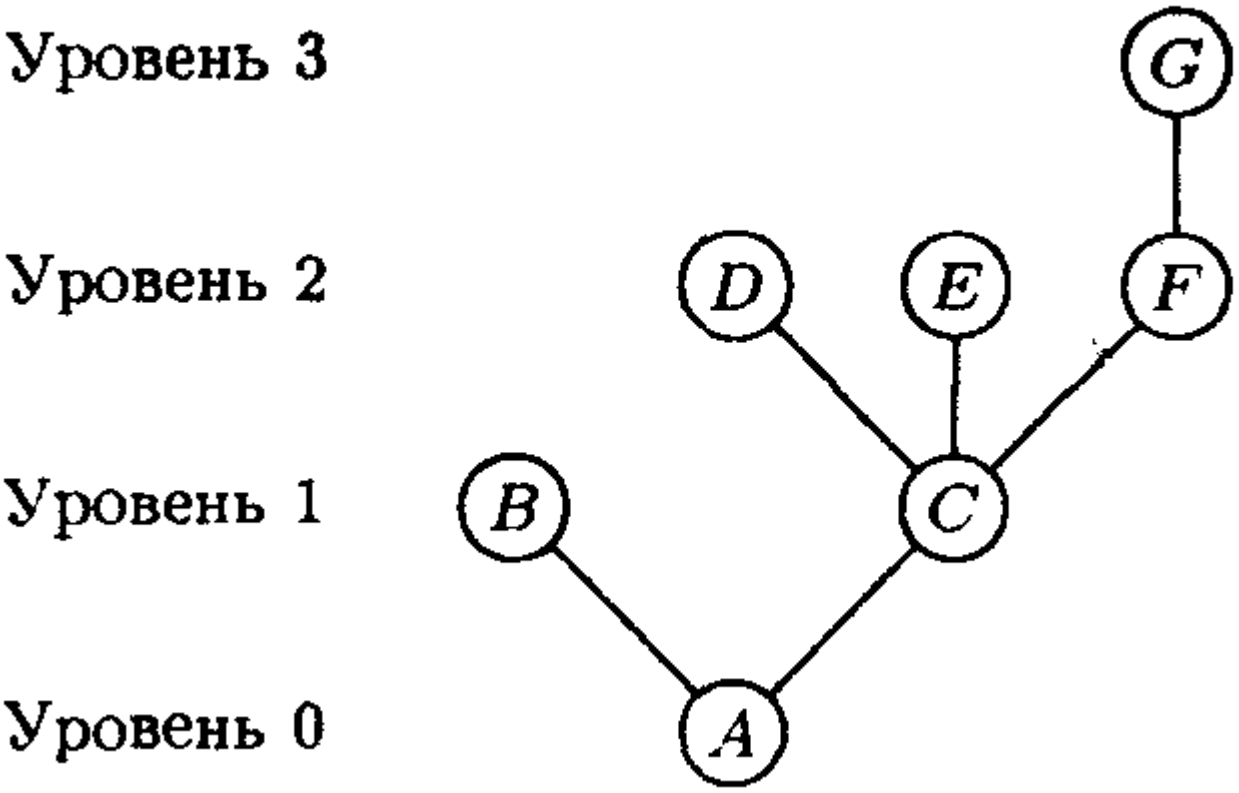
Деревья

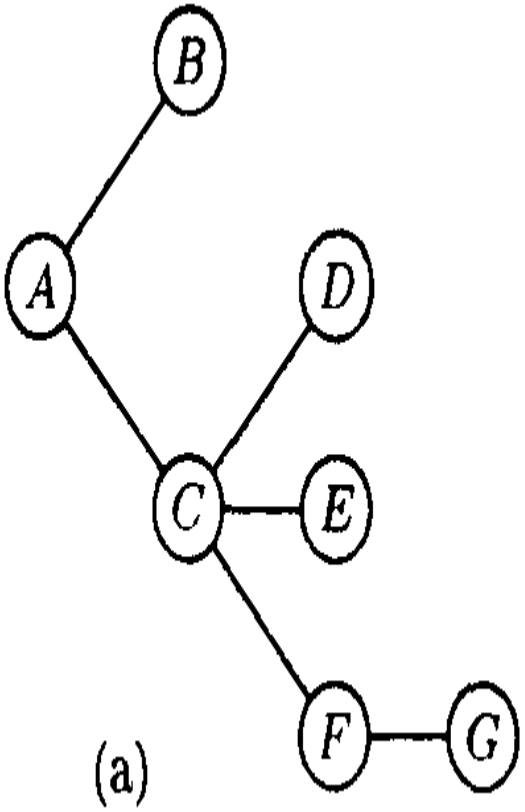
Формально *дерево (tree)* определяется как конечное множество *T* одного или более узлов со следующими свойствами:

- **существует один выделенный узел — *корень (root)* данного дерева *T;***
- **остальные узлы (за исключением корня) распределены среди $m > 0$ непересекающихся множеств $T_1,\ldots,T_m$, и каждое из этих множеств, в свою очередь,является деревом; деревья $T_1,\ldots,T_m$ называются *поддеревьями (subtrees)* данного корня.**

