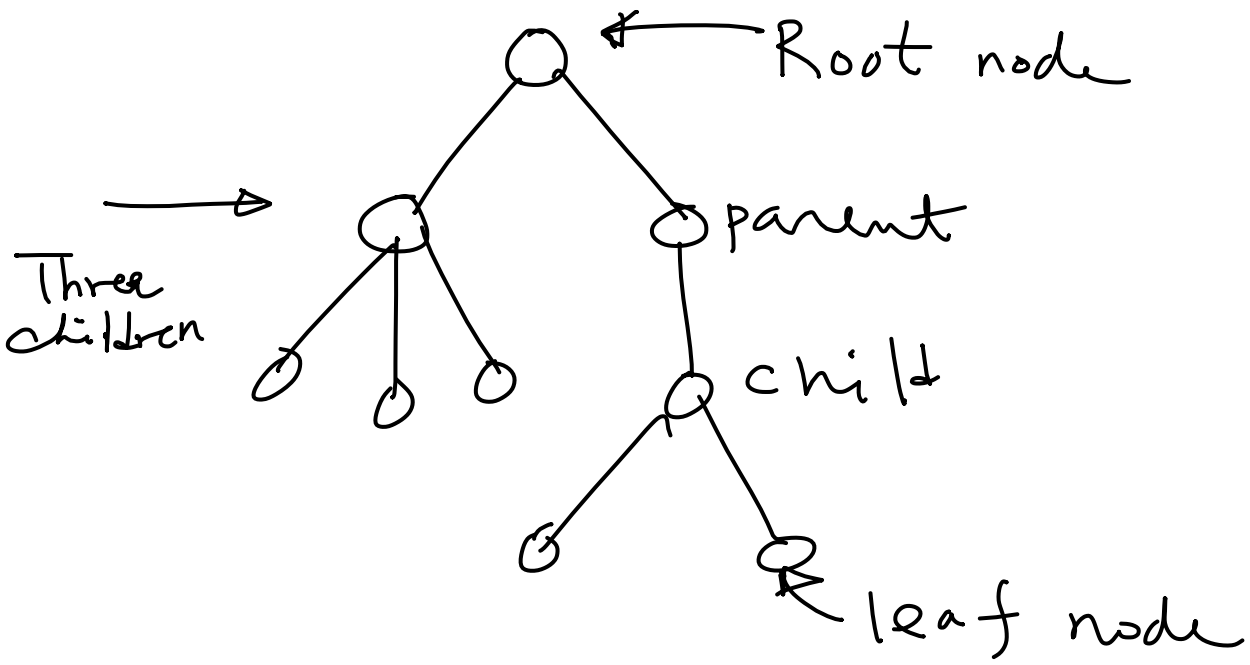
Trees



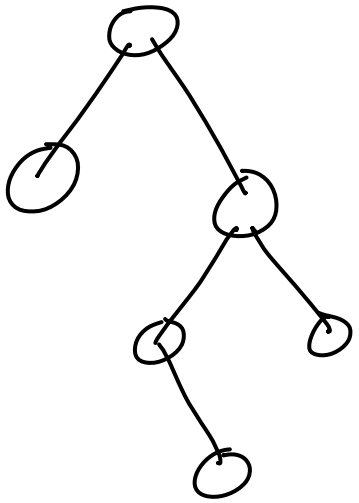
ordering on children

(leftmost child, rightmost child)

Rooted ordered Tree.

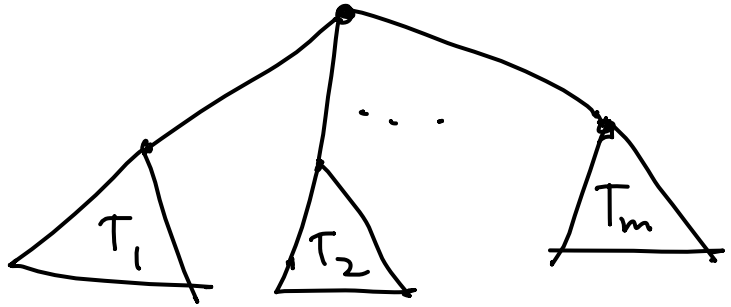
Binary Trees

Any node may have at most two children.



