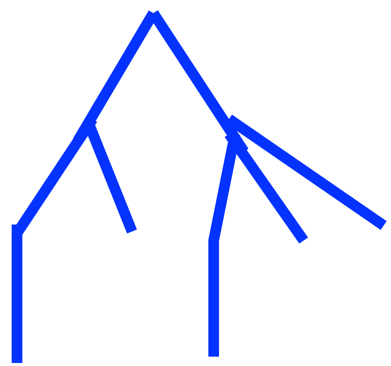
Advanced Data Structures

Succinct Data Structures

A bi a O de ed T ee

- U e a e he i a i
- Re e e he ee

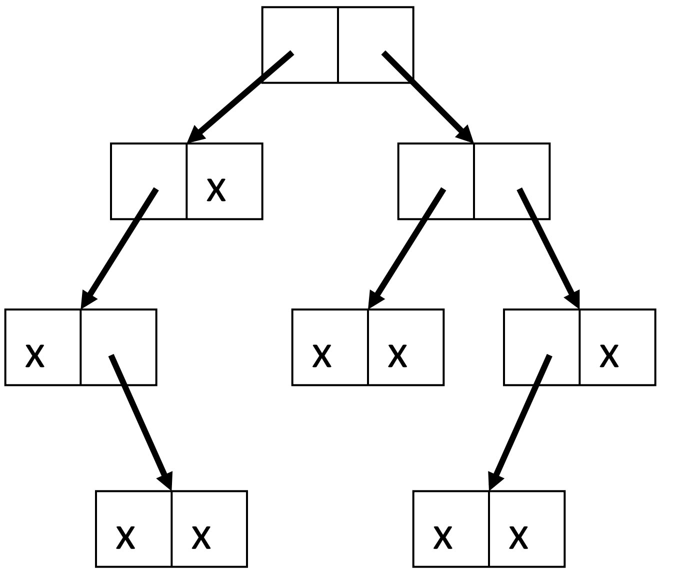


- A he bi a i g (((())())((())()())):
  a e e ee a “(“ f de, he b ee , he “)”
- 2 Bi e de

Space for trees

•

Standard representation



Binary tree: each node has two pointers to its left and right children

An n-node tree takes
2n pointers or 2n lg n bits

Supports finding left child or right child of a node (in constant time).

For each extra operation (eg. parent, subtree size) we have to pay, roughly, an additional n lg n bits.

Can we improve the space bound?

- There are less than 2^{2n} distinct binary trees on n nodes.

- $2n$ bits are enough to distinguish between any two different binary trees.

- Can we represent an n node binary tree using $2n$ bits?