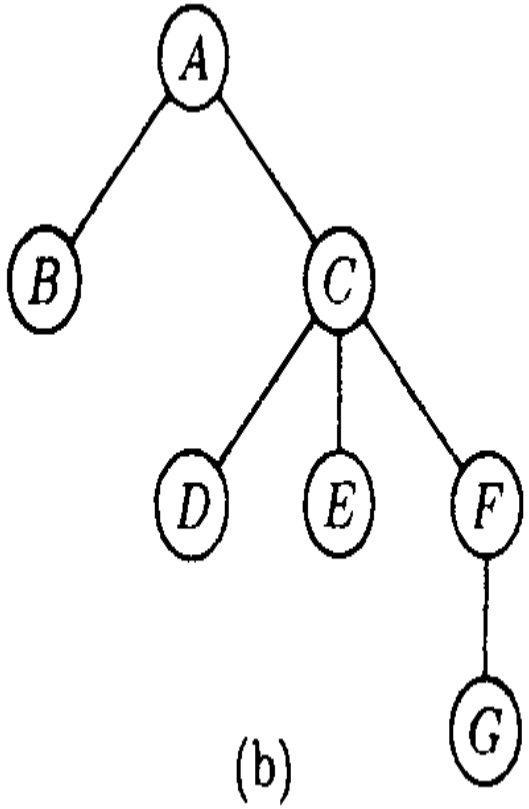
Деревья



Деревья

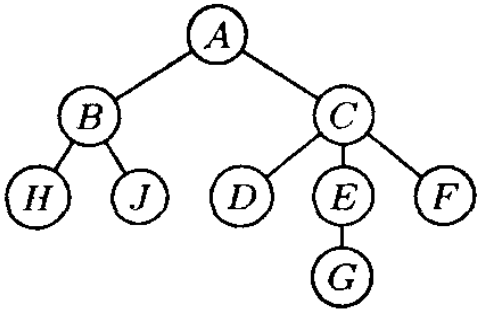


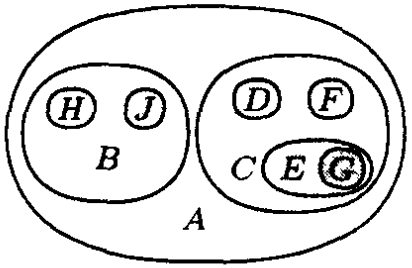
(a)

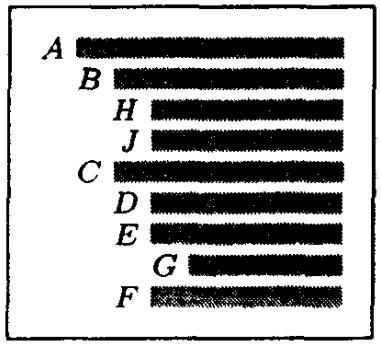


(b)

Деревья







Деревья

- **С точки зрения представления в памяти важно различать два типа деревьев: бинарные и сильноветвящиеся.**

Деревья

- **Степенью узла в дереве называется количество дуг, которое из него выходит (число поддеревьев)**
- **Степень дерева равна максимальной степени узла, входящего в дерево.**

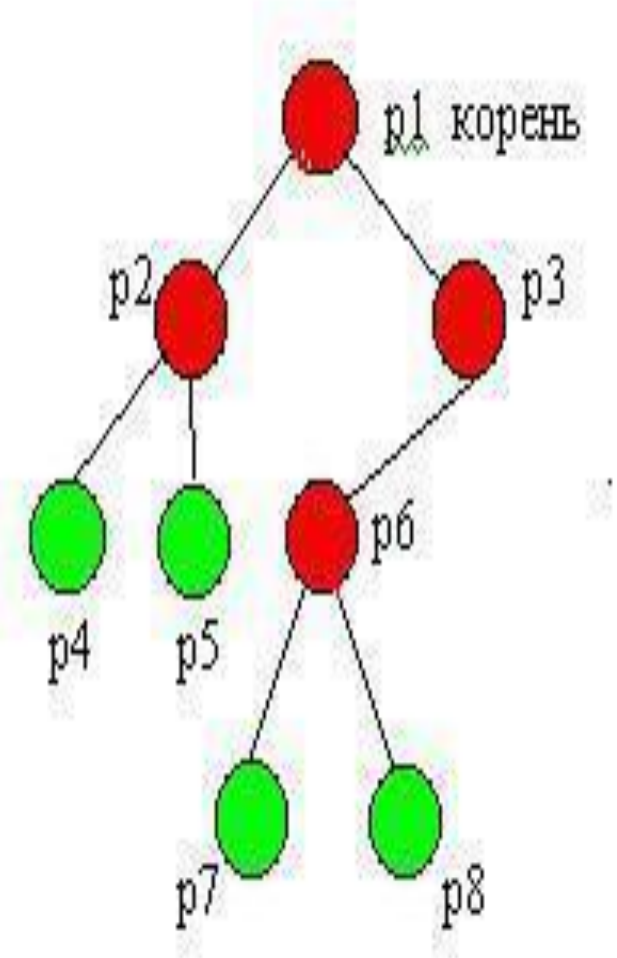
Бинарные деревья

Степень узла бинарного дерева не превышает 2. При этом листьями (leaf) в дереве являются вершины, имеющие степень ноль.

Бинарное дерево - это *конечное множество* узлов, *которое является пустым* или состоит из *корня* и *двух*

непересекающпхся бинарных *деревьев*, которые называются левым и правым поддеревьями данного корня.

