Digital Search Trees and Tries

digital search tree

↓ 降低比較次數 (縮短執行時間)

trie

↓ 減少索引使用空間 (黃色節點)

compressed trie

↓ 減少資料使用空間 (黃色節點儲存資料)

patricia

Digital Search Trees & Binary Tries

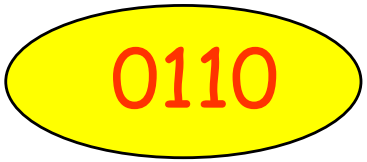
- 從二元表示資料的角度操作資料
  - 先前探討的資料結構, e.g., b-tree, 則將keys視為有某種物理意義上的數值，得比較大小，或定義前與後 (that is, total ordering)
- Keys are binary bit strings.
  - Fixed length – 0110, 0010, 1010, 1011.
  - Variable length – 01, 00, 101, 1011.
- Applications
  - Networks, e.g., hardware IP routing, packet classification
  - Databases indexing with Patricia, e.g., Bitcoin
  - …

Digital Search Tree

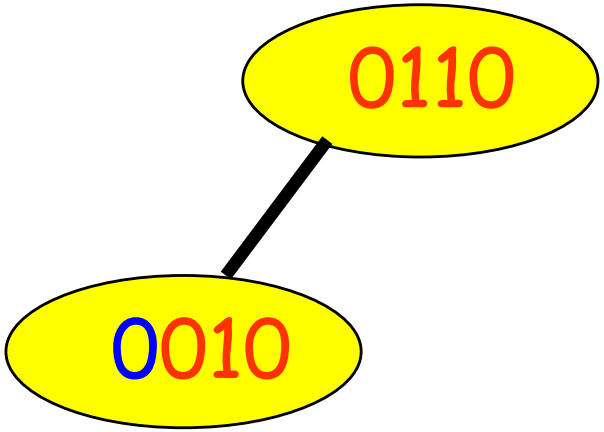
- Assume fixed number of bits. //課程僅討論這種情境
- Not empty =>
  - Root contains one data pair (any pair).
  - All remaining pairs whose key begins with a 0 are in the left subtree.
  - All remaining pairs whose key begins with a 1 are in the right subtree.
  - Left and right subtrees are digital subtrees on remaining bits.

Example

- Start with an empty digital search tree and insert a pair whose key is 0110.

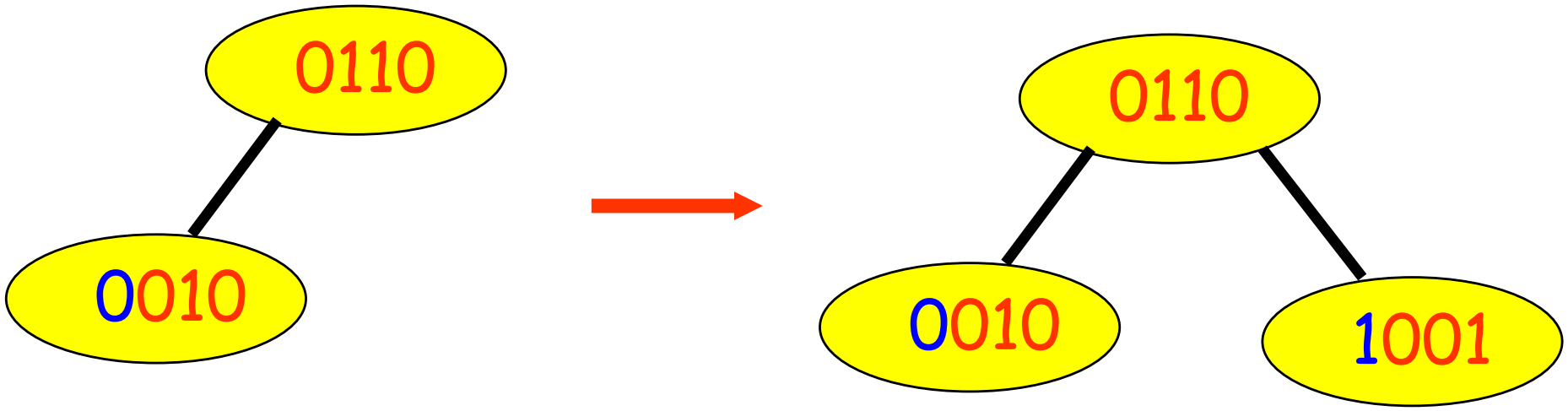


- Now, insert a pair whose key is 0010.



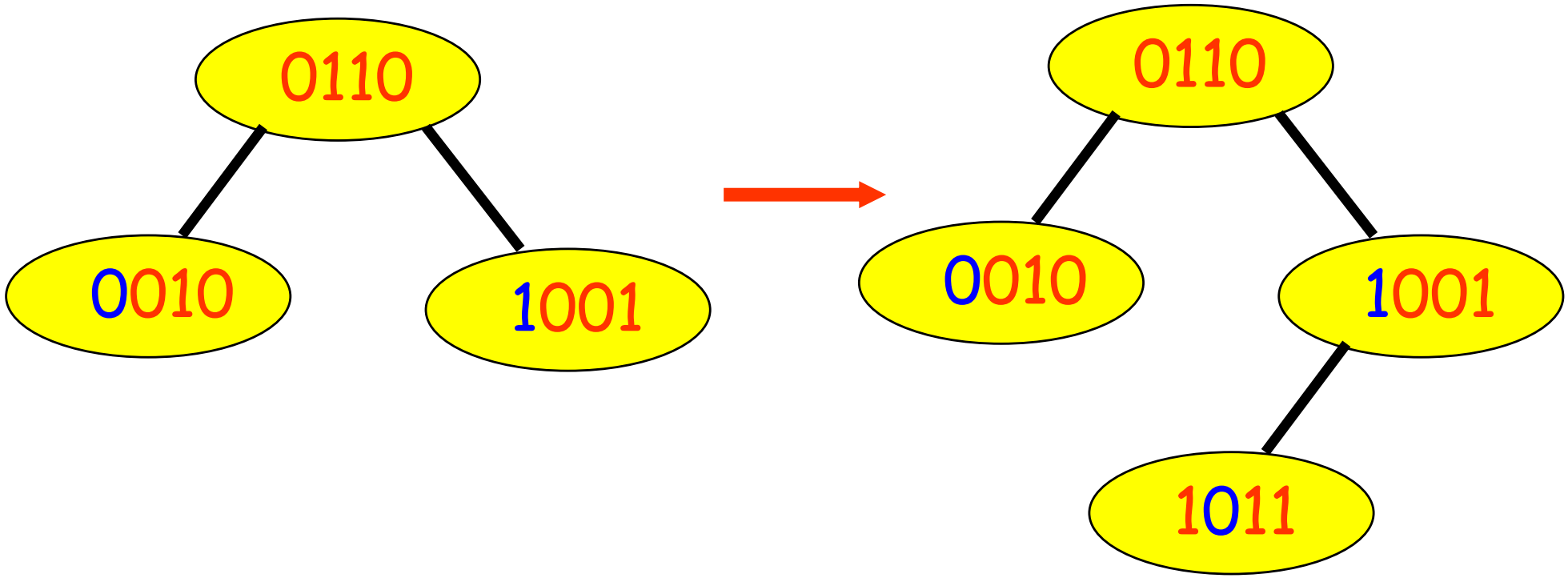
Example

- Now, insert a pair whose key is 1001.



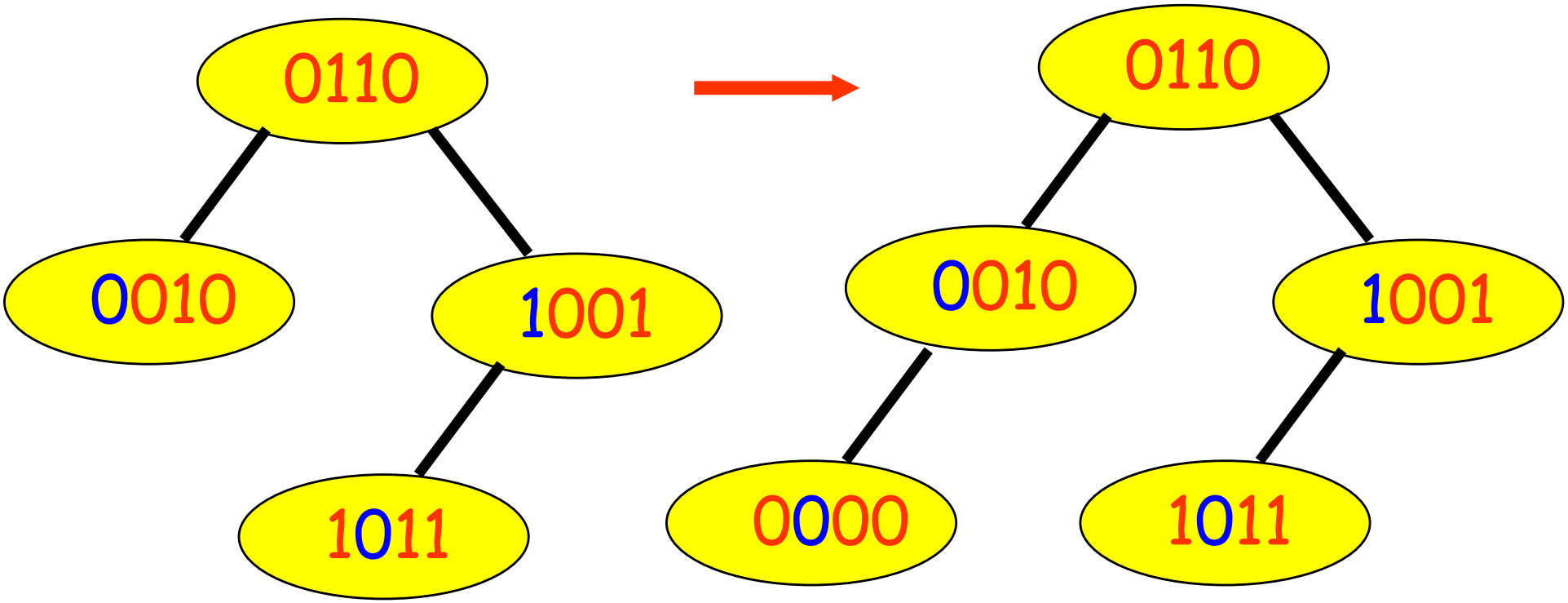
Example

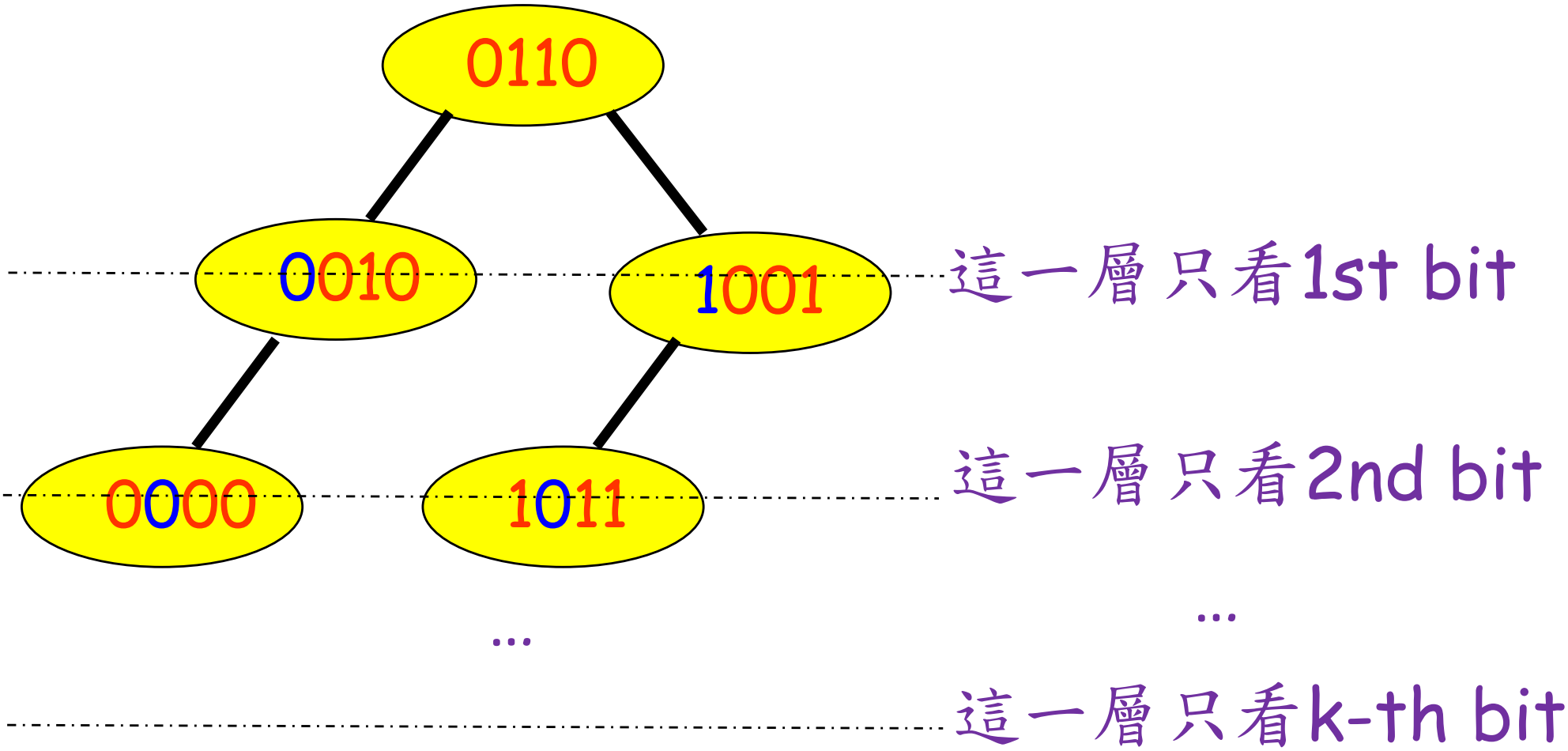
- Now, insert a pair whose key is 1011.