Inductive Definition (Rooted Ordered Trees)

- A single node is a tree
- If $T_1, \ldots, T_m$ are trees then



is also a tree

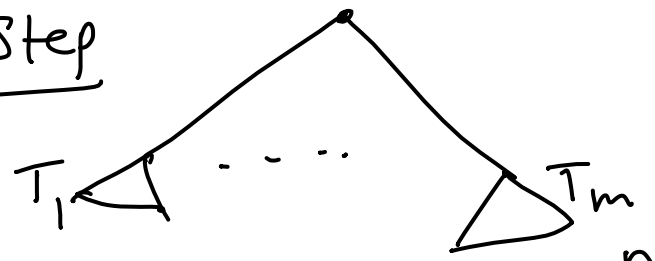
Lemma Any Tree with n nodes has $n-1$ edges.

Proof: By induction on n.

Base Case: $n=1$

- $n=1$ satisfied
- $e=0$

Induction Step



T_i $1 \le i \le m$

n_i nodes and $n_i - 1$ edges

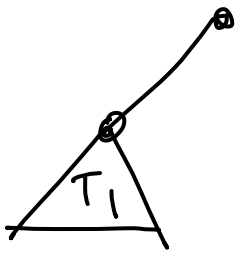
$$n = n_1 + n_2 + \cdots + n_m + 1$$

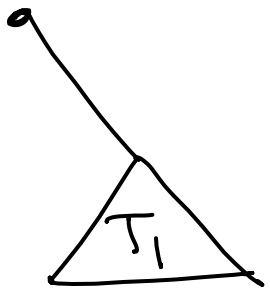
$$e = (n_1 - 1) + (n_2 - 1) + (n_m - 1) + m$$

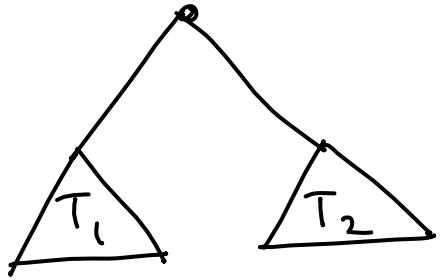
$$= n_1 + n_2 + \cdots + n_m = n - 1$$ □

Inductive Definition (Binary Trees)

- A single node is binary Tree
- If T_1, T_2 are binary trees then



,


,


are also binary trees.

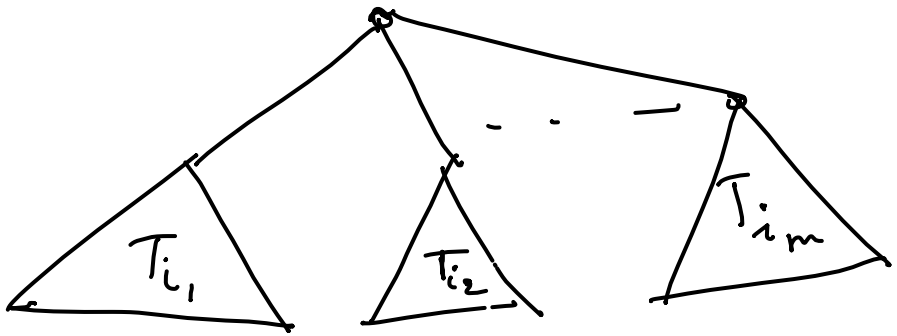
Not strictly special case of general definition

Right child may be present
without left child being present.

k-ary Trees

Each node has at most k children

- A single node is k-ary Tree.
- If $1 \le i_1 < i_2 < \cdots < i_m \le k$ and $T_{i_1}, T_{i_2}, \ldots, T_{i_m}$ are k-ary trees then



is also a k-ary tree.

Root has i_1^{th} child, i_2^{th} child, ..., i_m^{th} child

Lemma Let T be a binary Tree. Let n_i be the no. of nodes of degree i in T for $0 \le i \le 2$. (degree i = no. of children). Then $n_0 = n_2 + 1$.