Бинарные деревья

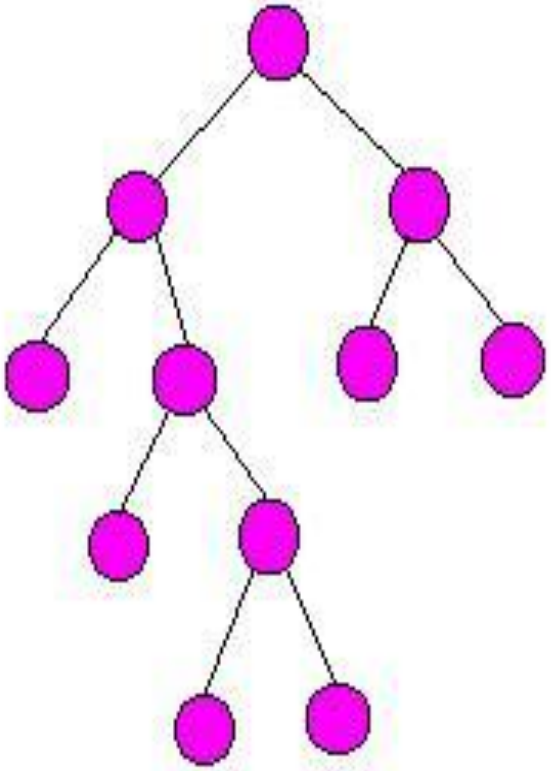


p1, p2, p3, p4, p5, p6, p7, p8 - узлы

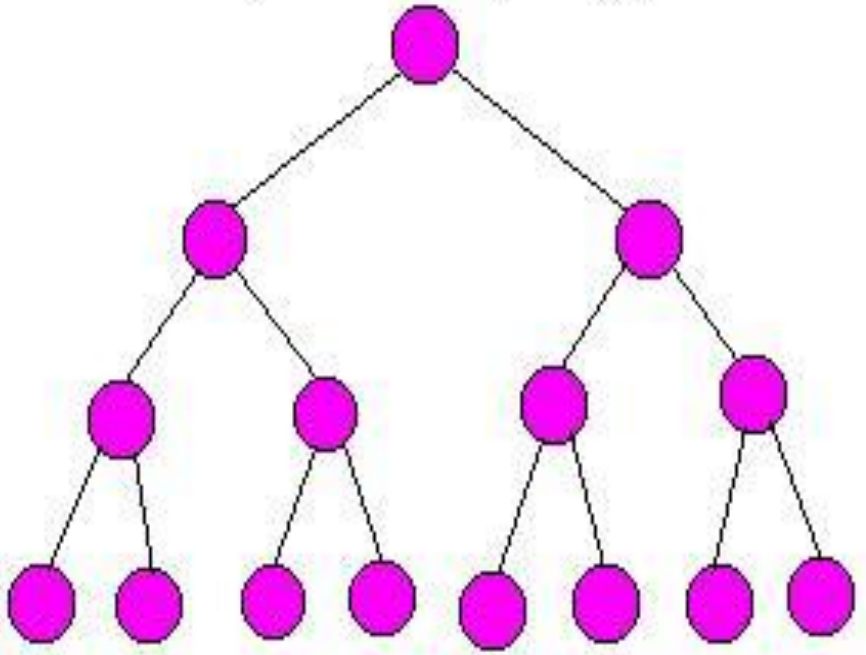
p4, p5, p7, p8 - листья

Бинарные деревья

а) неполное бинарное дерево



б) полное бинарное дерево



- на всех уровнях меньше, чем n, узлы имеют степень 2, на уровне n - 0

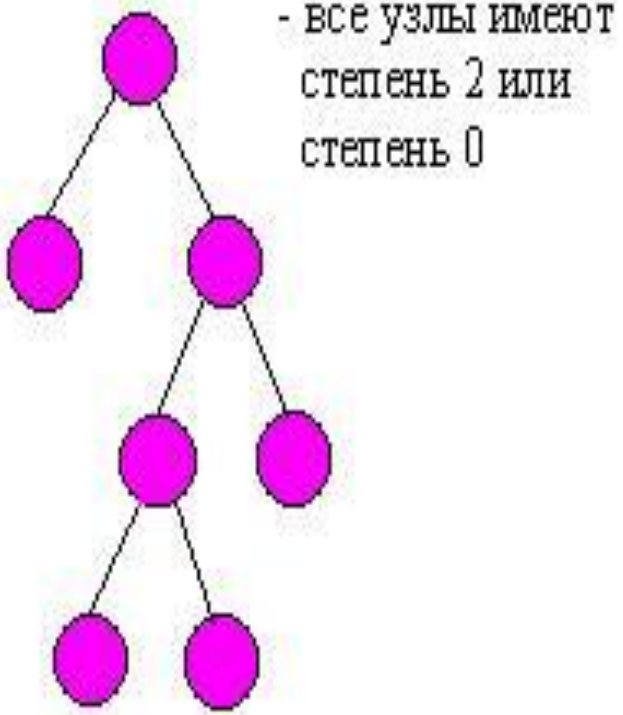
Бинарные деревья

- **Строго бинарное дерево состоит только из узлов, имеющих степень два или степень ноль.**

- **Нестрого бинарное дерево содержит узлы со степенью равной одному.**

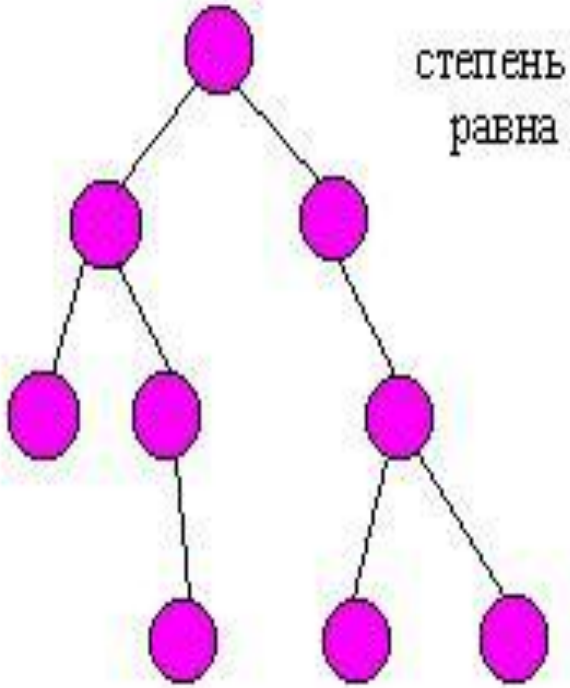
Бинарные деревья

а) строго бинарное дерево



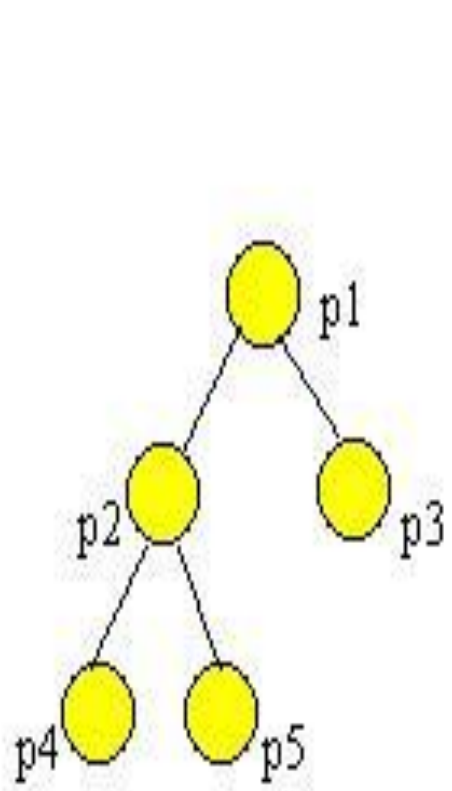
- все узлы имеют степень 2 или степень 0