Example

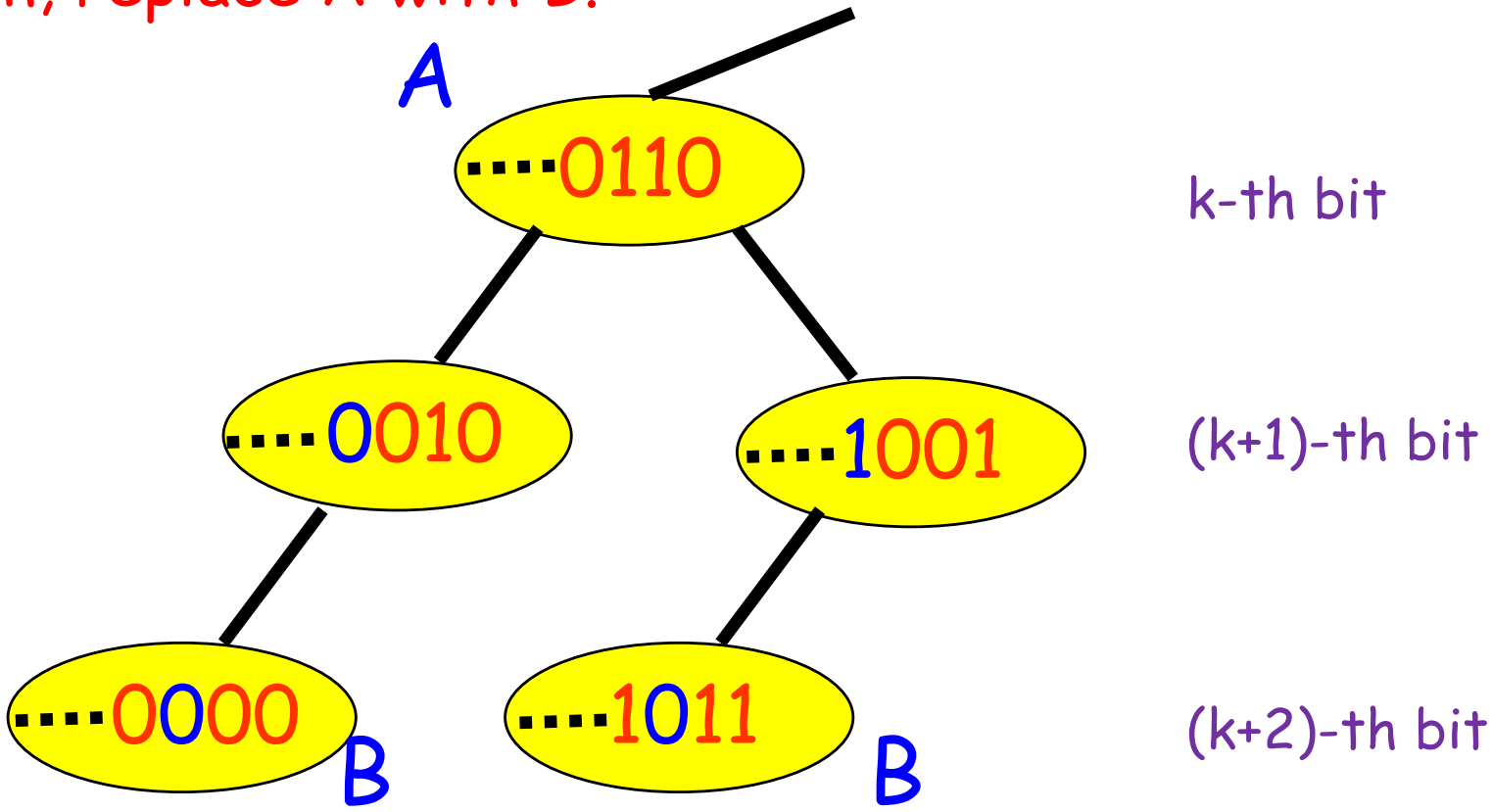
- Now, insert a pair whose key is 0000.




0110
0010
1001
0000
1011
...
這一層只看1st bit
這一層只看2nd bit
...
這一層只看k-th bit

Search & Insert: Done! Delete?

Say to delete node A. Find any leaf, B, in the subtree rooted at A. Then, replace A with B.



這兩個B的任一個都可以替代A

因為B與A一定有相同的k-bit common prefix

Some Discussions

- Complexity of each operation is O(#bits in a key).
- #key comparisons = O(height).
  - 從root開始，途中經過的節點都需要比較節點內的數值與被搜尋的目標數值
- Expensive when keys are very long.
  - If keys are computed by SHA-1 hash function with 160 bits, then…

Binary Trie

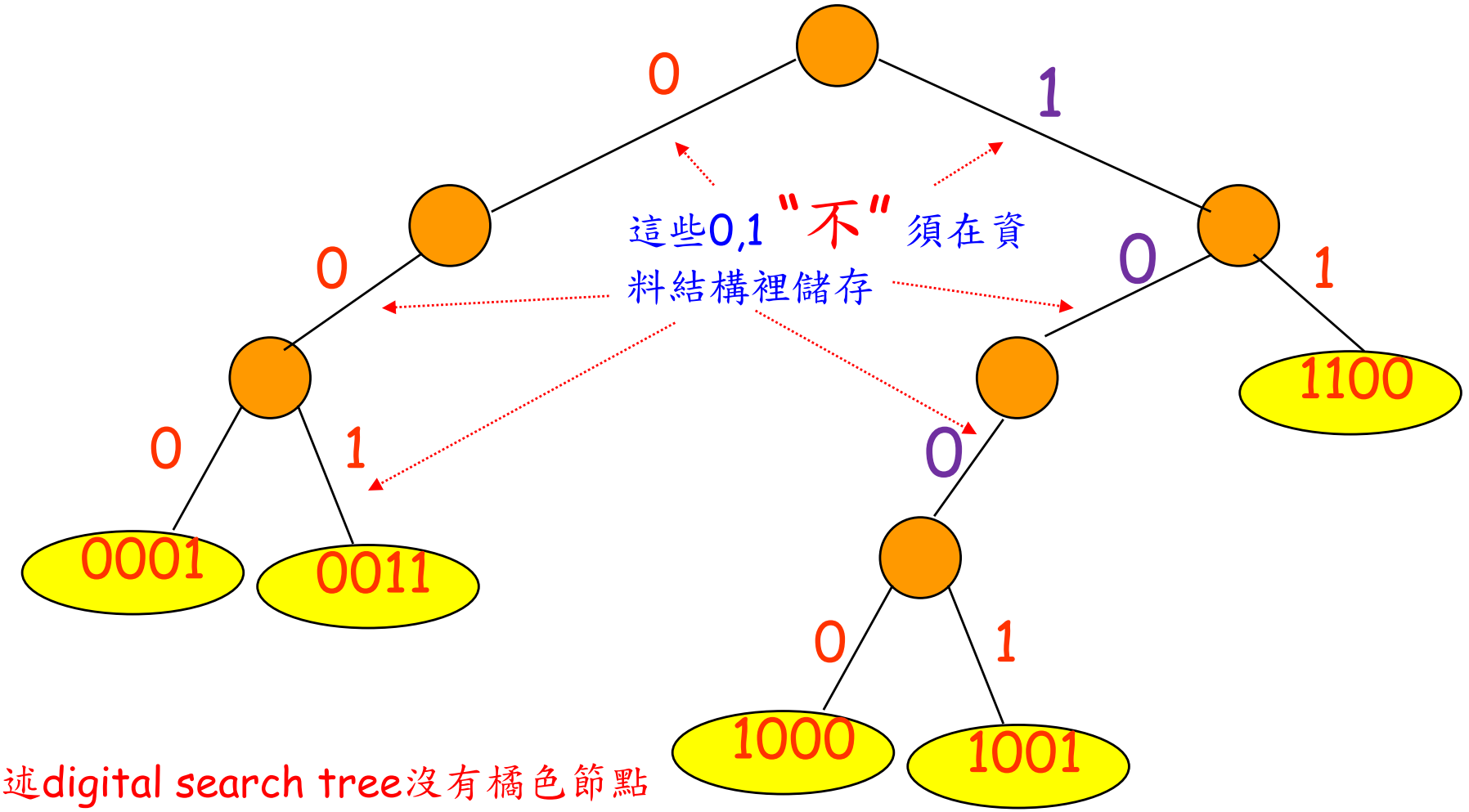
- Information Re**trie**val.
  - 關於命名，可以不需要有任何與技術相關的理由
  - e.g., NTHU's Tornado (a P2P technology)
- At most one key comparison per operation.
- Fixed length keys. //我們課程只討論這個
  - Branch nodes.
    - Left and right child pointers.
    - No data field(s).
  - Element nodes.
    - No child pointers.
    - Data field to hold dictionary data pair.

Example

Common prefix of {1000,1001} = 100

Common prefix of {1000,1001,1100}=1

Essentially, common prefix strings are represented as a trie

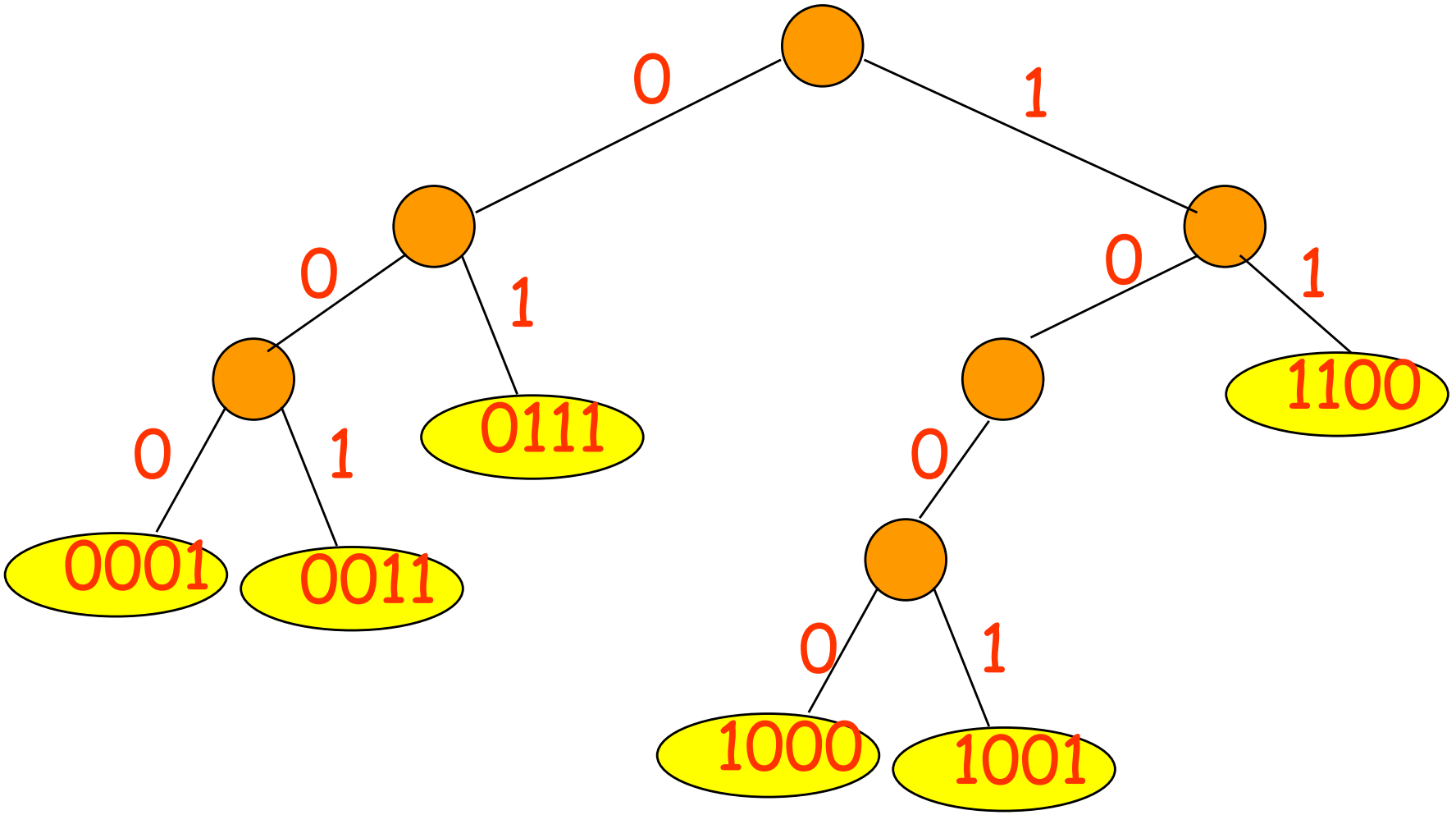


前述digital search tree沒有橘色節點

At most one key comparison for a search //走到某個黃色節點後，還需要好好比較一下keys值

Bits on edges guide the direction to traverse top-down the tree, and the traversal does not introduce comparisons.