Proof: No. of nodes in tree T be n.

$$n = n_0 + n_1 + n_2, \qquad e = 2n_2 + n_1$$

$$e = n - 1$$

$$2n_2 + n_1 = n_0 + n_1 + n_2 - 1$$

$$n_2 = n_0 - 1$$

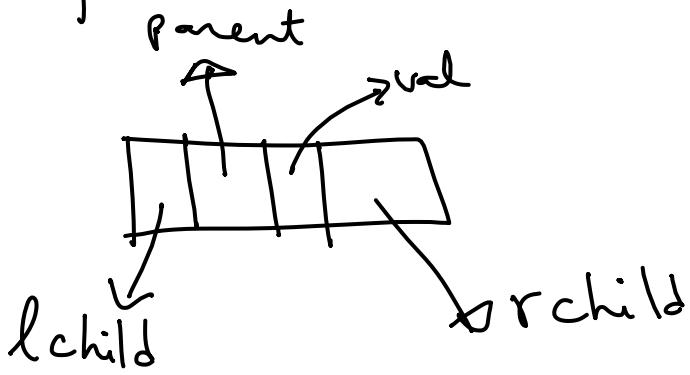
$$\Rightarrow n_0 = n_2 + 1$$

□

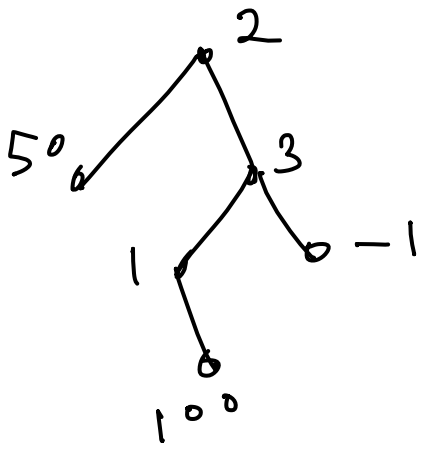
Representation of Trees in Computer

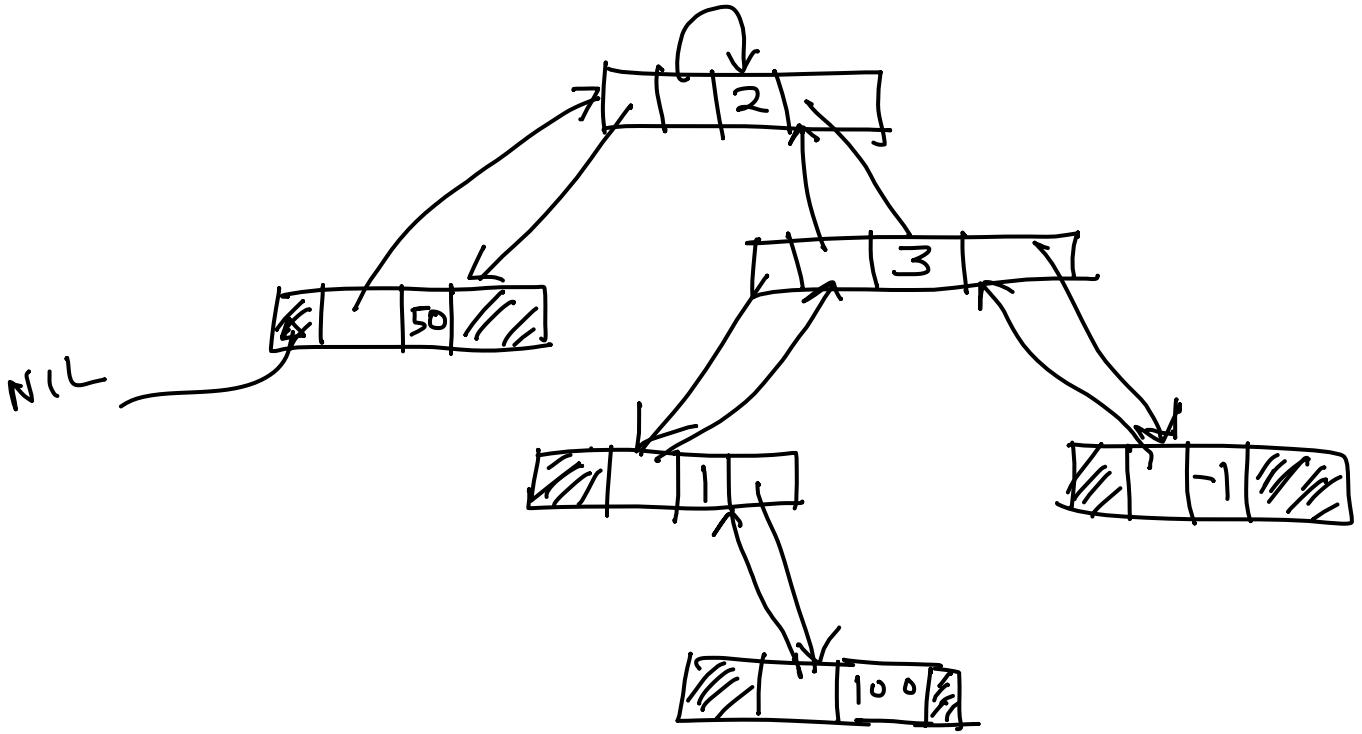
Common representation uses links (Exception: Heap)