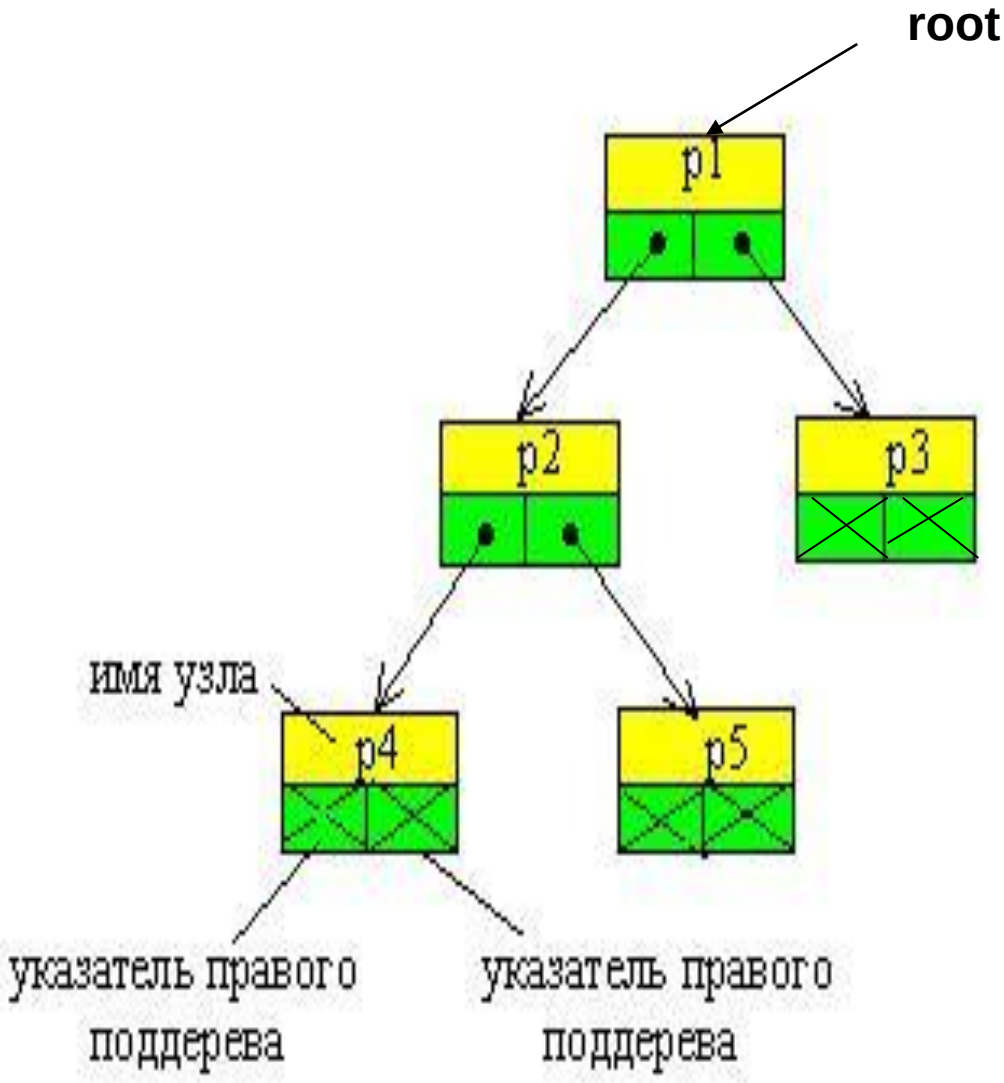
б) не строго бинарное дерево



степень узлов равна 2, 1, 0

Представление бинарных деревьев





Бинарные деревья

```
Type
 PTree = ^TTree;
 TTree = record
              Item: T; {элемент дерева}
               Left, Right: PTree;
             {указатели на поддеревья}
            end;
```

где **Left, Right** равны **nil**, если соответствующие поддеревья пусты

Бинарные деревья

Пусть мы имеем дело с полным двоичным деревом, состоящим из n уровней.

Можно организовать такое хранение дерева в массиве Value: array[1..N] of T, что:

Value[1] – корень дерева;

Value[2*i] – левый сын вершины Value[i];

Value[2*i+1] – правый сын вершины Value[i].

Бинарные деревья

Адрес любой вершины в массиве вычисляется как

$$адрес = 2^{к-1}+i-1,$$

где k-номер уровня вершины, i- номер на уровне k в полном бинарном дереве.

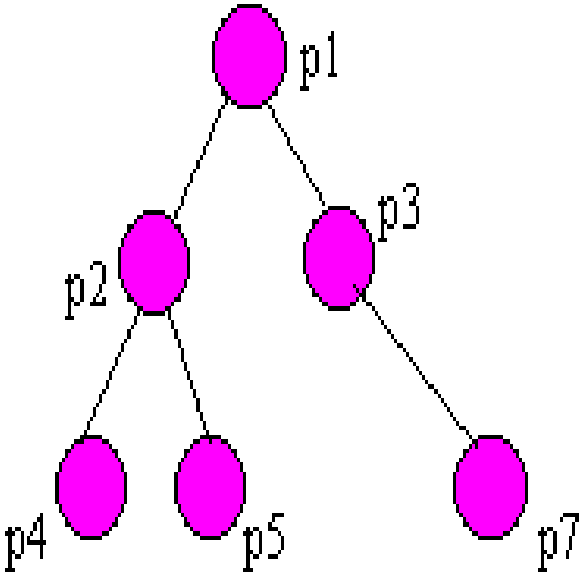
Адрес корня будет равен единице.

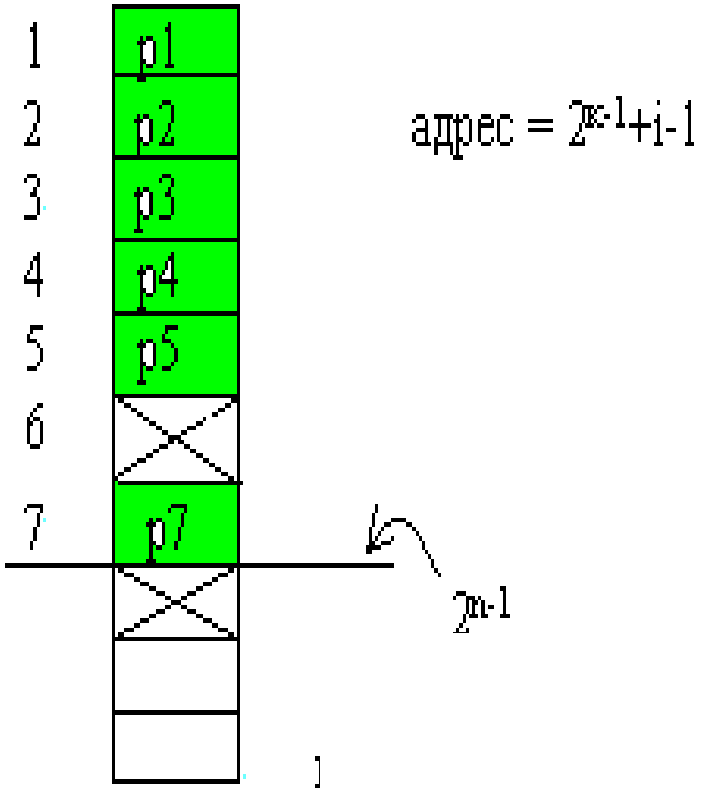
Для любой вершины можно вычислить адреса левого и правого потомков