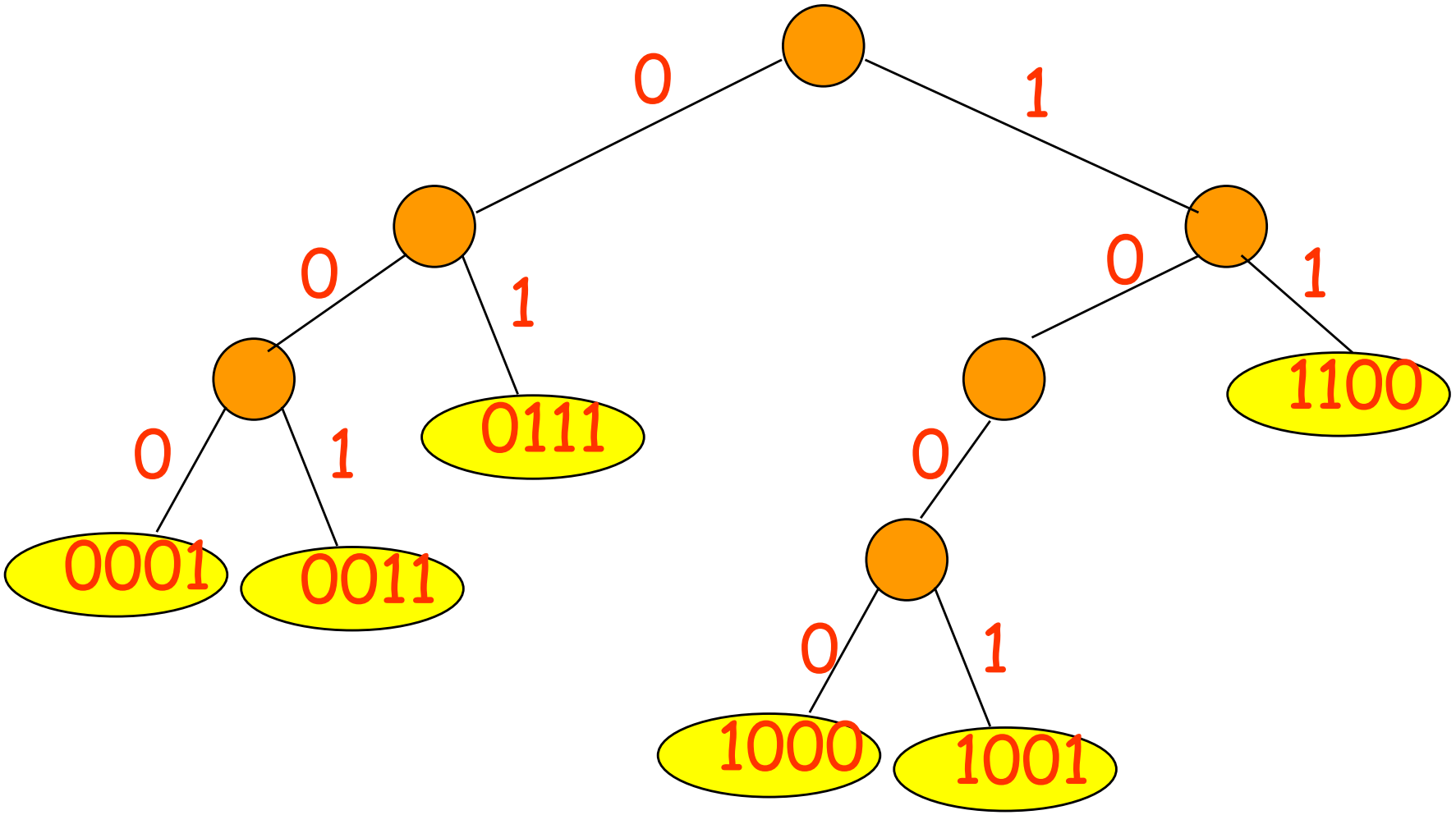
Insert



Insert 0111.

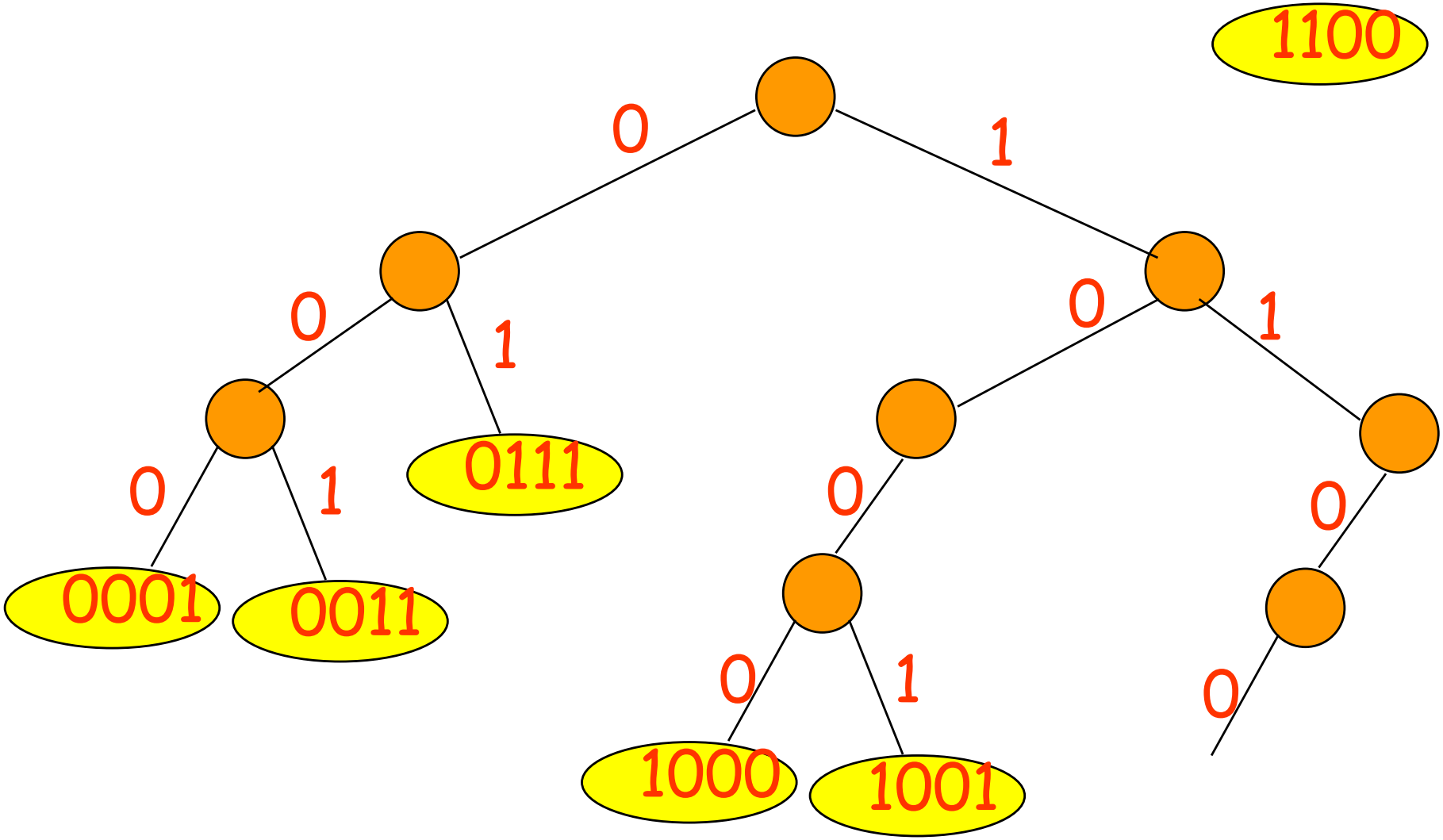
Zero compares.

Insert



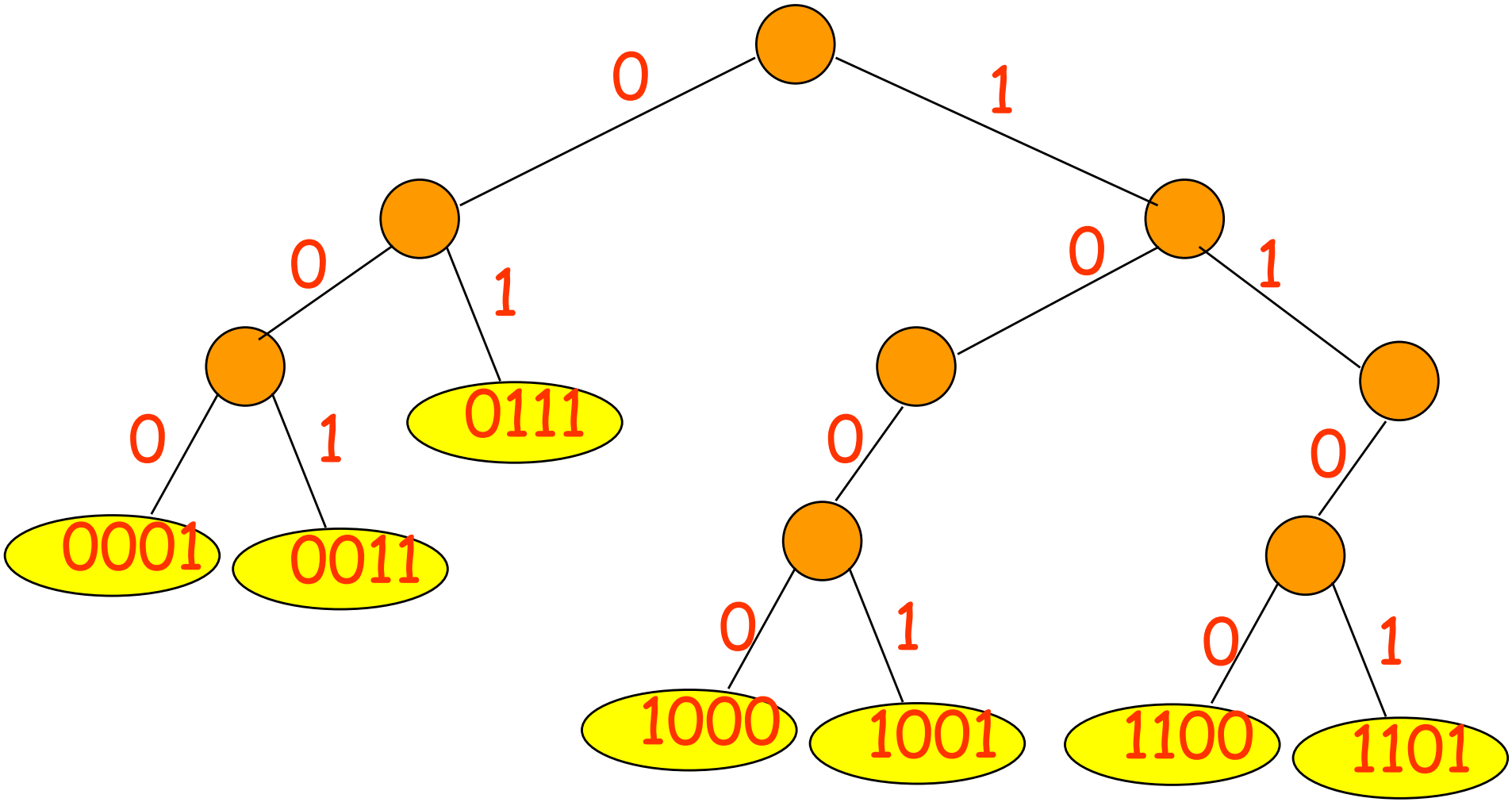
Insert 1101.

