Внешние сортировки слиянием

Внешние сортировки слиянием

Пусть имеется 5 млн. записей.
Разобьём весь набор на 5 частей, отсортируем их в оперативной памяти, и запишем на носители поочередно.

Лента 1	$R_1 \dots R_{1000000}$; $R_{2000001} \dots R_{3000000}$; $R_{4000001} \dots R_{5000000}$
Лента 2	$R_{1000001} \dots R_{2000000}$; $R_{3000001} \dots R_{4000000}$
Лента 3	(Пустая)
Лента 4	(Пустая)

Выполним первый этап слияния:

Лента 3	$R_1 \dots R_{2000000}$; $R_{4000001} \dots R_{5000000}$
Лента 4	$R_{2000001} \dots R_{4000000}$

Выполним второй этап слияния:

Лента 1	$R_1 \dots R_{4000000}$
Лента 2	$R_{4000001} \dots R_{5000000}$

Сбалансированное слияние (1/2)

1. Распределить начальные серии попеременно на ленты T1 и T2.
2. Выполнить слияние серий с лент T1 и T2 на T3, затем остановиться, если T3 содержит только одну серию.
3. Скопировать серии с T3 попеременно на T1 и T2, затем вернуться к шагу B2.

1) **Лента 1: R1….R1000000, R2000001…R3000000 , R4000001…R5000000**
   Лента 2: R1000001….R2000000, R3000001…R4000000
   Лента 3: (пустая)

2) Лента 1: R1….R1000000, R2000001…R3000000 , R4000001…R5000000
   Лента 2: R1000001….R2000000, R3000001…R4000000
   Лента 3: R1….R2000000, R2000001…R4000000 , R4000001…R5000000

3) **Лента 1: R1….R2000000, R4000001…R5000000**
   Лента 2: R2000001….R4000000
   Лента 3: R1….R2000000, R2000001…R4000000 , R4000001…R5000000

4) Лента 1: R1….R2000000, R4000001…R5000000
   Лента 2: R2000001….R4000000
   Лента 3: R1….R4000000, R4000001…R5000000

Сбалансированное слияние (2/2)

1. Распределить начальные серии попеременно на ленты T1 и T2.
2. Выполнить слияние серий с лент T1 и T2 на T3, затем остановиться, если T3 содержит только одну серию.
3. Скопировать серии с T3 попеременно на T1 и T2, затем вернуться к шагу B2.

4) Лента 1: R1….R2000000, R4000001…R5000000
Лента 2: R2000001….R4000000
Лента 3: R1….R4000000, R4000001…R5000000

5) Лента 1: R1….R4000000
Лента 2: R4000001….R5000000
Лента 3: R1….R4000000, R4000001…R5000000

6) Лента 1: R1….R4000000
Лента 2: R4000001….R5000000
Лента 3: R1…R5000000

Недостаток: шаг 3 – просто копирование без слияний

Двухфазное слияние

A1. Распределить начальные серии попеременно на ленты T1 и T2.

A2. Слить серии с лент T1 и T2 на T3; остановиться, если на T3 содержится только одна серия.

A3. Скопировать *половину* серий T3 на T1.

A4. Слить серии с лент T1 и T3 на T2; остановиться, если на T2 содержится только одна серия.

A5. Скопировать *половину* серий с T2 на T1. Вернуться к шагу A2. ▮

1-A1) **Лента 1: R1….R1000000, R2000001…R3000000 , R4000001…R5000000**
Лента 2: R1000001….R2000000, R3000001…R4000000
Лента 3: (пустая)

2-A2) Лента 1: R1….R1000000, R2000001…R3000000 , R4000001…R5000000
Лента 2: R1000001….R2000000, R3000001…R4000000
Лента 3: R1….R2000000, R2000001…R4000000 , R4000001…R5000000

3-A3) **Лента 1: R1….R2000000**
Лента 2: R1000001….R2000000, R3000001…R4000000
Лента 3: R1….R2000000, **R2000001…R4000000 , R4000001…R5000000**

Двухфазное слияние

A1. Распределить начальные серии попеременно на ленты T1 и T2.

A2. Слить серии с лент T1 и T2 на T3; остановиться, если на T3 содержится только одна серия.