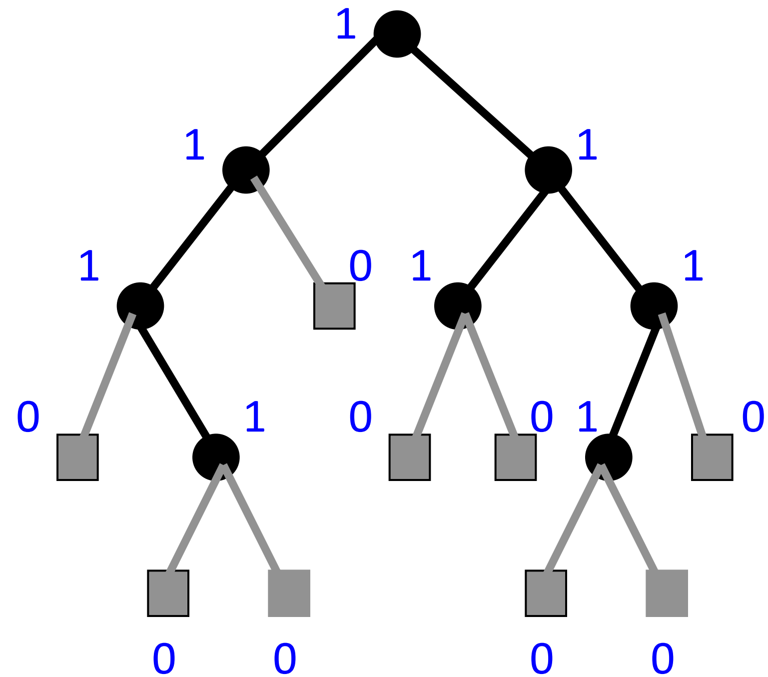
Heap-like notation for a binary tree

Add external nodes

Label internal nodes with a 1
and external nodes with a 0

Write the labels in level order

1 1 1 1 0 1 1 0 1 0 0 1 0 0 0 0 0



One can reconstruct the tree from this sequence

An n node binary tree can be represented in $2n$ 1 bits.

What about the operations?

Heap-like notation for a binary tree

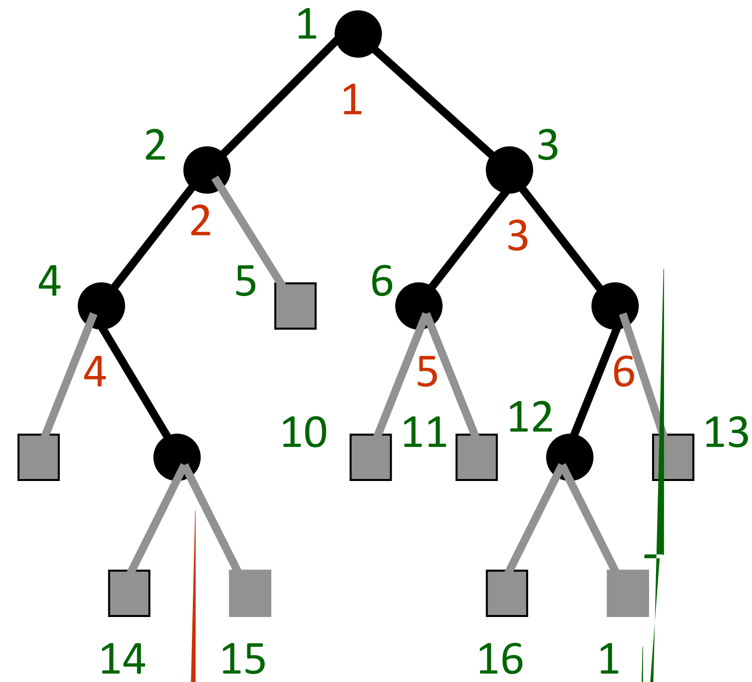
left child(x) [2x]

right child(x) [2x 1]

parent(x) [⌊x/2⌋]

x → x: # 1's up to x

x → x: position of x-th 1



1 2 3 4 5 6

1 1 1 1 0 1 1 0 1 0 0 1 0 0 0 0 0

1 2 3 4 5 6 10 11 12 13 14 15 16 1

Rank/Select on a bit vector

Given a bit vector B

1 2 3 4 5 6 10 11 12 13 14 15

B: 0 1 1 0 1 0 0 0 1 1 0 1 1 1 1

$rank_1(i)$ # 1's up to position i in B

$select_1(i)$ position of the i-th 1 in B

(similarly $rank_0$ and $select_0$)

$rank_1(5)$ 3

$select_1(4)$

$rank_0(5)$ 2

$select_0(4)$

> Given a bit vector of length n, by storing an additional o(n)-bit structure, we can support all four operations in constant time.

An important substructure in most succinct data structures.

Have been implemented.