Insert



Insert 1101.

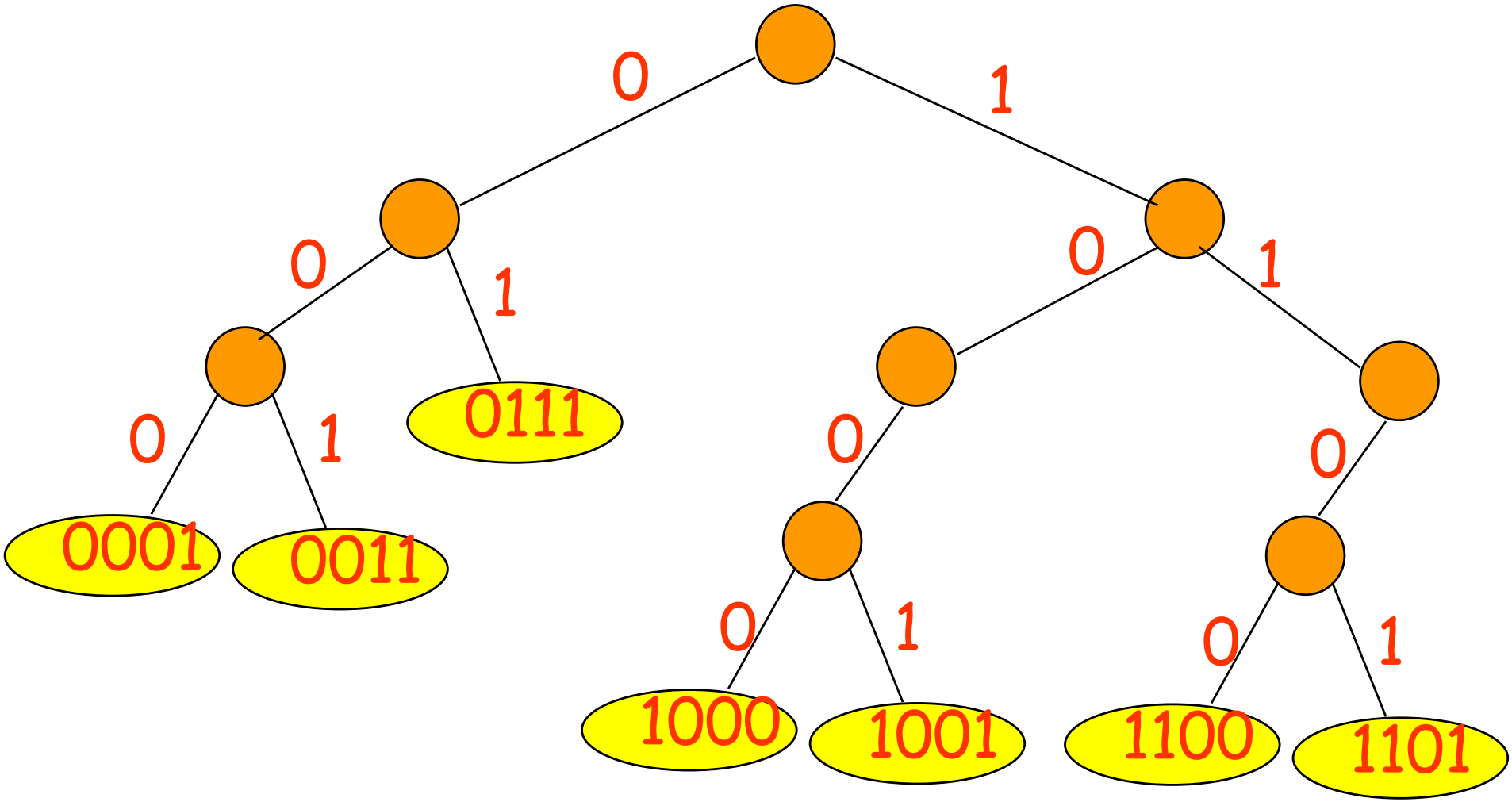
Insert



Insert 1101.

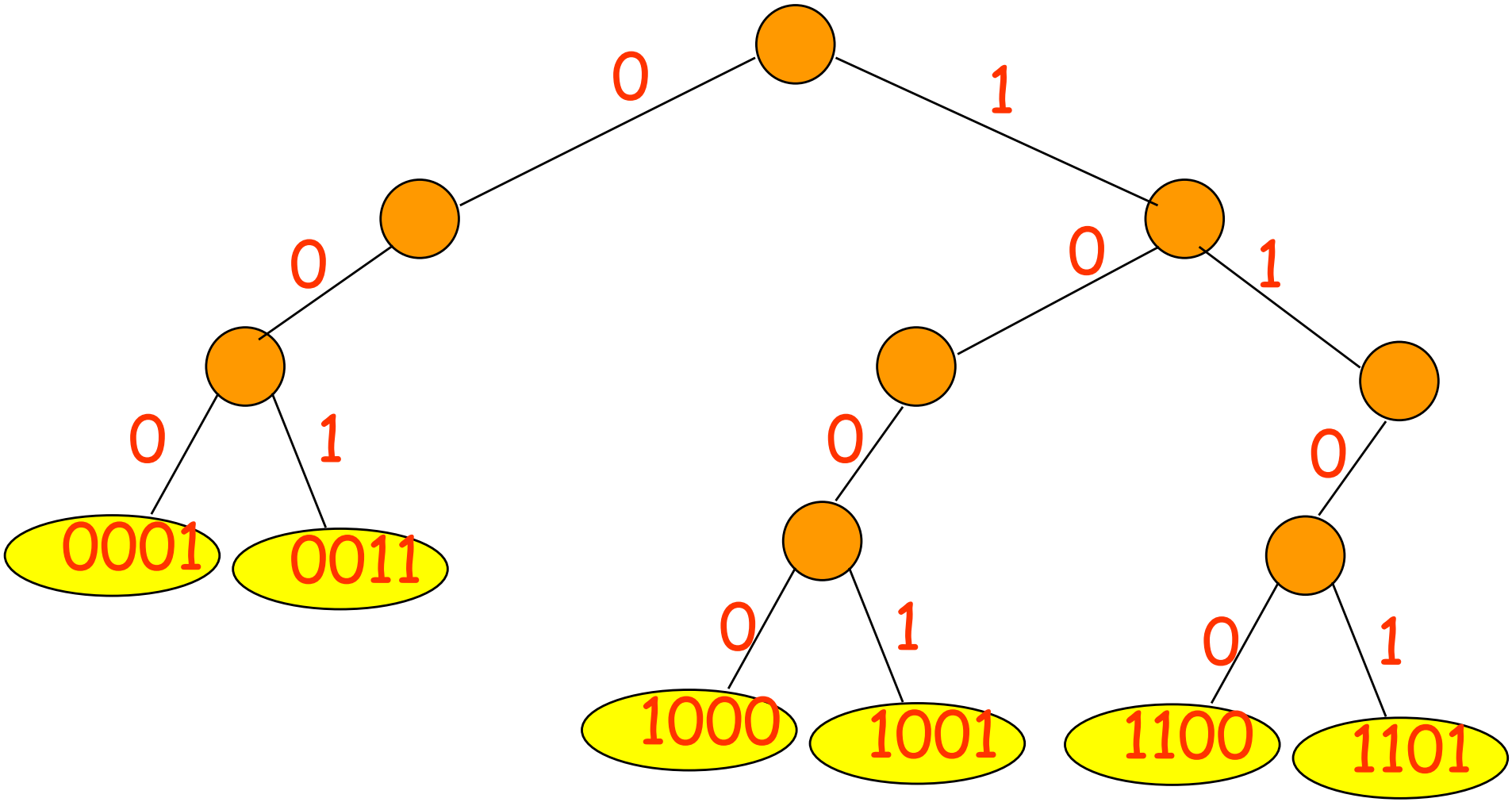
One compare.

Delete



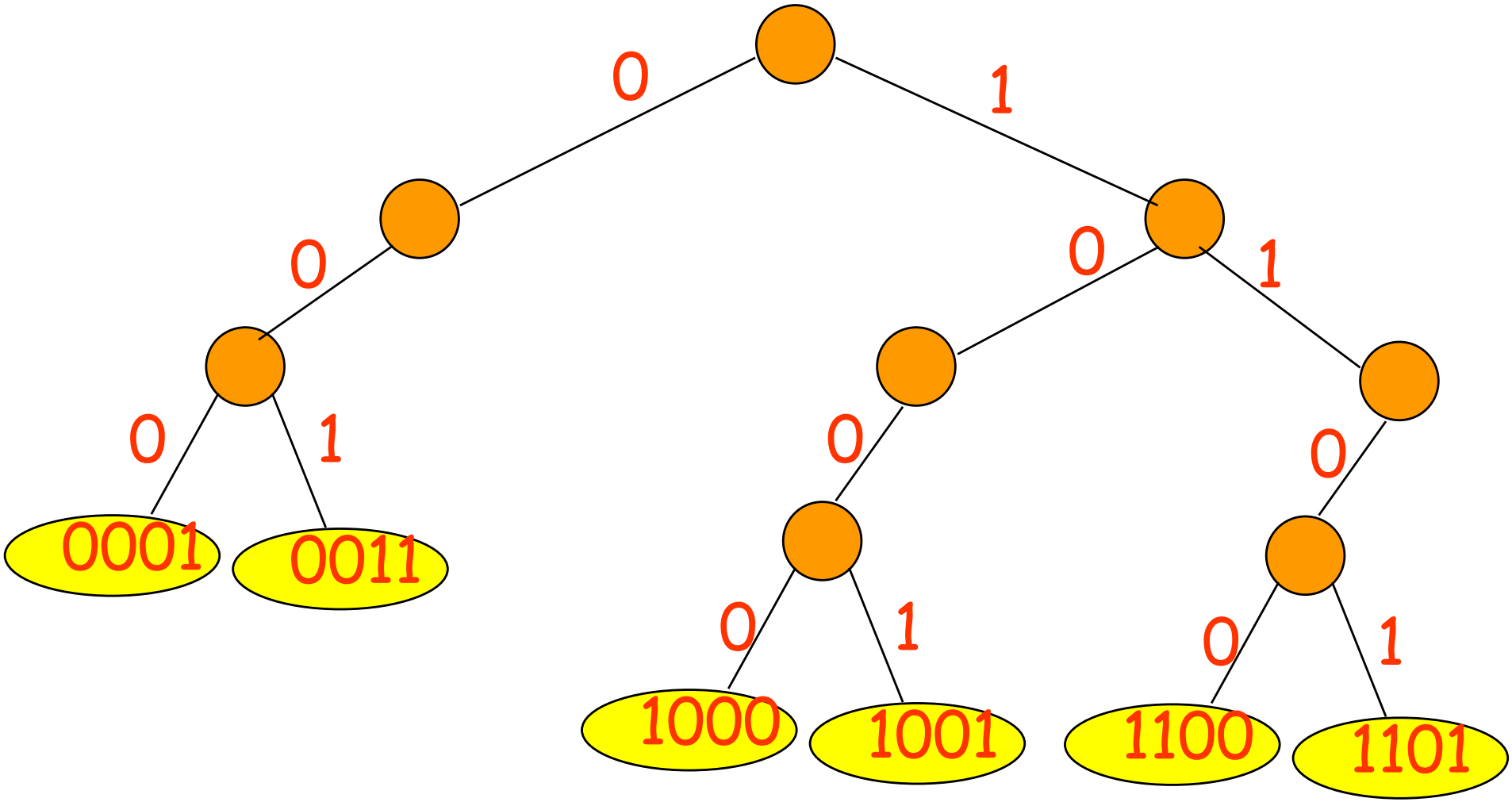
Delete 0111.

