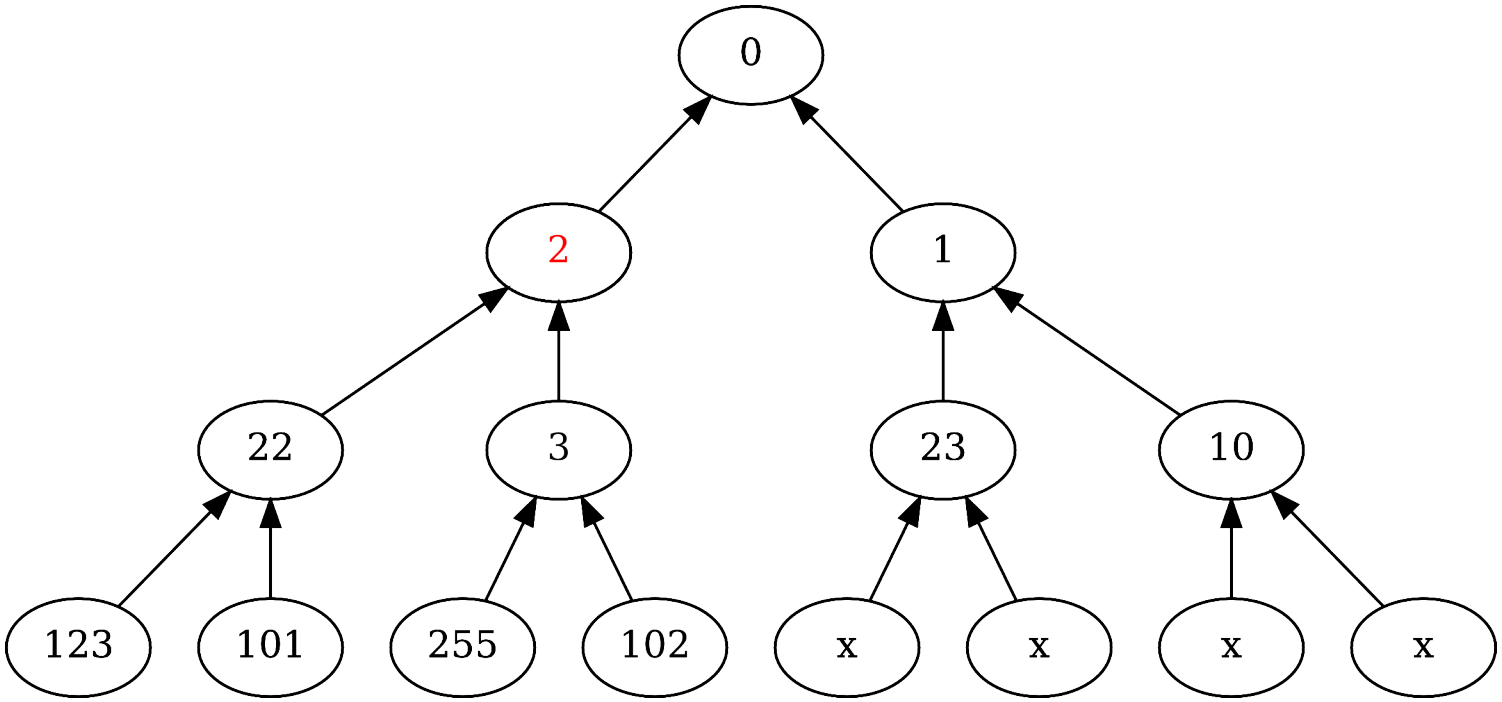
Heap (7)

- Análisis de una operación: moveUp()



Heap (8)

- Análisis de una operación: moveUp()



Heap (9)

- Comentarios
  - Costo: peor caso
  - Costo promedio: 2.6 → O(1)
  - Construcción: O(n)

	insert	delete	search	find max	r. access	almac.
u. array	O(n/2)	O(n/2)	O(n/2)	O(n/2)	O(1)	O(n)
u. linked list	O(1)	O(1)	O(n/2)	O(n/2)	O(n/2)	O(n)
heap	O(1)	O(1)	X	O(1)	X	O(n)

Heap (9)

- Implementación C

ITEM

`int label`
`float value`
`int idx`

HEAP

`int n`
`int n_max`
`ITEM** data`

- Notas
  - Unidireccional: un ITEM no sabe donde está ubicado
  - idx: auxiliar para búsqueda inversa

Bibliotecas

- C++: Standard Template Library
  - Completa
  - Genérica
- Boost