Binary Trees



Example

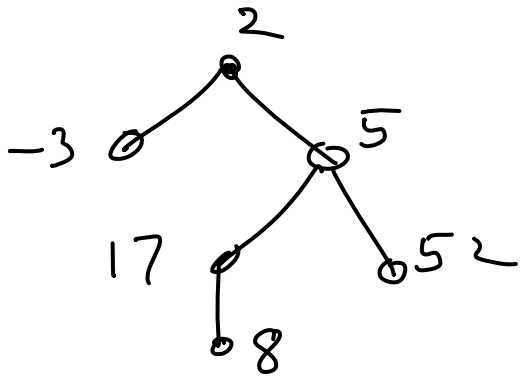




Prefix_Traversal

visit current node
visit left subtree
Right subtree

C L R



2, -3, 5, 17, 8, 52

Inorder / L C R

Post order L R C

```
Prefix_Traversal (T)
  If T ≠ nil then
        Print (T.val)
        Prefix_Traversal (T.lchild)
        Prefix_Traversal (T.rchild)
```

Non-recursive Algo using stacks

last ↓ first ↙

If $S = [T_1, \ldots, T_m]$ then
prints prefix-Trav(T_1), prefix-Trav(T_2)
.... prefix-Trav(T_m)

```
Prefix_Trav_stack (S)
  While S not empty do
      R = Pop(S)
      If R ≠ nil then
          Print (R.val)
          Push (S, R.rchild)
          Push (S, R.lchild)
```

Ex: Prove this specification for the program by induction.
(induction on what?)

Use of stack data structure to remove recursion.

Main

```
create_Stack(S)
Push(S,T)
Prefix_Trav_stack(S)
```

Exercise (difficult) Write the non-recursive procedure for prefix-Traversal without using stack.

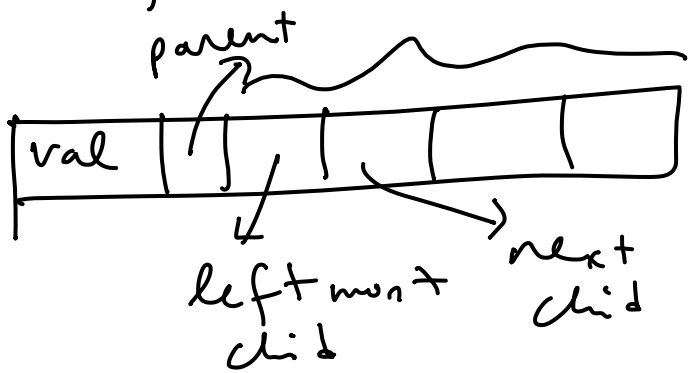
(need to use parent pointer,
need to see if moving to parent node
from its leftchild or rightchild)

Ex: Inorder & post order traversal
(Write recursive proc, Non-rec proc. using stack)

Representing arbitrary trees in Computer

no. of children of a node not bounded.