Delete



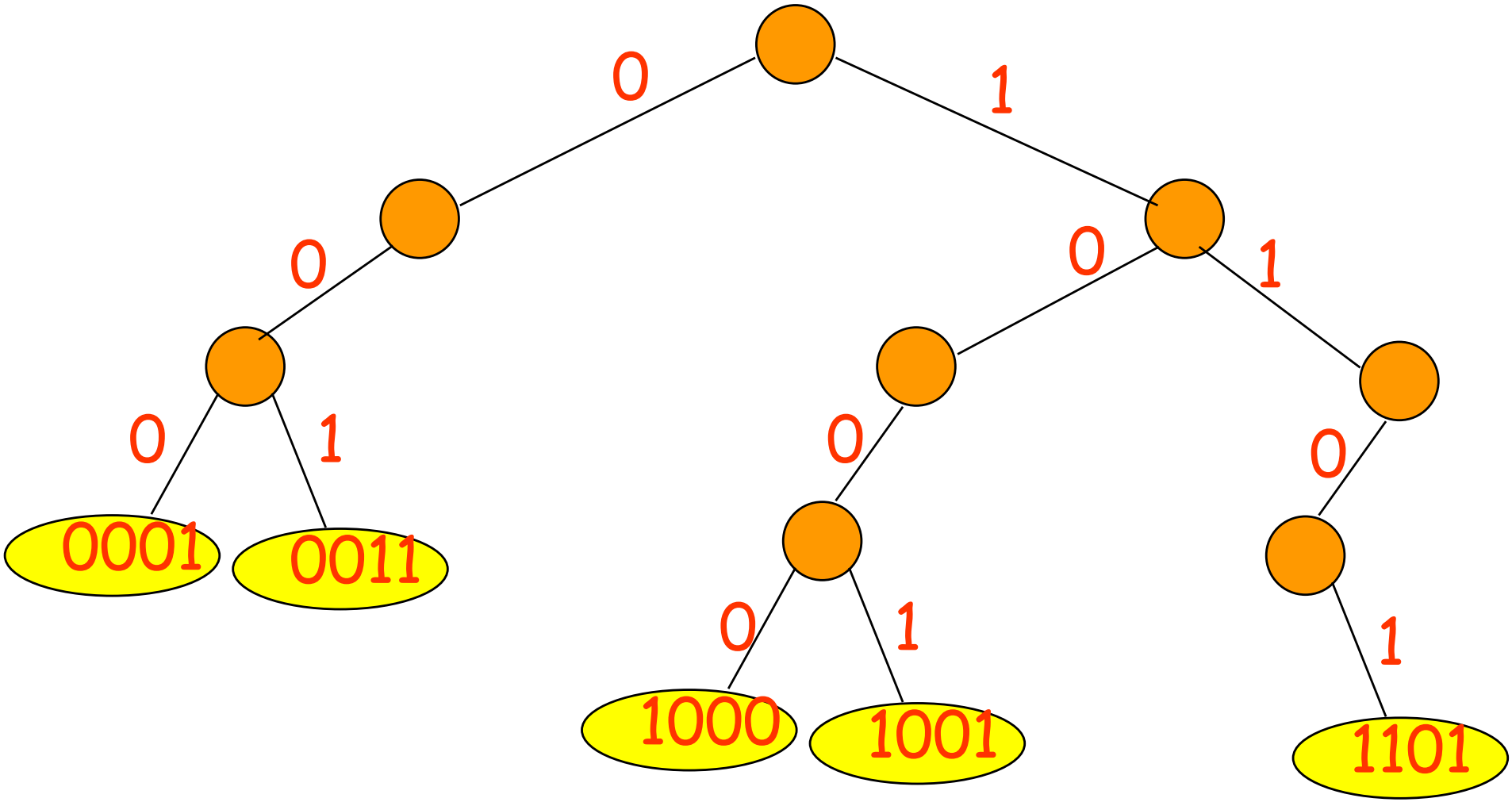
Delete 0111.

Delete



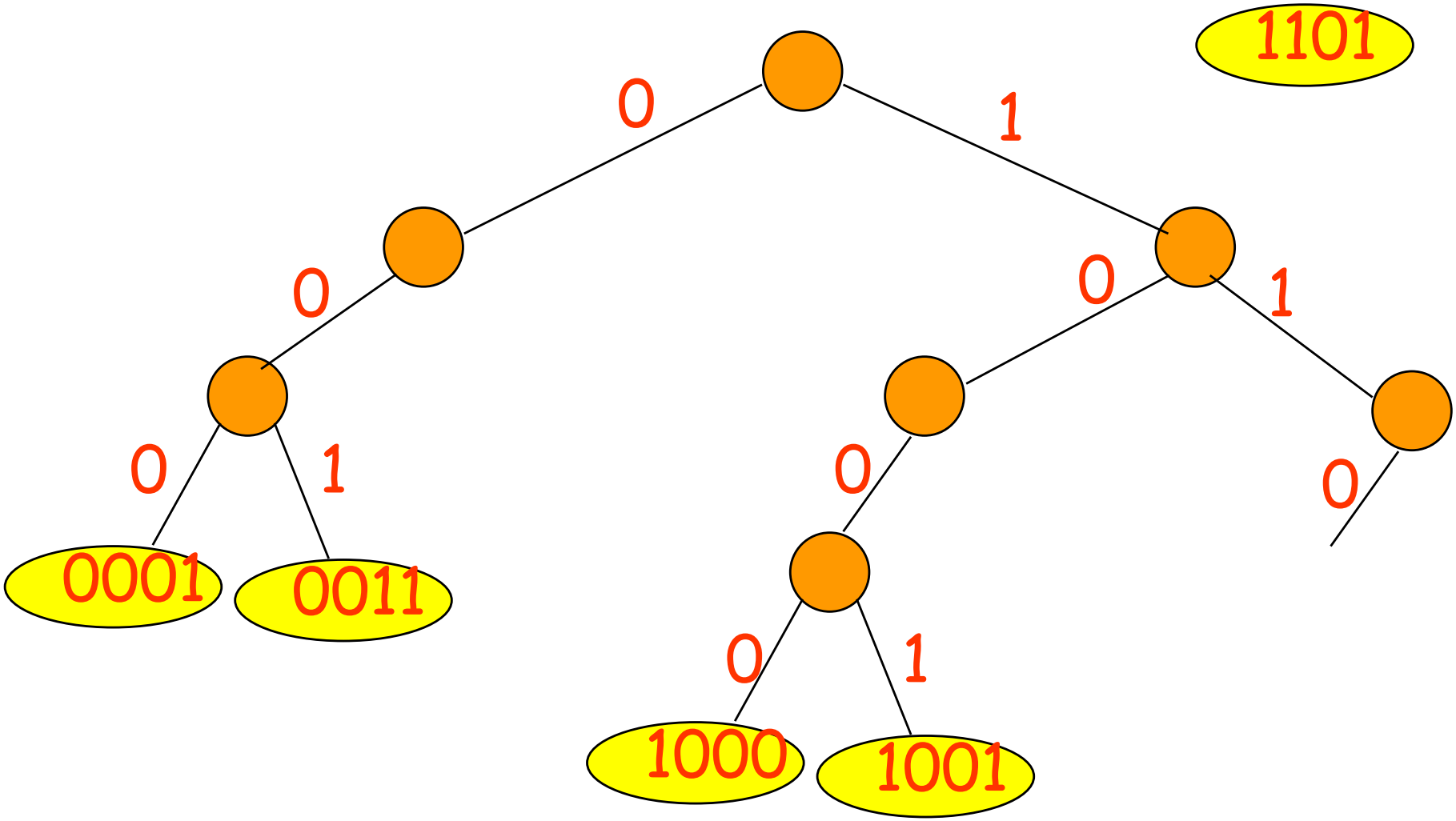
Delete 1100.

Delete



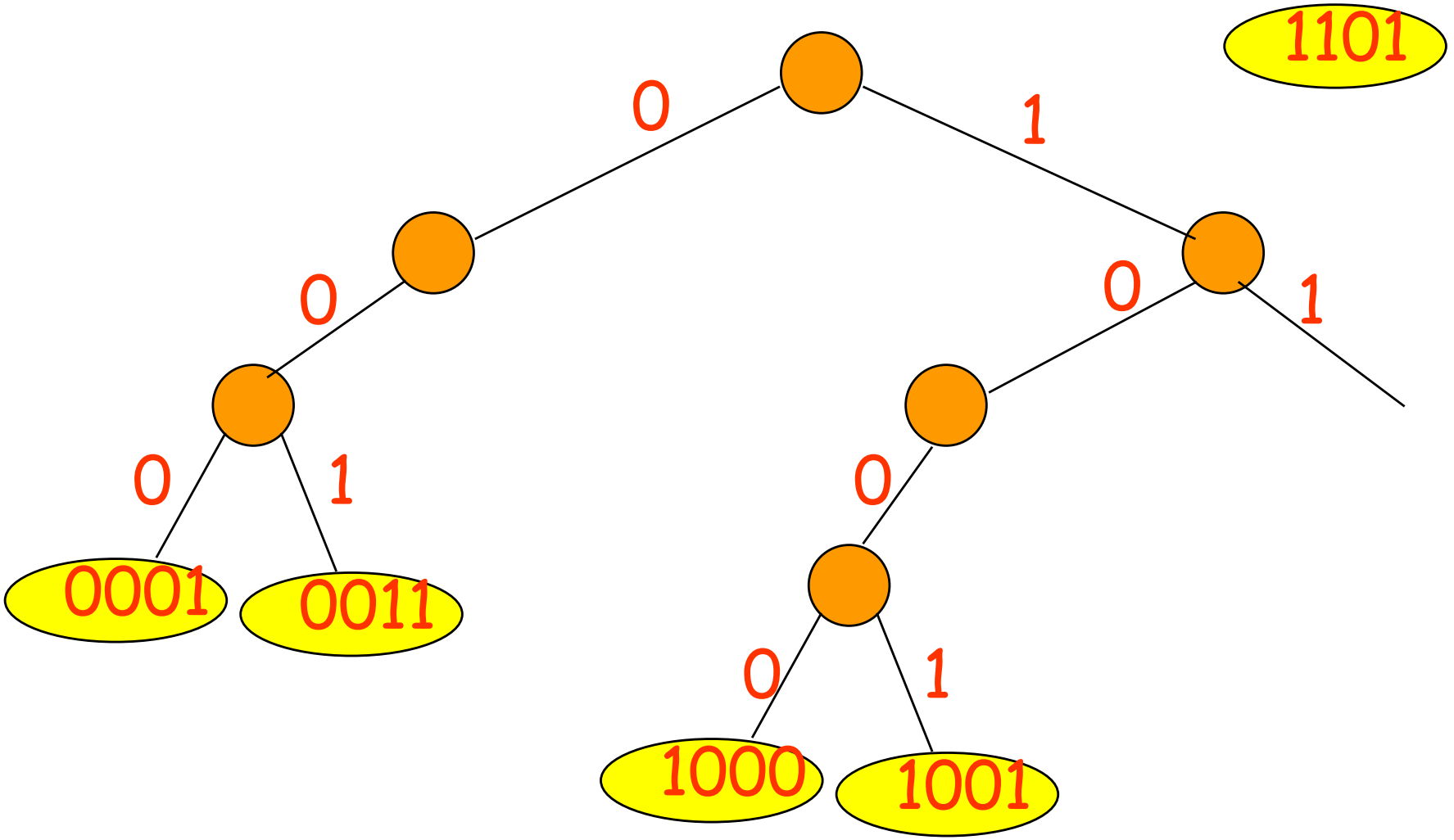
Delete 1100.