$$адрес\_L = 2^{к}+2(i-1)$$

$$адрес\_R = 2^{к}+2(i-1)+1$$

Бинарные деревья





Прохождение (traversing/walk) бинарных деревьев

Пошаговый перебор элементов дерева по связям между предками-узлами и потомками-узлами называется обходом дерева

Прохождение (обход) бинарных деревьев

Обход, при котором каждый узел-предок просматривается прежде его потомков называется предупорядоченным обходом или обходом в прямом порядке (pre-order walk),

а когда просматриваются сначала потомки, а потом предки, то обход называется поступорядоченным обходом или обходом в обратном порядке (post-order walk).

Прохождение бинарных деревьев

Существует также симметричный обход, при котором посещается сначало левое поддерево, затем узел, затем - правое поддерево,

и обход в ширину, при котором узлы просматриваются уровень за уровнем.

Каждый уровень обходится слева направо.

Прохождение бинарных деревьев

Первые три способа обхода рекурсивно можно определить следующим образом:

1. **если дерево *Tree* является пустым деревом, то в список обхода заносится пустая запись**
2. **если дерево *Tree* состоит из одной вершины, то в список обхода записывается эта вершина;**
3. **если *Tree* дерево с корнем n и поддеревьями $Tree_1$, $Tree_2$, …, $Tree_k$,…**

то :

Прохождение бинарных деревьев

- **при прохождении в прямом порядке сначала посещается корень и, затем в прямом порядке узлы поддерева**

***Tree*$_1$**

далее в прямом порядке узлы поддерева *Tree*$_2$ и т. д.

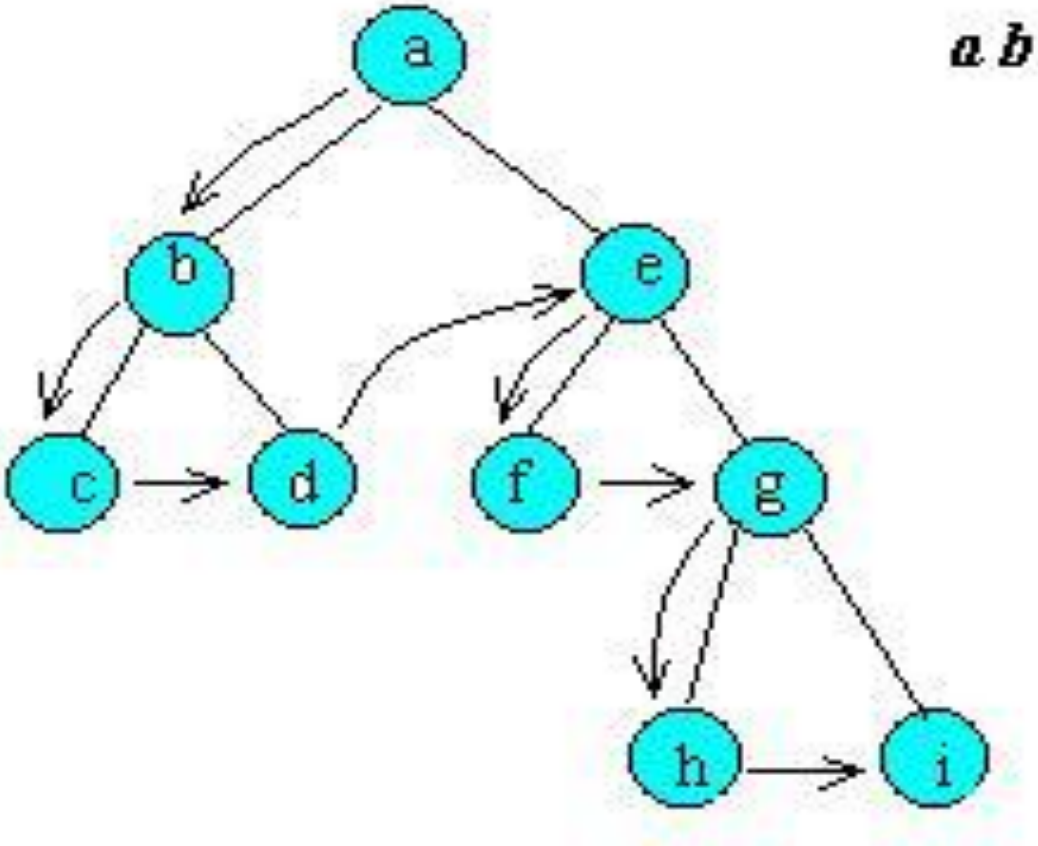
последними в прямом порядке посещаются узлы поддерева *Tree*$_k$

Прохождение бинарных деревьев

те, прямой порядок прохождения бинарного дерева можно определить следующим образом:

1. **попасть в корень**
2. **пройти в прямом порядке левое поддерево**
3. **пройти в прямом порядке правое поддерево**

Прохождение бинарных деревьев



a b c d e f g h i

Прохождение бинарных деревьев

procedure PreOrder(n: вершина);

{Обход дерева в прямом порядке}

Занести в список обхода вершину n;

for для каждого потомка s вершины n в порядке слева направо do

PreOrder(s);

Прохождение бинарных деревьев

- **при прохождении в обратном порядке сначала посещаются в обратном порядке узлы поддерева $Tree_1$**

далее последовательно в обратном порядке посещаются узлы поддеревьев $Tree_2$ ..., $Tree_k$