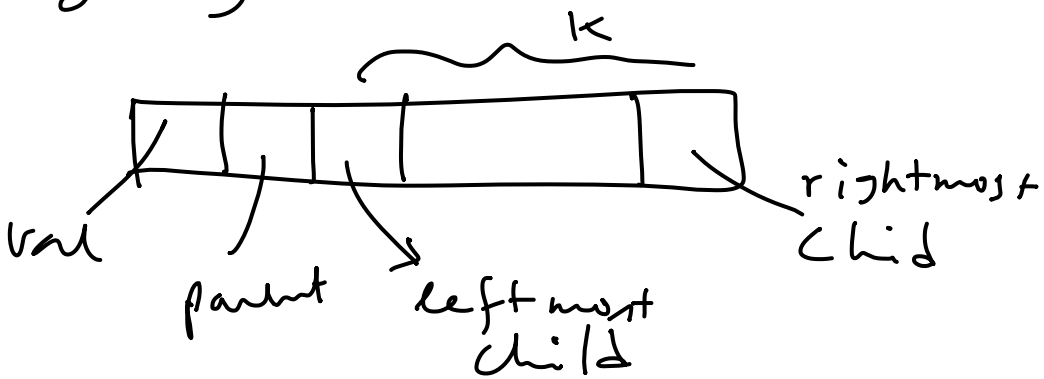
If we generalize bin-tree rep then



no. of fields in not known

Node can't have fixed structure

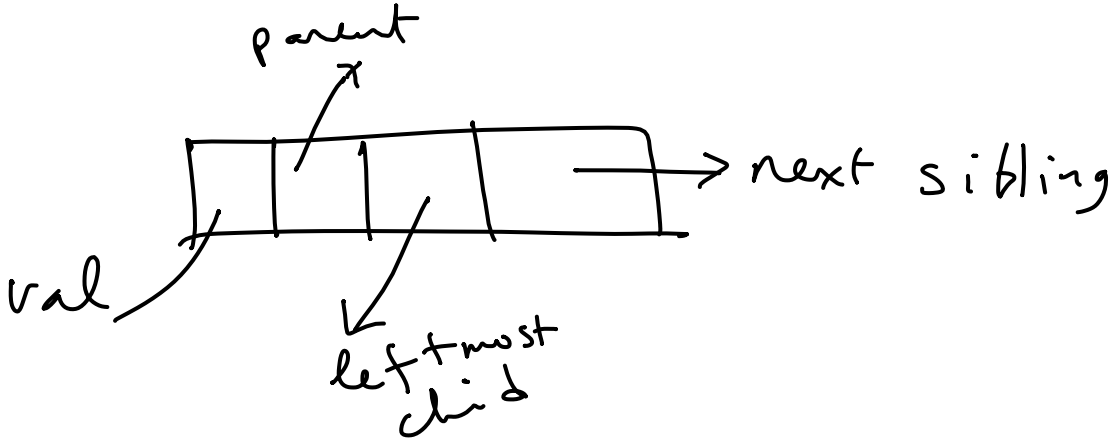
If we put some very big bound on the no. of children



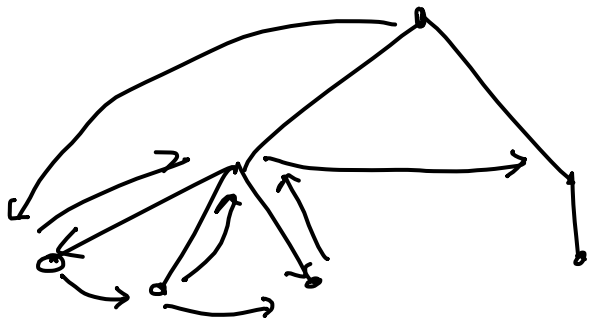
If most nodes have children << k then these nodes have many unused field

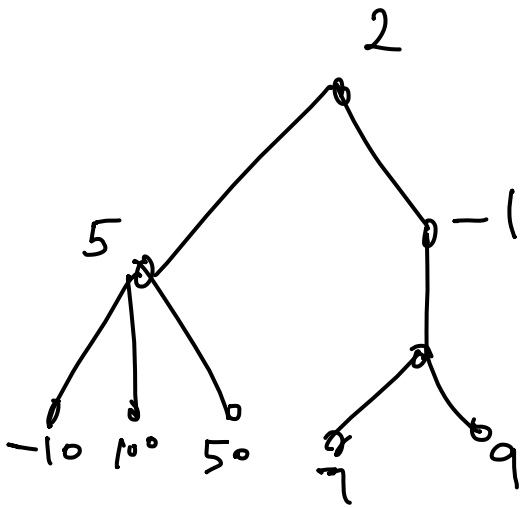
(leads to wastage of memory space)