Delete



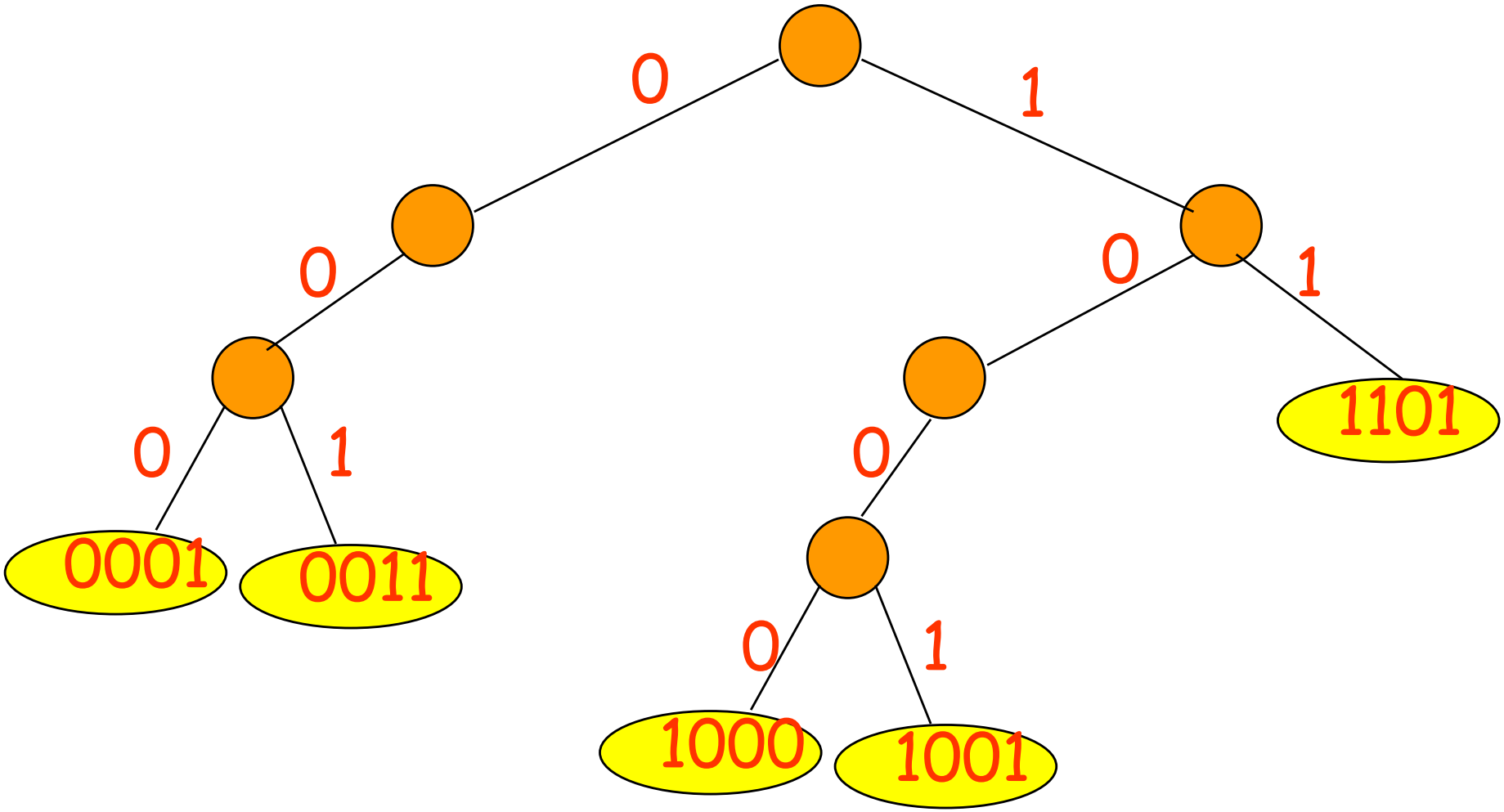
Delete 1100.

Delete



Delete 1100.

Delete



Delete 1100.

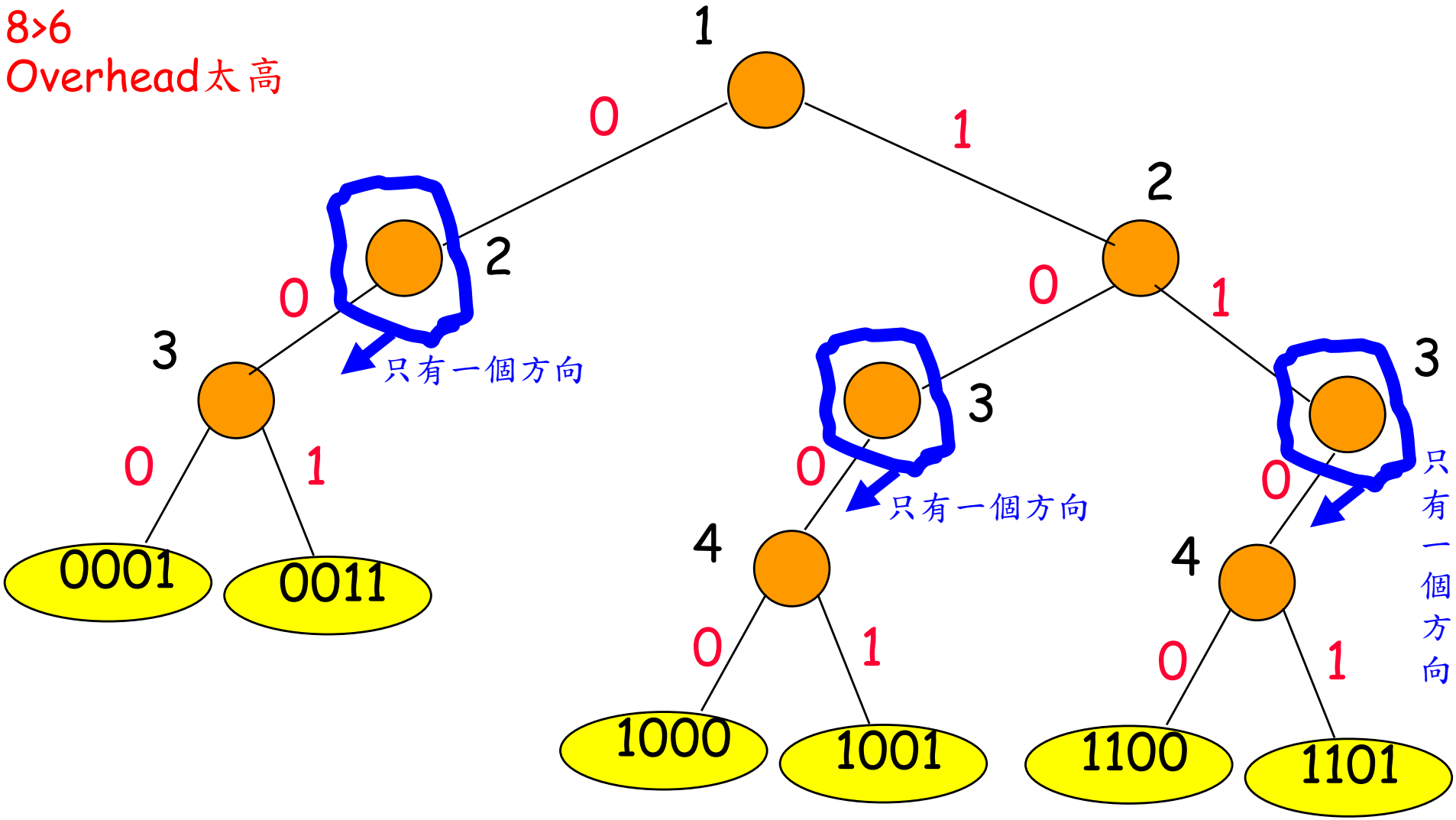
One compare.

Compressed Binary Tries

- No branch node whose degree is 1.
- Add a bit# field to each branch node.
- bit# tells you which bit of the key to use to decide whether to move to the left or right subtrie.

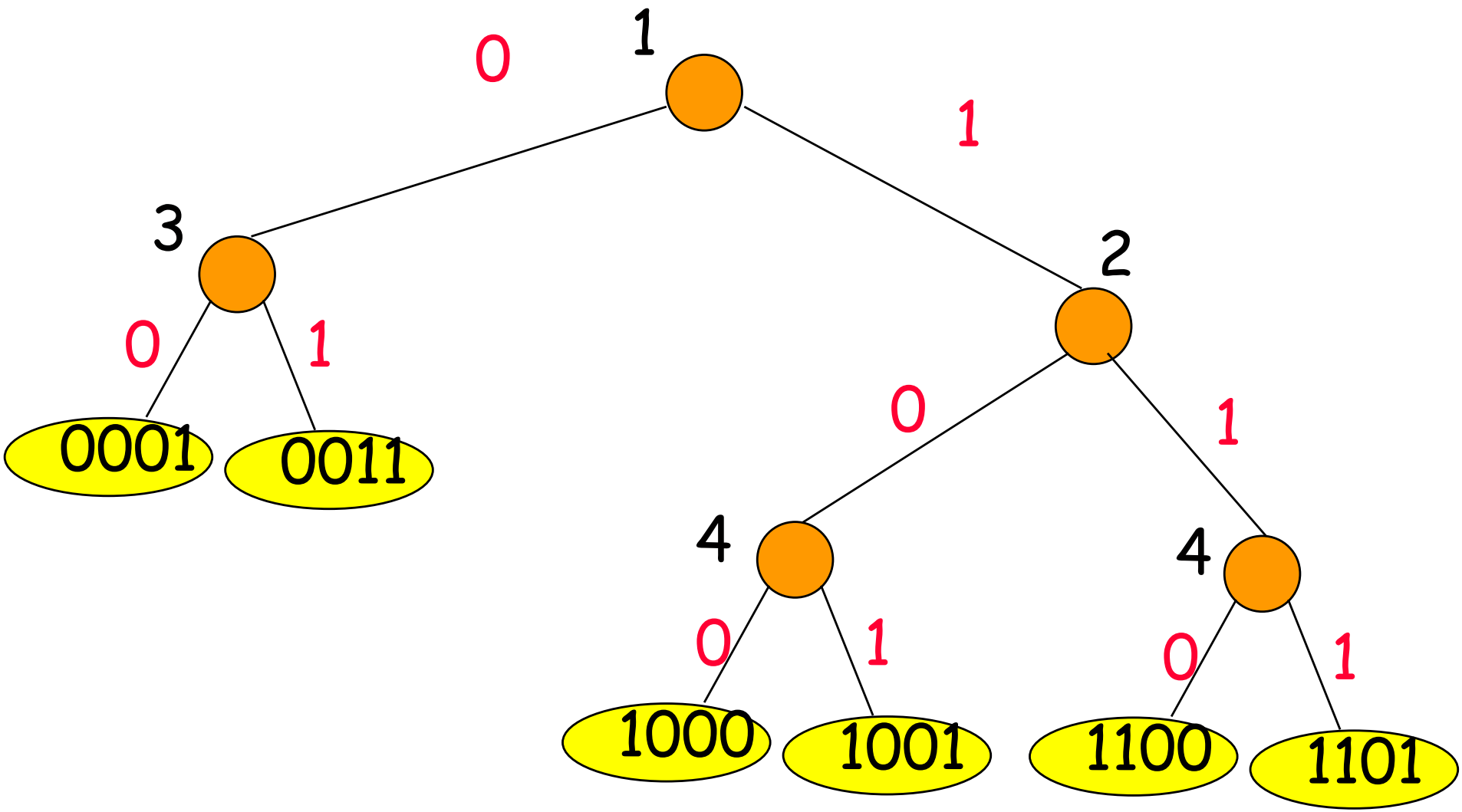
Binary Trie

- 6筆資料
- 8個內部橘色節點
- 8>6
- Overhead太高



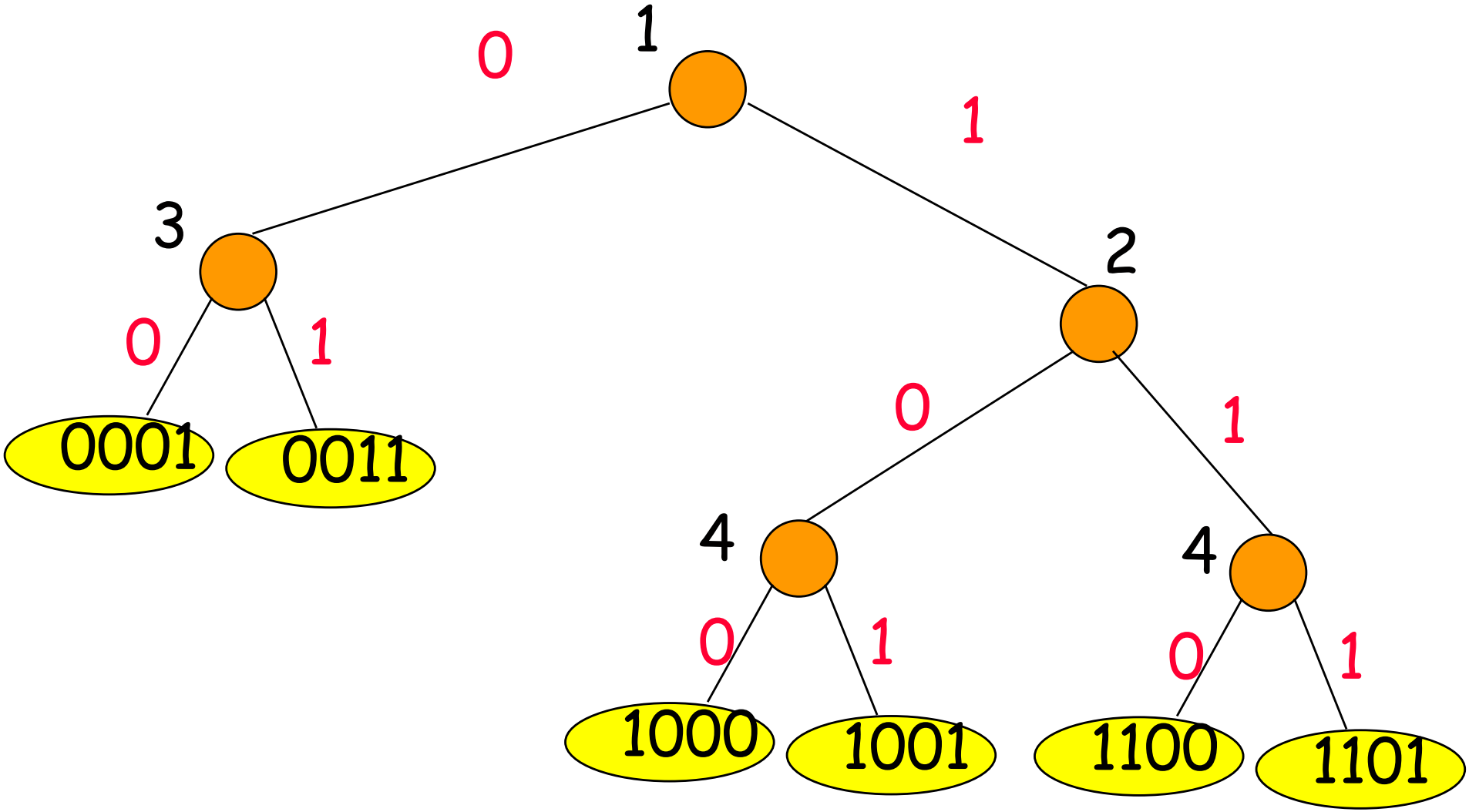
bit# field shown in black outside branch node