Binary tree representation

- A binary tree on n nodes can be represented using $2n$ $o(n)$ bits to support:
  - parent
  - left child
  - right child

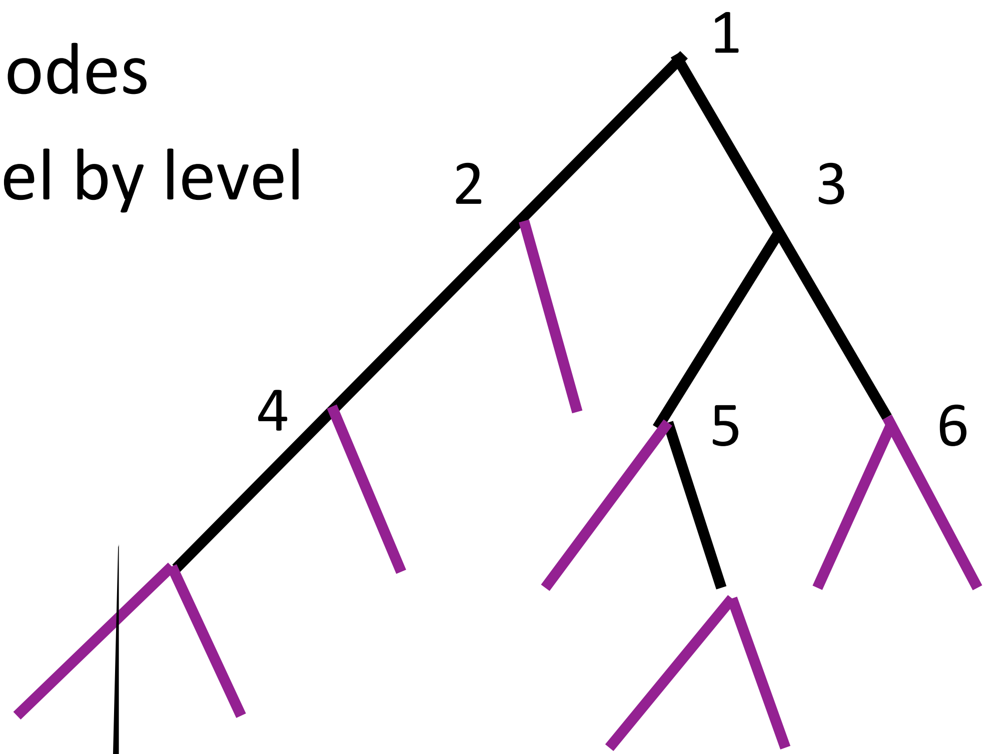
in constant time.

- 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0

Hea -i e N ai f a Bi a T ee

Add external nodes

Enumerate level by level



1 2 3 4 5 6

Store vector 1 1 1 -1 0 1 1 1 0 0 1 0 00000 length 2n 1

1 2 3 4 5 6 0 1 2 3456

Ordered trees

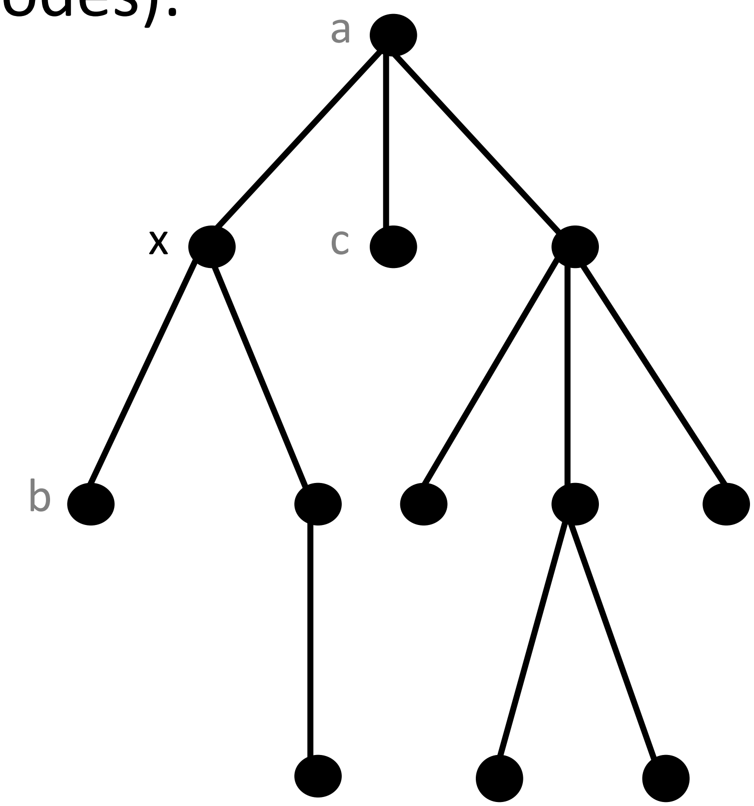
A rooted ordered tree (on n nodes):