A3. Скопировать *половину* серий T3 на T1.

A4. Слить серии с лент T1 и T3 на T2; остановиться, если на T2 содержится только одна серия.

A5. Скопировать *половину* серий с T2 на T1. Вернуться к шагу A2. ▮

3-A3) **Лента 1: R1….R2000000**
Лента 2: R1000001….R2000000, R3000001…R4000000
Лента 3: R1….R2000000, **R2000001…R4000000 , R4000001…R5000000**

4-A4) Лента 1: R1….R2000000
Лента 2: R1….R4000000, R4000001…R5000000
Лента 3: R1….R2000000, R2000001…R4000000 , R4000001…R5000000

4-A5) Лента 1: **R1….R4000000**
Лента 2: R1….R4000000, **R4000001…R5000000**
Лента 3: R1….R2000000, R2000001…R4000000 , R4000001…R5000000

6-A2) **Лента 3: R1….R5000000**

Программа курса

Алгоритмы поиска:

Последовательный, бинарный поиск и интерполяционный поиск

Алгоритмы сортировки:

Метод простого включения, бинарных вставок, метод Шелла.

Сортировка простым выбором (извлечением).

Обменные сортировки, простая обменная сортировка. шейкерная сортировка.

Быстрая сортировка.

Сортировки слияниями

Деревья:

Бинарные деревья. Сбалансированные деревья. Деревья поиска.

Хэширование:

Основные понятия и принципы. Способы разрешения коллизий

Графы.

Основные понятия и определения. Способы представления графов

Алгоритмы обхода графов: поиск в глубину, поиск в ширину

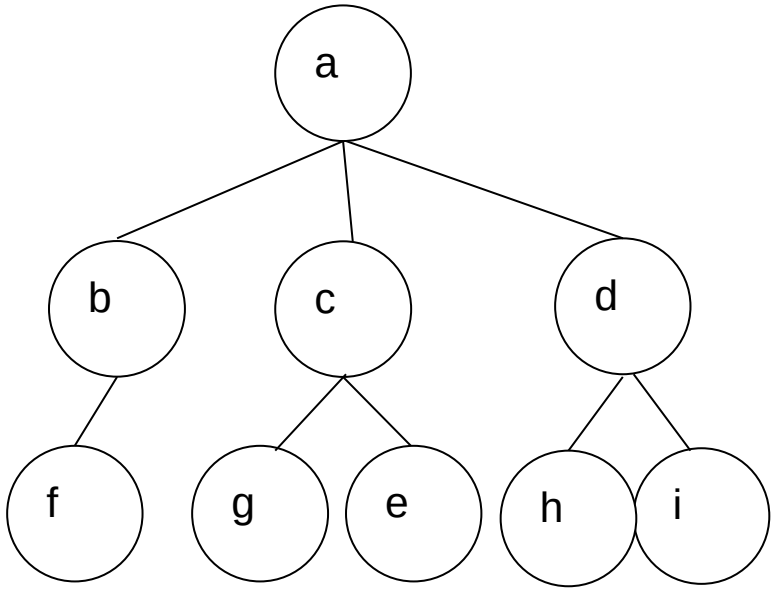
Алгоритмы поиска кратчайших путей: в ориентировнных ациклических графах. алгоритм Беллмана-Форда, алгоритм Дейкстры