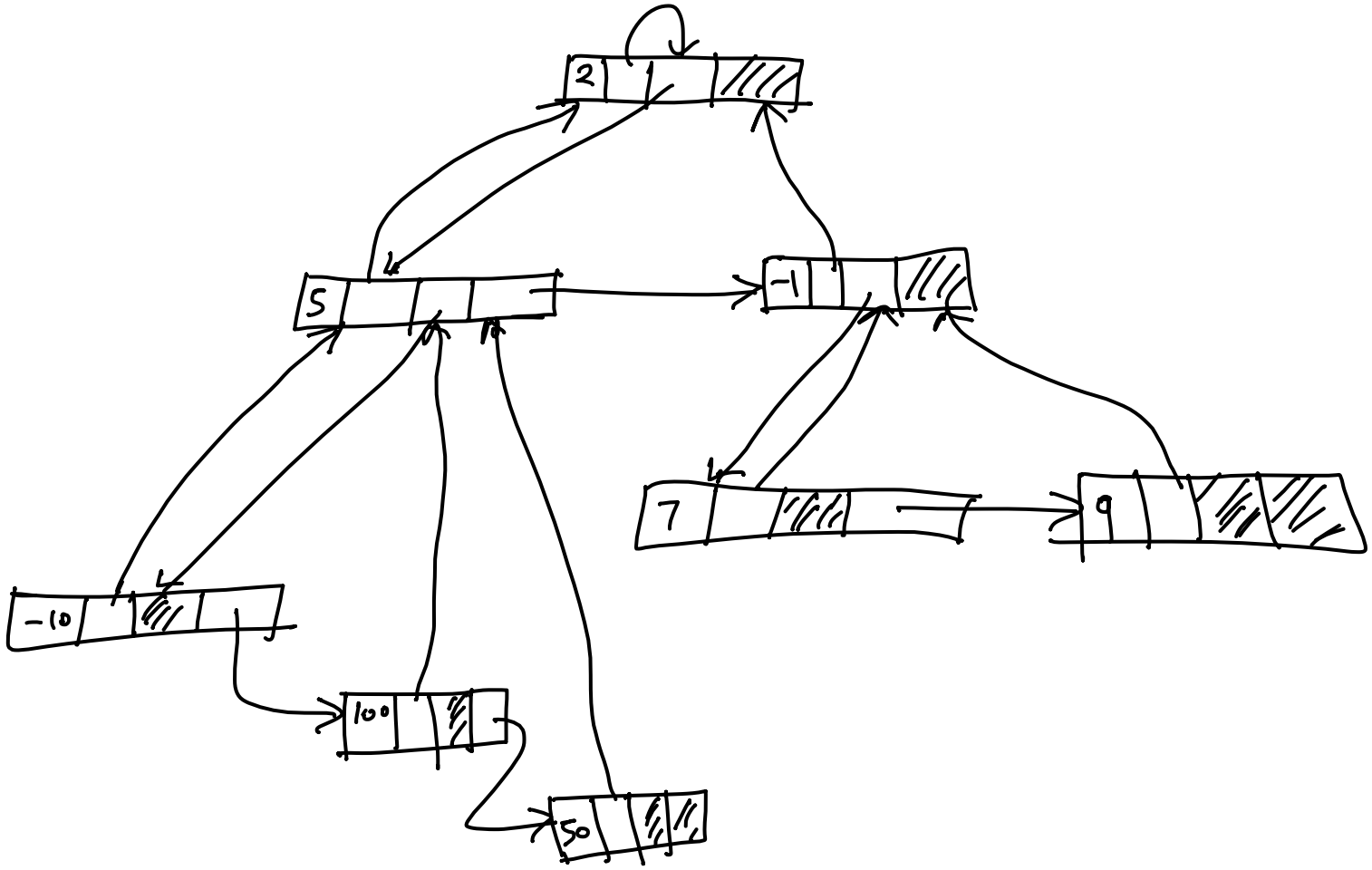
Better Solution



All children of a given node are linked linearly.
A node points only to its leftmost child.







Node have a fixed structure
with a small no. of field (4 fields).

Exercise Define prefix Traversal for such trees and write a pseudo code to print values in prefix order

(a) using recursion

(b) without recursion, by using a stack.