Compressed Binary Trie



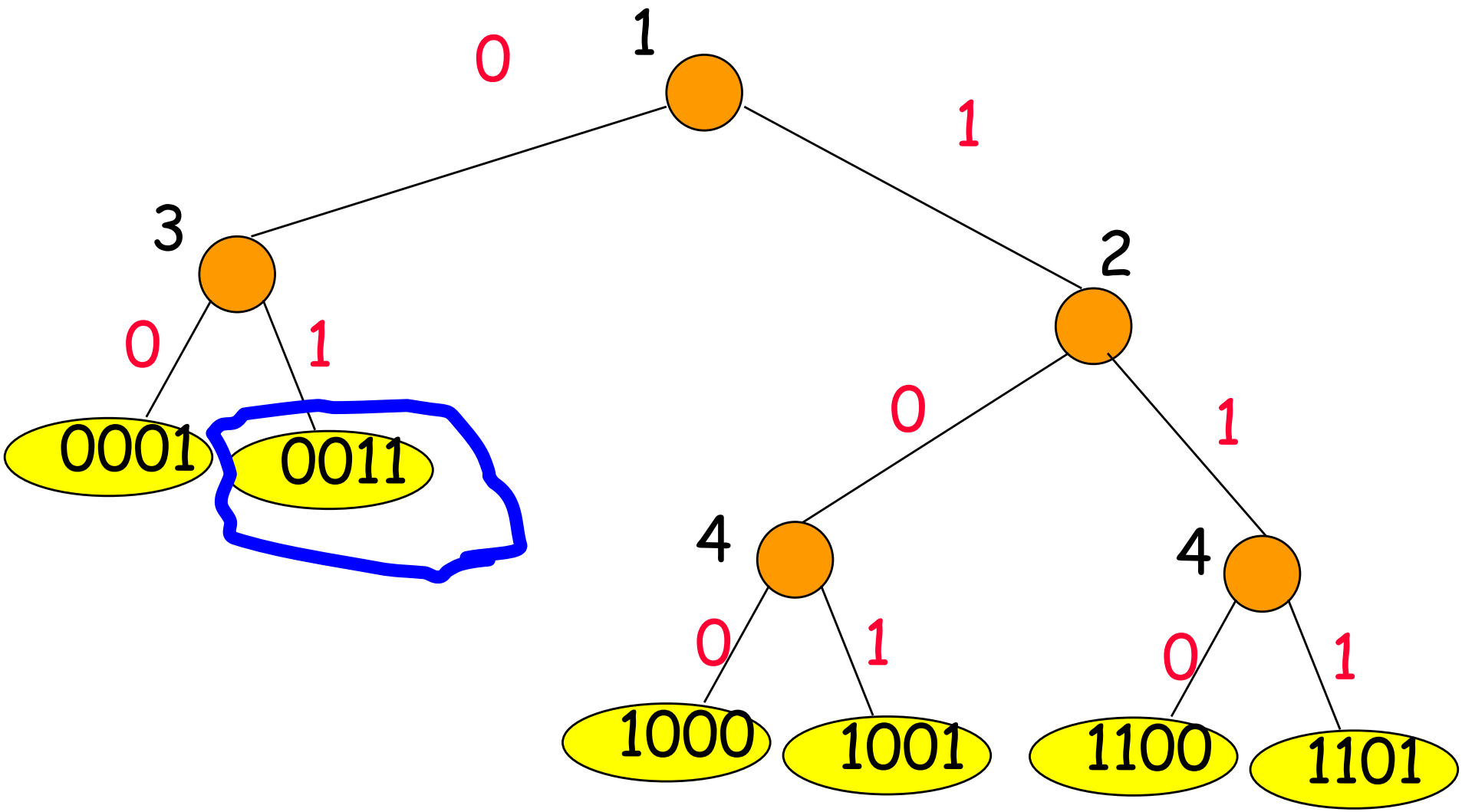
Compressed Binary Trie

n: 資料筆數 (黃色節點)



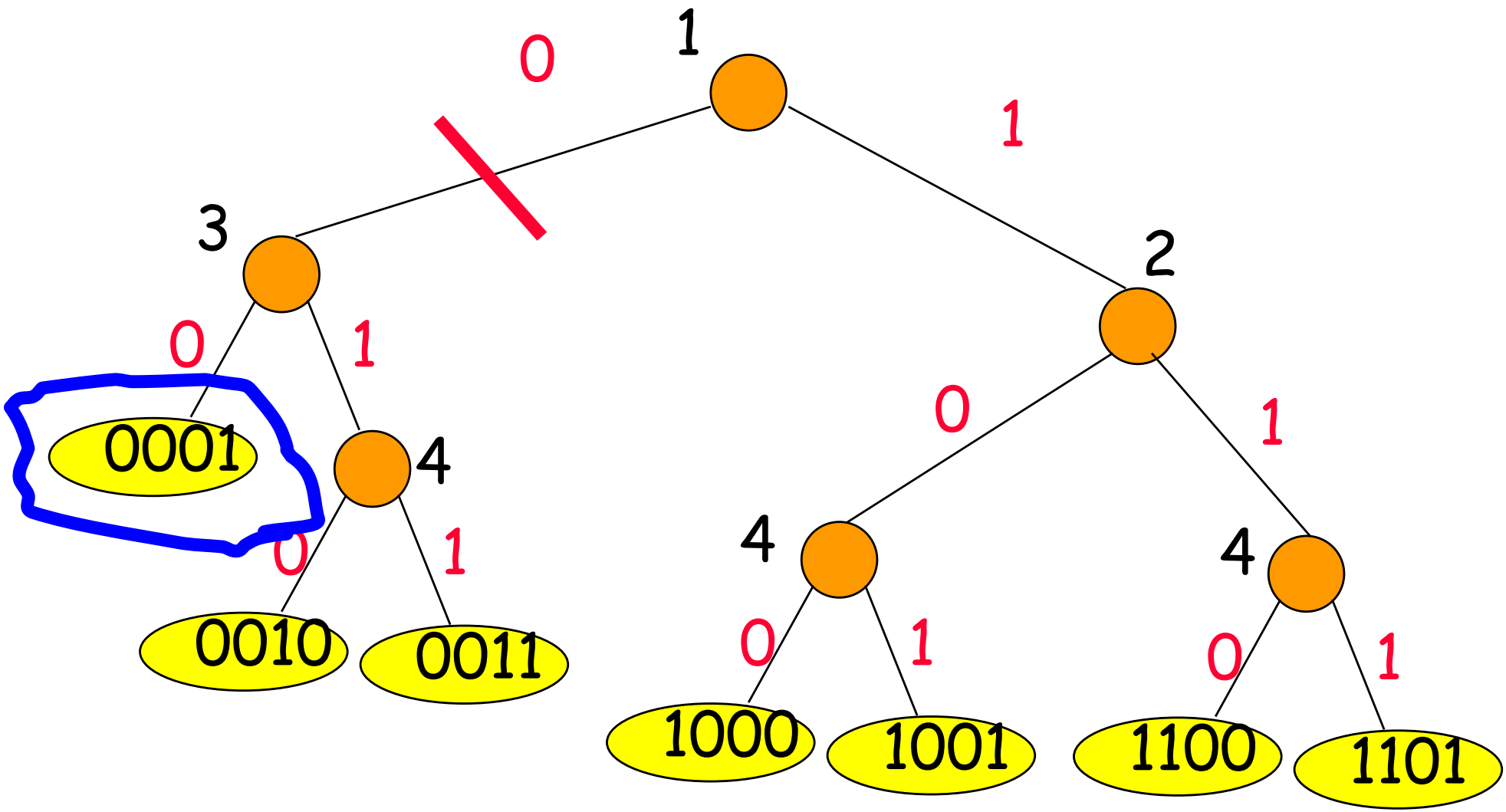
#branch nodes = n - 1. //某次上課說過

Insert



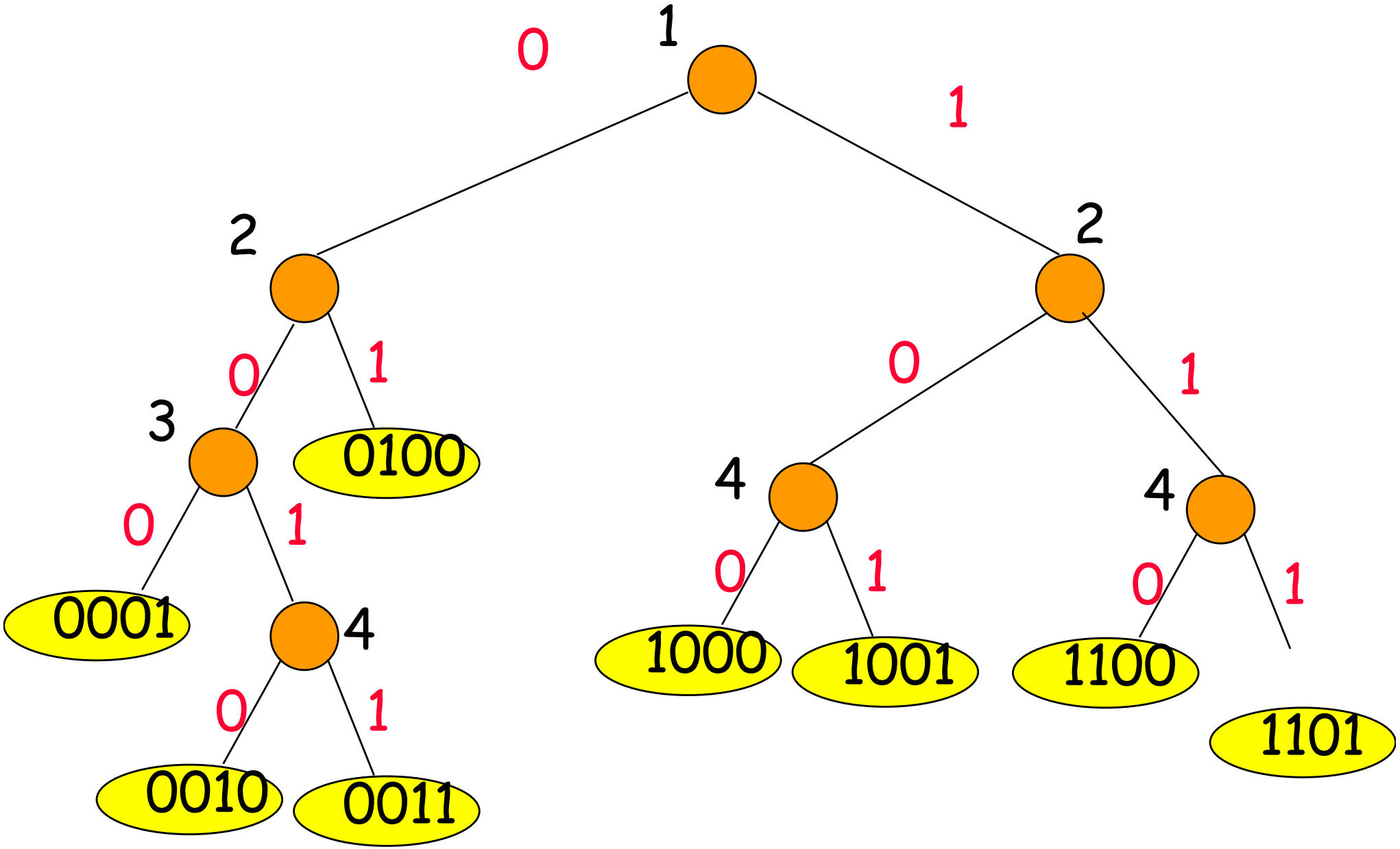
Insert 0010.