Деревья

Определение дерева

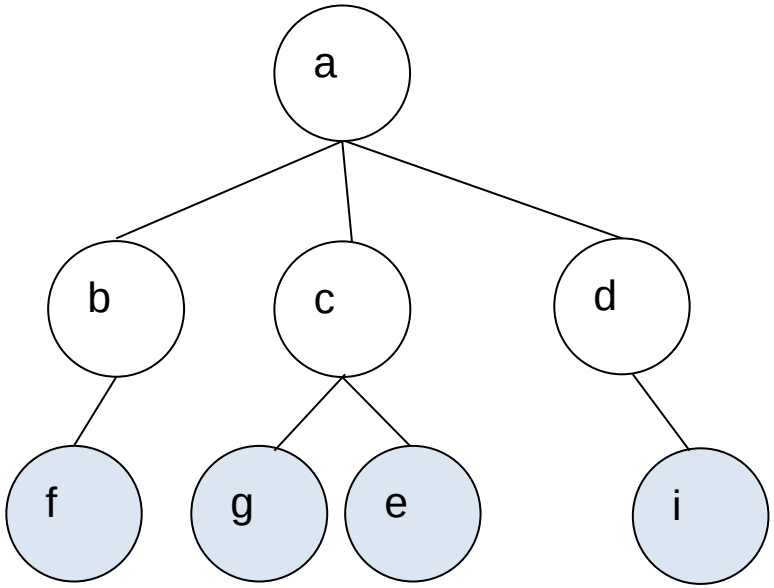
Деревом T называют конечное множество узлов, в котором имеется один выделенный узел t, называемый ***корнем***, а остальные узлы разбиты на $M \geq 0$ непересекающихся множеств $T_1, T_2, \ldots, T_M$, каждое из которых является деревом и называется ***поддеревом*** узла t.

Корень дерева t также называют *предшественником, предком* или *родительским узлом* по отношению к корням $t_1, t_2, \ldots, t_N$ своих поддеревьев $T_1, T_2, \ldots, T_N$.
Узлы $t_1, t_2, \ldots, t_N$ называют п*реемниками, потомками* или *дочерними узлами.*



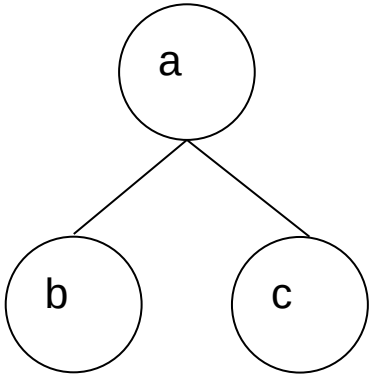
Основные определения

Степенью узла называются число потомков данного узла.
Степенью дерева называют наибольшую из степеней всех его узлов.
Узлы нулевой степени не имеют потомков и называются ***листьями*** (или ***концевыми узлами***) дерева.
Полным называется дерево, у которого степень всех узлов, не являющихся листьями, равна степени дерева.
Уровнем узла называется выраженная в числе узлов длина пути, соединяющая этот узел с корнем дерева.
Высотой дерева называют максимальный уровень для узлов дерева.



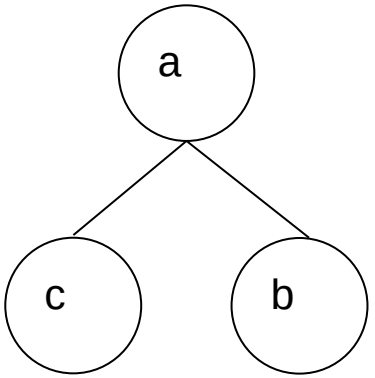
Упорядоченные деревья

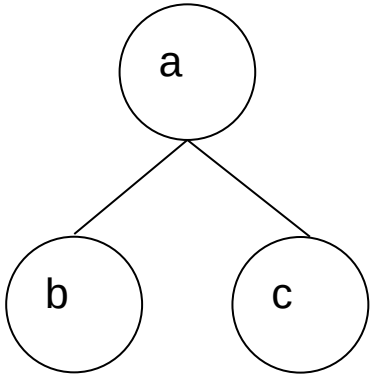
Различают упорядоченные и неупорядоченные деревья. Дерево называют ***упорядоченным***, если для любого узла дерева, за исключением корня, известно, каким по счету потомком является этот узел.