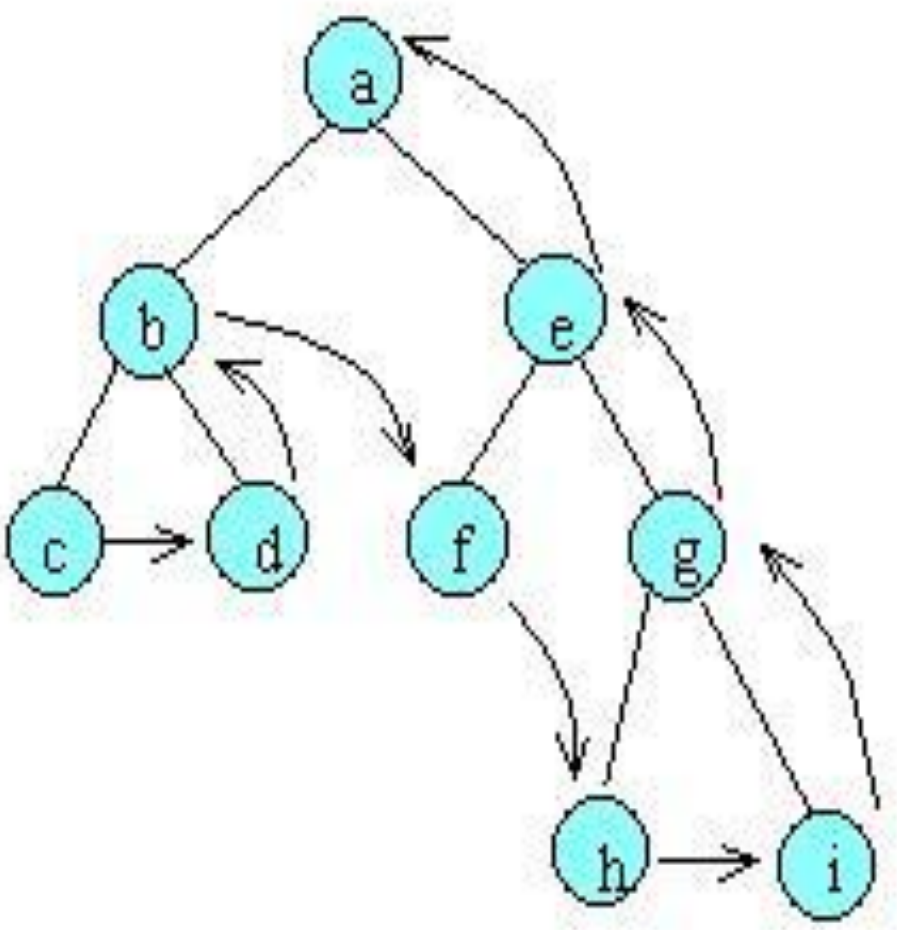
последним посещается корень n;

Прохождение бинарных деревьев

Прохождение бинарного дерева в обратном порядке:

1. **пройти в обратном порядке левое поддерево**
2. **пройти в обратном порядке правое поддерево**
3. **попасть в корень**

Прохождение бинарных деревьев



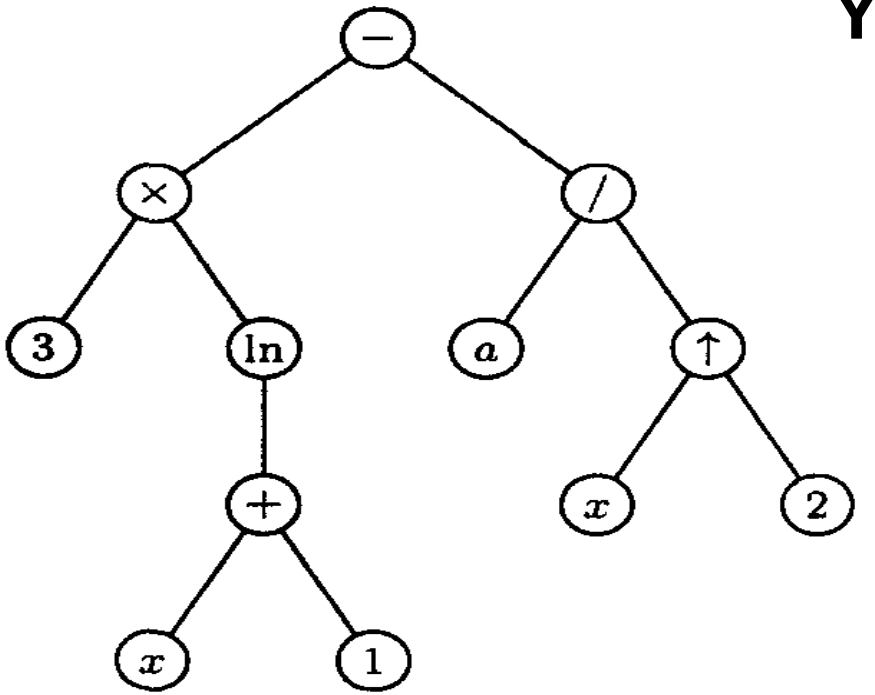
c d b f h i g e a

Прохождение бинарных деревьев

```
procedure  LastOrder(n:   вершина);
{Обход дерева в обратном порядке}
for для каждого потомка s  вершины n в
      порядке слева направо do LastOrder(s);
Занести в список обхода вершину n;
```

Прохождение бинарных деревьев

$Y=3\ln(x+1) - a/x^2$



$- \quad \times \quad 3 \quad \ln \quad + \quad x \quad 1 \quad / \quad a \quad \uparrow \quad x \quad 2$ для прямого порядка;

$3 \quad x \quad 1 \quad + \quad \ln \quad \times \quad a \quad x \quad 2 \quad \uparrow \quad / \quad -$ для обратного порядка.

Прохождение бинарных деревьев

- **при прохождении в симметричном порядке сначала посещаются в симметричном порядке вершины поддерева $Tree_1$,**

  далее корень n,

  затем последовательно в симметричном порядке вершины поддеревьев $Tree_2$ ..., $Tree_k$.