Insert

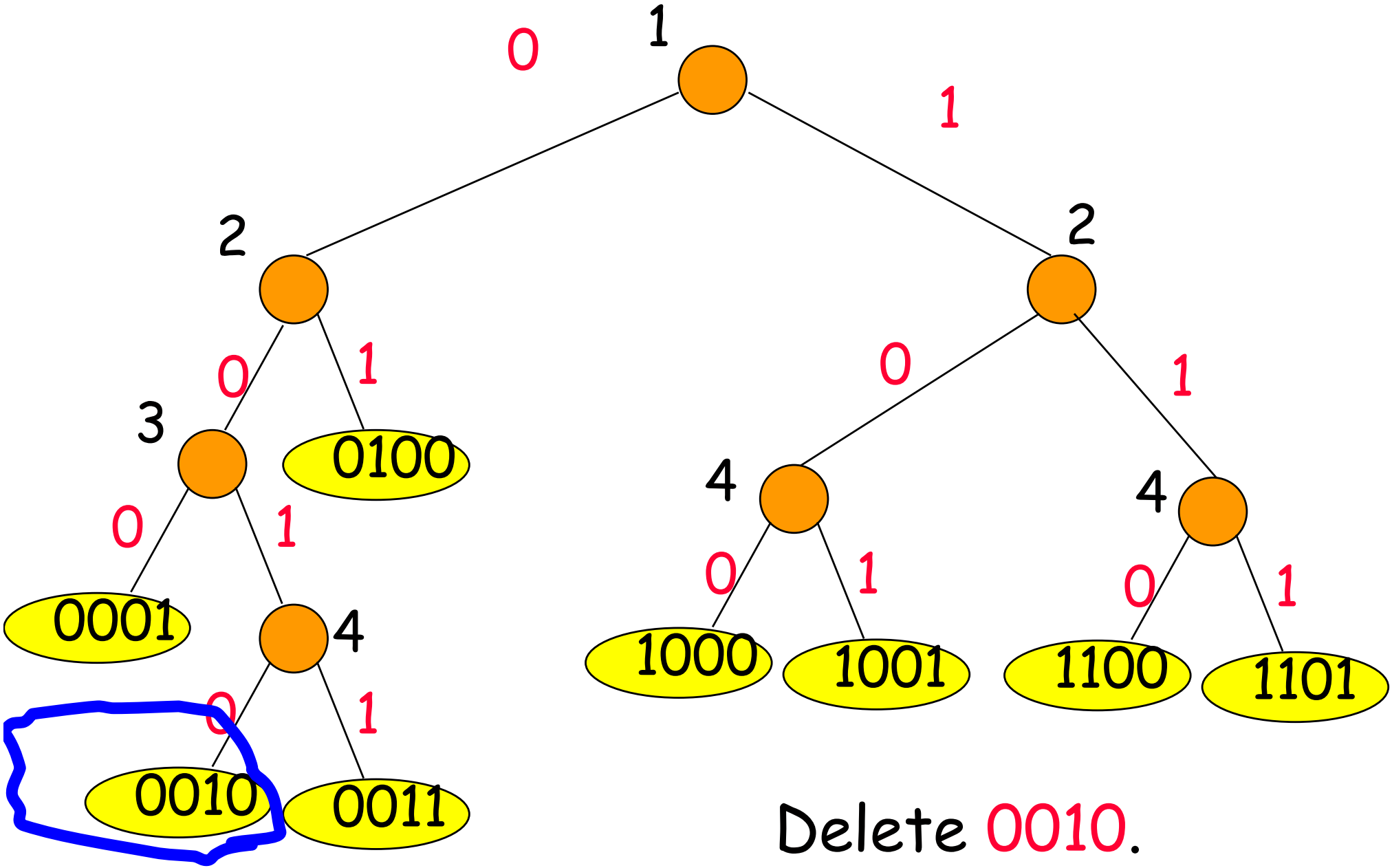


Insert 0100.

Insert

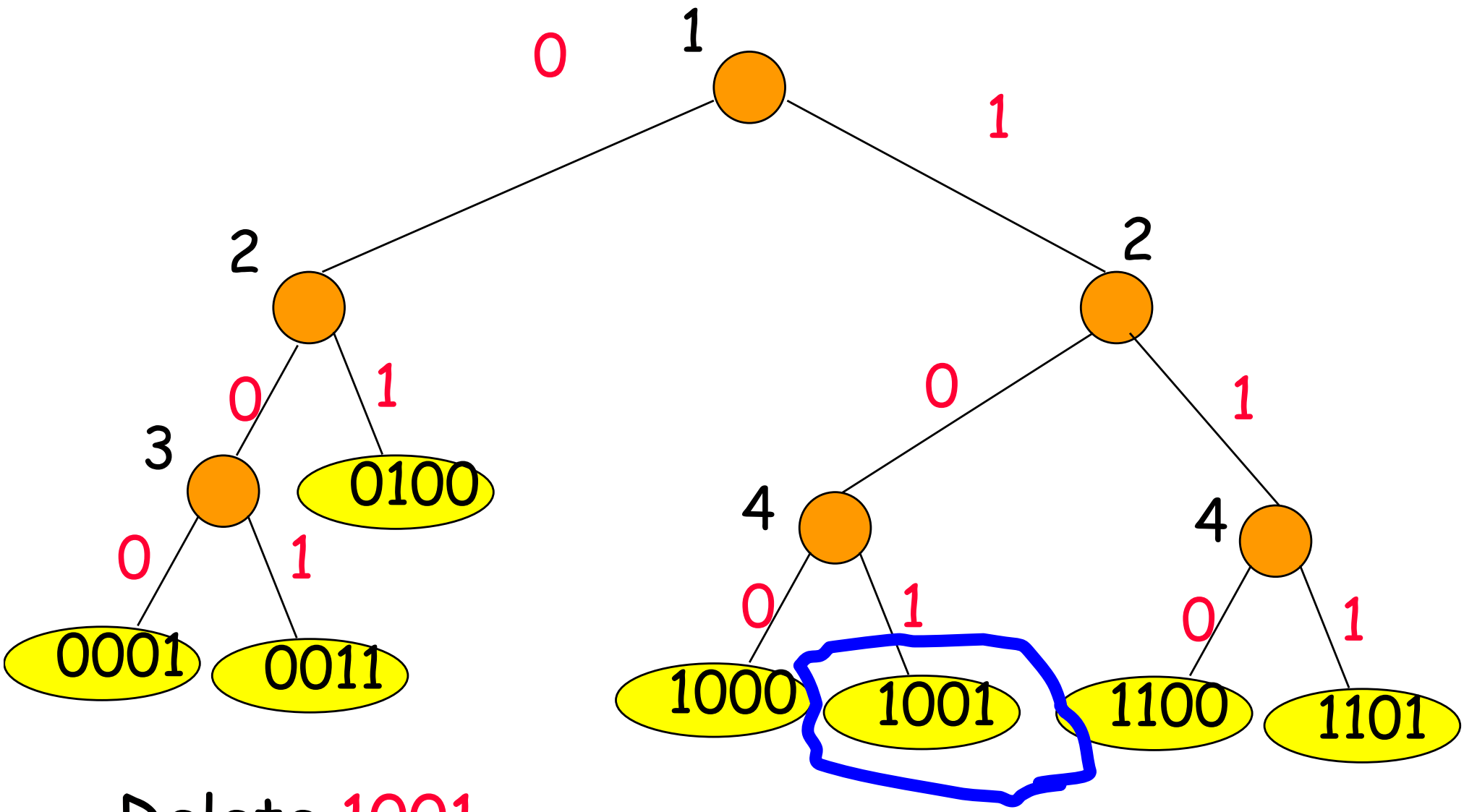

1
0
1
2
2
0
1
0
1
3
0100
4
4
0
1
0
1
0
1
0001
4
1000
1001
1100
1101
0
1
0010
0011

Delete



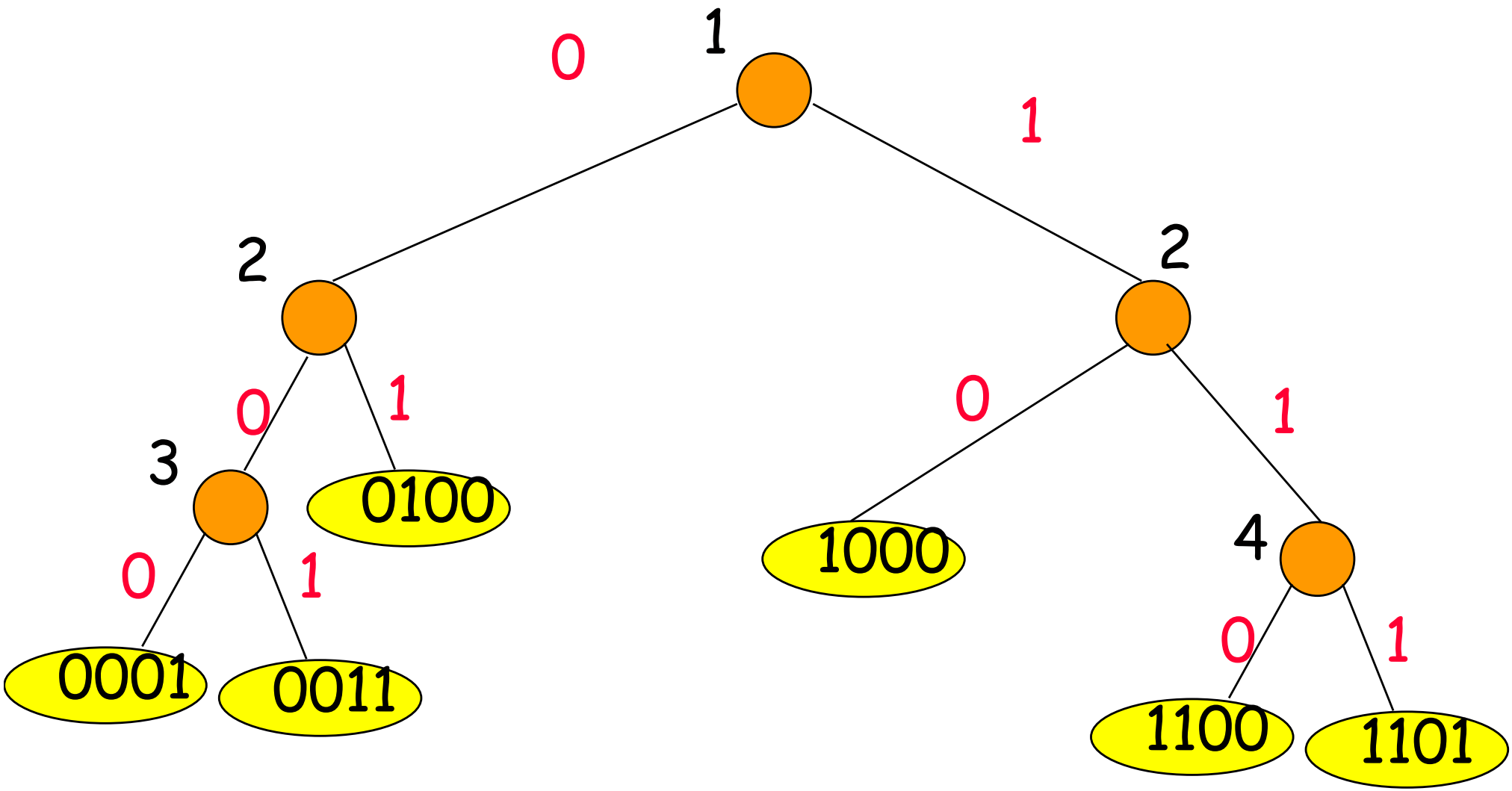
Delete 0010.

Delete



Delete 1001.

Delete



效能 (Performance)

digital search tree ✔

降低比較次數 (縮短執行時間)

trie ✔

減少索引使用空間 (黃色節點)

compressed trie ✔

減少資料使用空間 (黃色節點儲存資料)

patricia

More on Performance "Metrics"

- Typically,
  - Time
  - Space

- Can be extended to any performance metric:
  - Availability
  - Reliability
  - 預測成功率