Неупорядоченные деревья

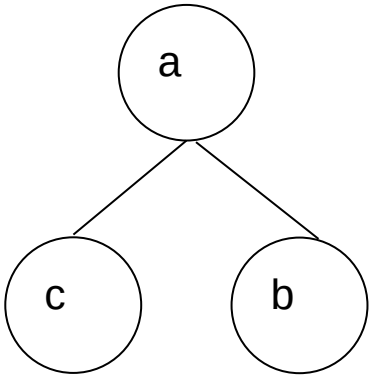
<- равные деревья ->





Упорядоченные деревья

<- разные деревья ->

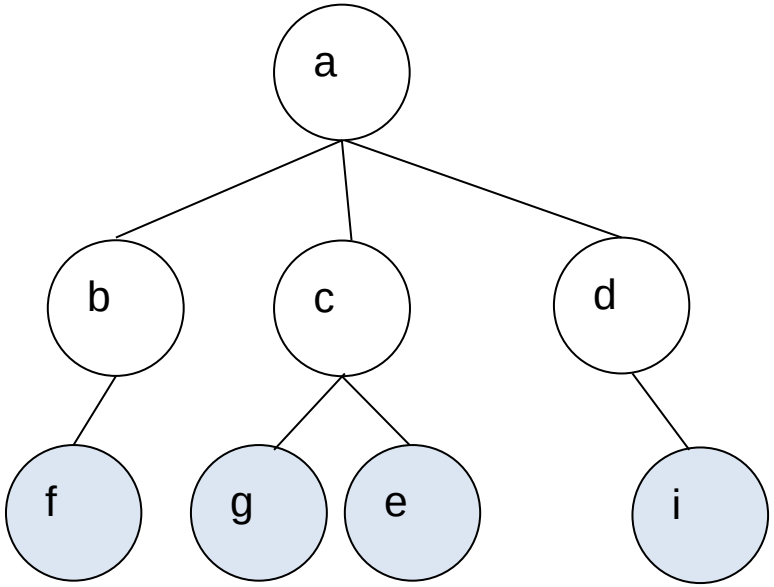


Обход деревьев

Для решения многих задач требуется организовать ***обход*** (прохождение) дерева - просмотр всех узлов дерева в определенном порядке.
Различают два основных алгоритма обхода деревьев: прямой и обратный.

При ***прямом обходе*** для каждого поддерева сначала просматривается корень, а затем все поддеревья данного корня в прямом порядке.

При ***обратном обходе*** сначала в обратном порядке просматривается каждое поддерево, а уже затем - корень. В любом случае при обходе все узлы дерева просматриваются по одному разу, то есть порождается линейная последовательность узлов дерева.

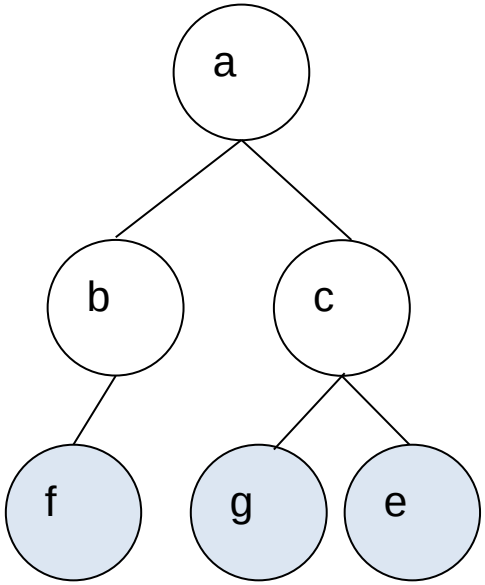


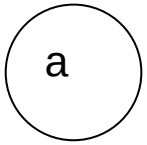
Прямой обход:
a b f c g e d i

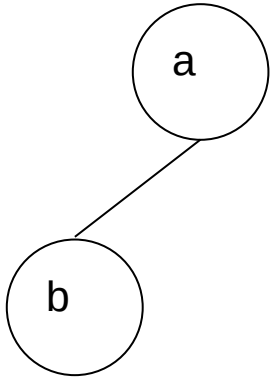
Обратный обход:
f b g e c i d a

Бинарные деревья

Бинарным деревом называют упорядоченное дерево второй степени.
При этом первого потомка любого узла бинарного дерева называют ***левым поддеревом*** этого узла, а второго потока – ***правым поддеревом***.
Узел может не иметь потомков вообще, может иметь только левое или только правое поддерево, может иметь оба поддерева. Бинарное дерево может быть пустым.