Прохождение бинарных деревьев

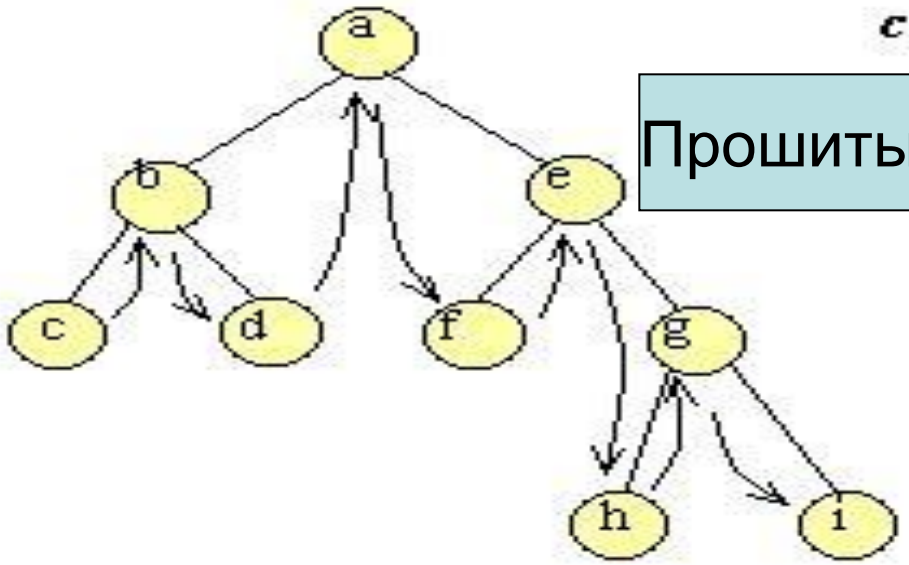
Симметричный порядок:

1. **пройти в симметричном порядке левое поддерево**
2. **попасть в корень**
3. **пройти в симметричном порядке правое поддерево**

Прохождение бинарных деревьев

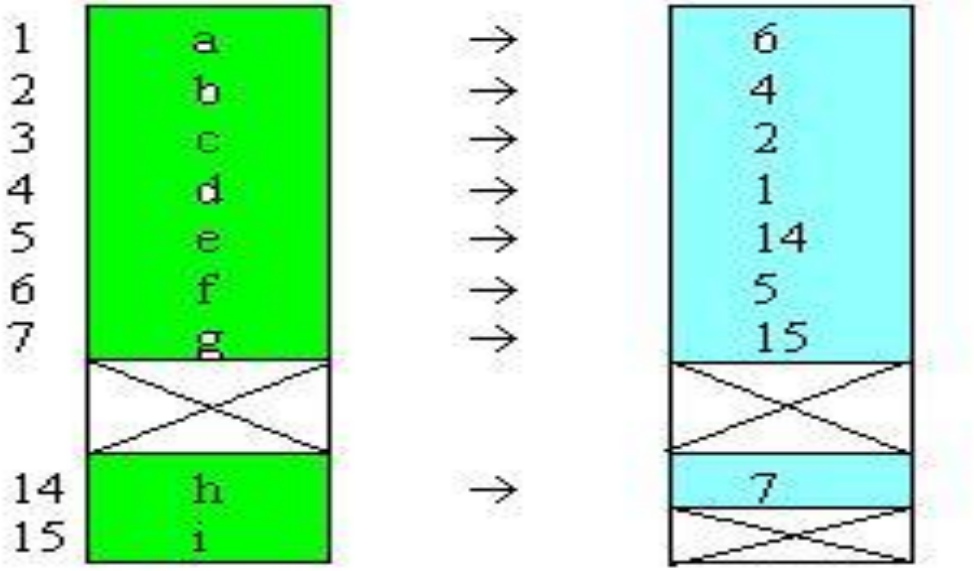
```
procedure  InOrder(n:   вершина);
{Обход дерева в симметричном порядке}

 if n — лист then

        занести в список обхода узел n;

else

        InOrder(самый левый потомок вершины n);
        Занести в список обхода вершину n;
        for для каждого потомка s  вершины n,
                  исключая самый левый,в порядке слева
                 направо do InOrder(s);
```