Navigational operations:

- parent(x)   a
- first child(x)   b
- next sibling(x)   c

Other useful operations:

- degree(x)   2
- subtree size(x)   4



Ordered trees

- A binary tree representation taking 2n o(n) bits that supports parent, left child and right child operations in constant time.

- There is a one-to-one correspondence between binary trees and rooted ordered trees

- Gives an ordered tree representation taking 2n o(n) bits that supports first child, next sibling (but not parent) operations in constant time.

- We will now consider ordered tree representations that support more operations.

Level-order degree sequence

Write the degree sequence in level order

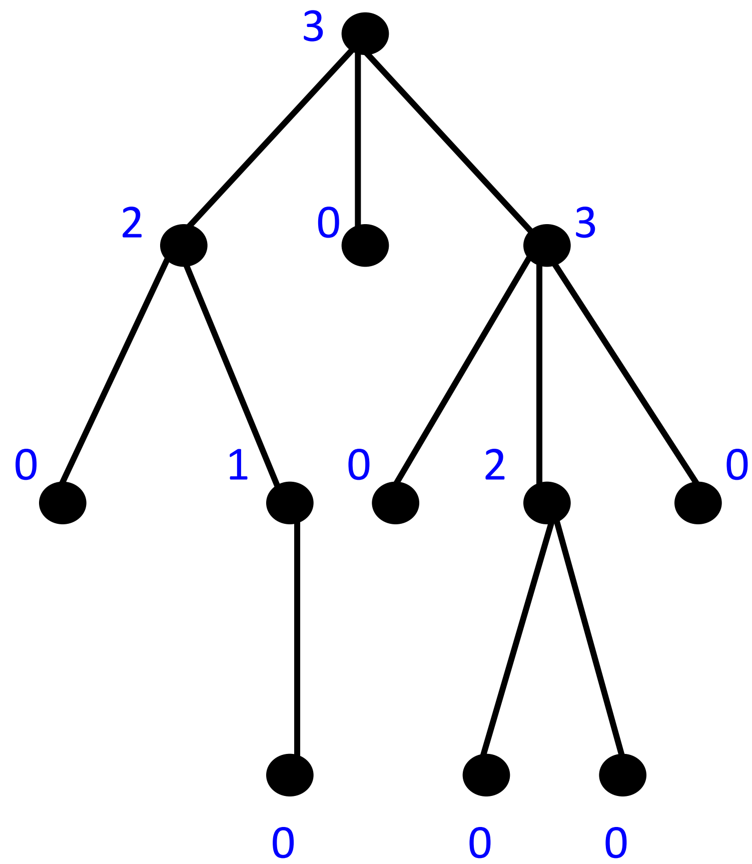
3 2 0 3 0 1 0 2 0 0 0 0

But, this still requires n lg n bits

Solution: write them in unary

1 1 1 0 1 1 0 0 1 1 1 0 0 1 0 0 1 1 0 0 0 0 0

Takes 2n-1 bits



A tree is uniquely determined by its degree sequence

Supporting operations

Add a dummy root so that each node has a corresponding 1

1 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 0 0 1 1 0 0 0 0 0

1   2 3 4   5 6   10   11 12