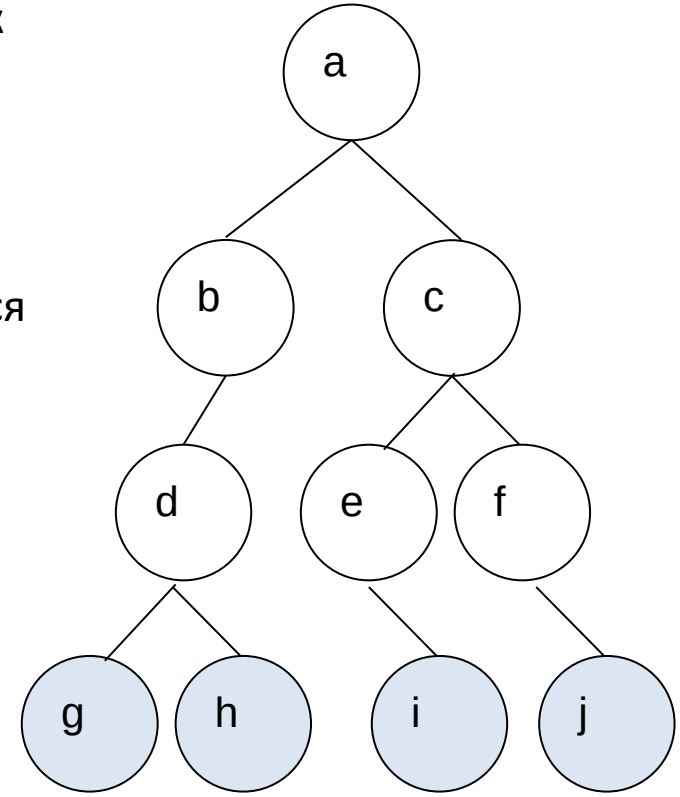




Обход бинарных деревьев (КЛП)

Для бинарных деревьев могут применяться как прямой, так и обратный алгоритмы обхода.
При ***прямом*** алгоритме обхода, называемом также ***нисходящим (КЛП)***, для каждого узла дерева, начиная с корня, сначала обрабатывается сам узел, затем обрабатывается левое поддерево в прямом порядке, а затем - правое поддерево в прямом порядке.

Результат обхода КЛП:
a b d g h c e i f j

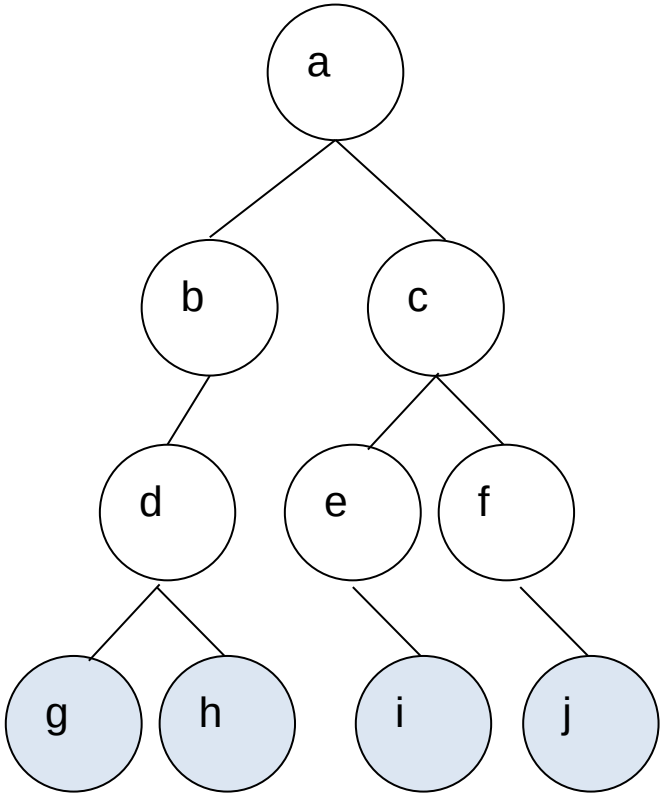


Обход бинарных деревьев (ЛПК)

При ***обратном*** алгоритме обхода, называемом также ***восходящим (ЛПК)***, сначала обрабатывается левое поддерево в обратном порядке, затем обрабатывается правое поддерево в обратном порядке, а уже затем – корень.