Прохождение бинарных деревьев



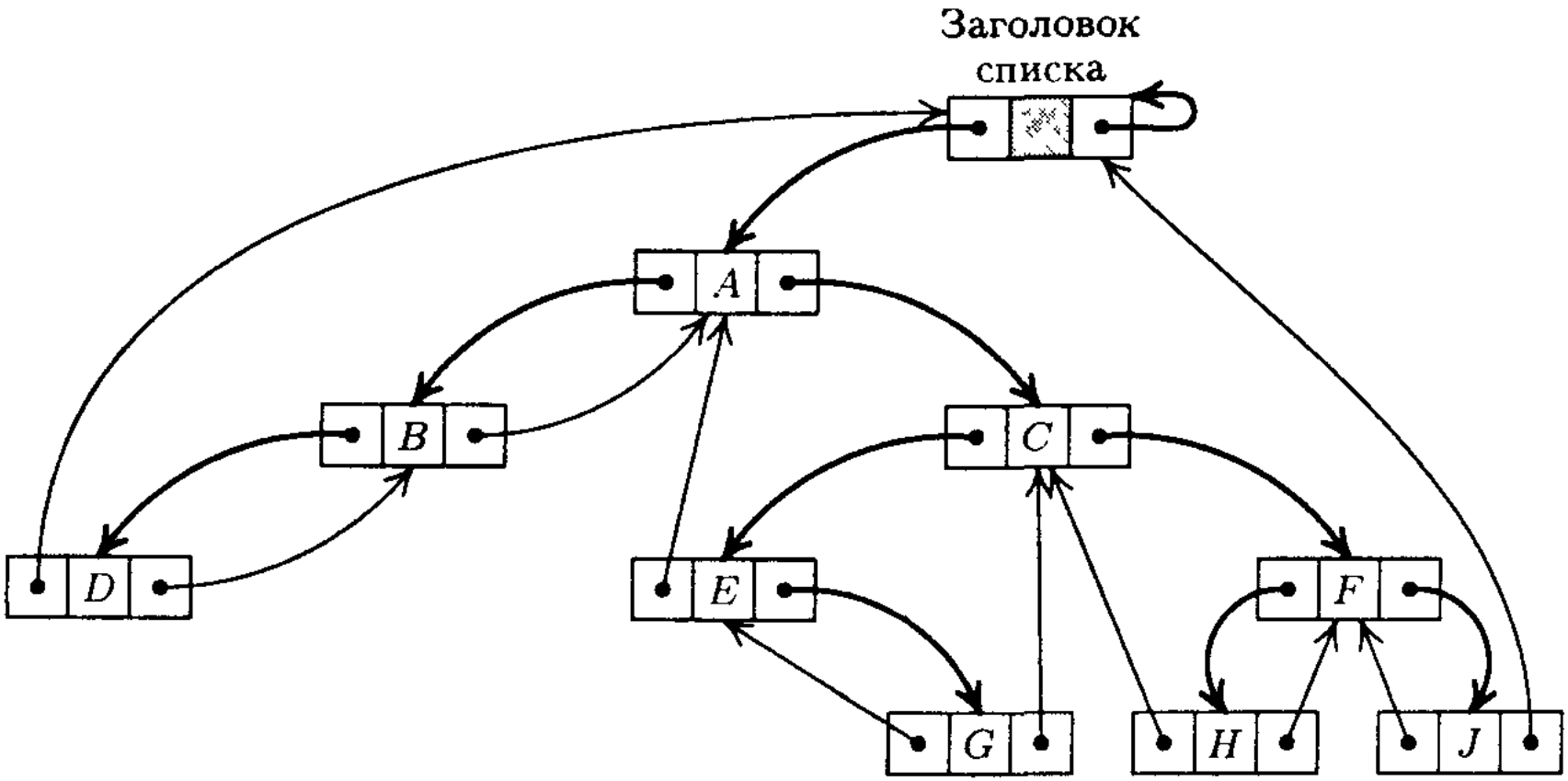
c b d a f e h g i

Прошитые (threaded) деревья



Прохождение бинарных деревьев

Прошитое дерево в динамическом представлении



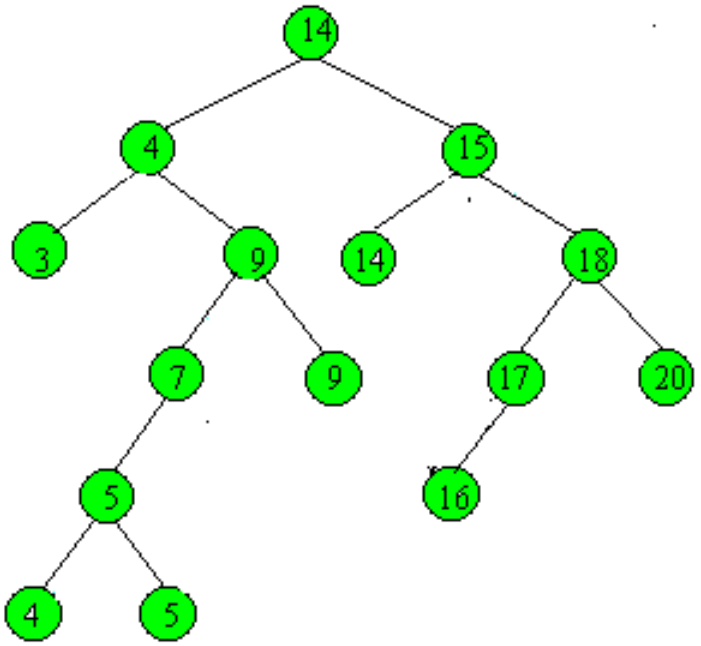
Сортировка с прохождением бинарного дерева

Дерево строится по следующим принципам: в качестве корня создается узел, в который записывается первый элемент массива.

Для каждого очередного элемента создается новый лист. Если элемент меньше значения в текущем узле, то для него выбирается левое поддерево, если больше или равен правое.

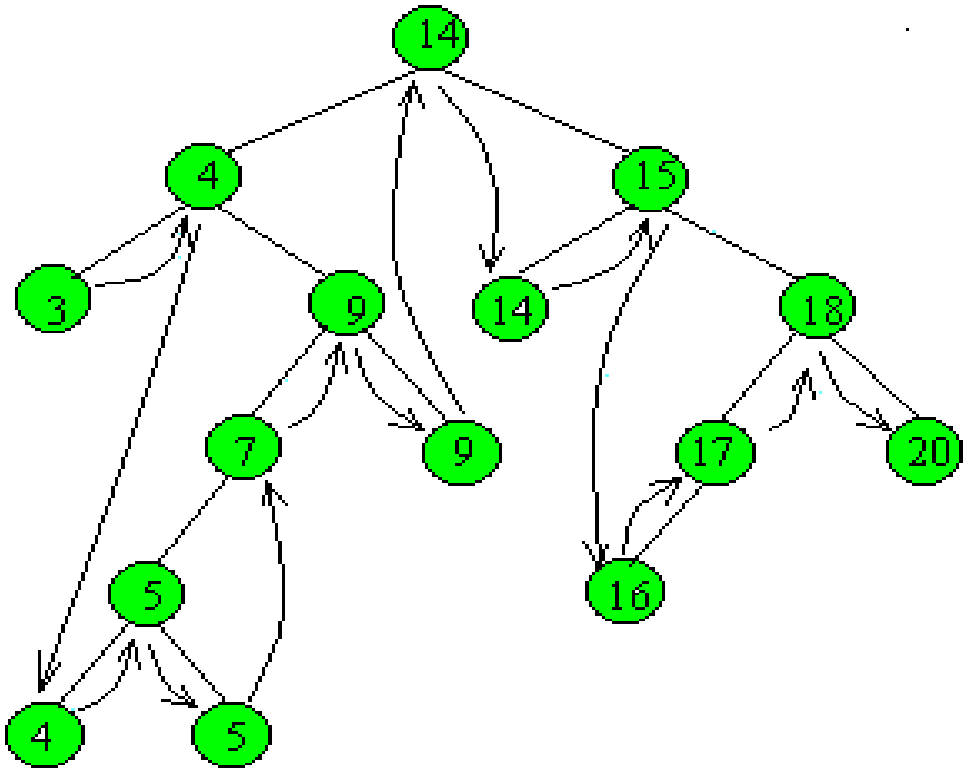
Сортировка с прохождением бинарного

Исходный массив:
14, 15, 4, 9, 7, 18, 3, 5, 17, 4, 20, 16, 9, 14, 5



Сортировка с прохождением бинарного

Исходный массив:
14, 15, 4, 9, 7, 18, 3, 5, 17, 4, 20, 16, 9, 14, 5



Прохождение в симметричном порядке:
3, 4, 4, 5, 5, 7, 9, 9, 14, 14, 15, 16, 17, 18, 20