Результат обхода ЛПК:

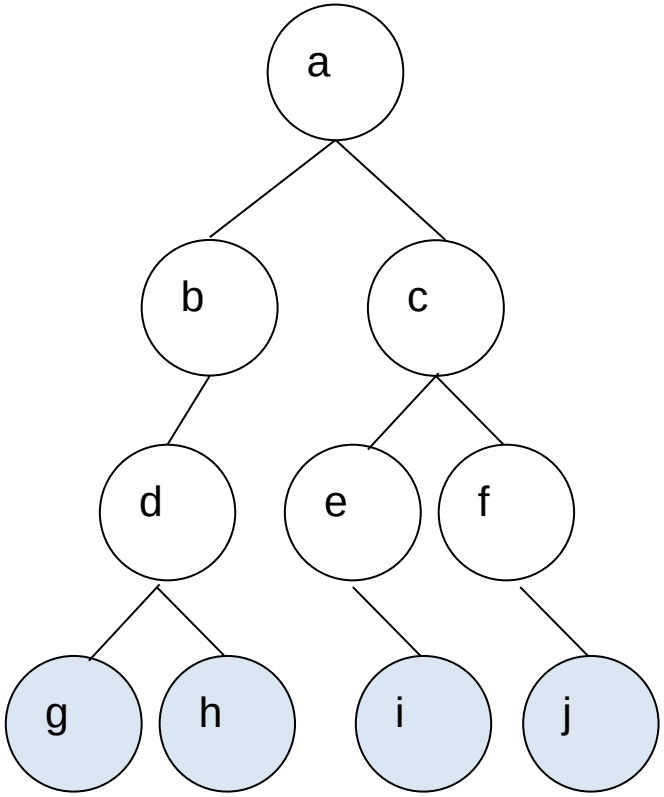
g h d b i e j f c a



Обход бинарных деревьев (ЛКП)

Для бинарных деревьев применим также ***симметричный*** (***смешанный, ЛКП***) алгоритм обхода, при котором для каждого узла дерева, начиная с корня, сначала просматривается левое поддерево в симметричном порядке, затем – сам узел, а затем – правое поддерево в симметричном порядке.

Результат обхода ЛКП:
g d h b a e i c f j



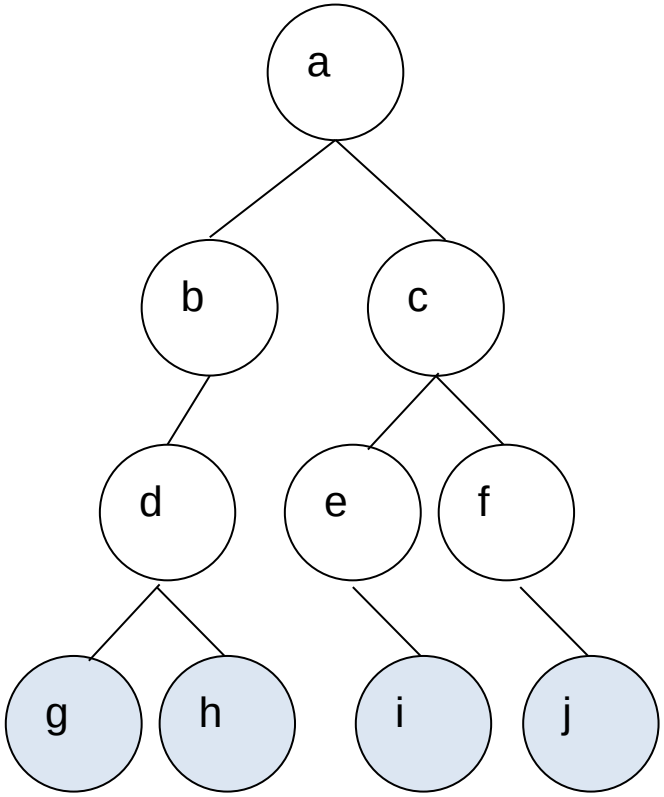
Обход бинарных деревьев (сохранение порядка)

Порядок обход а листьев не зависит от алгоритма обхода, а только от направления обход (левосторонний/правосторонний).

Результат обхода КЛП:
a b d g h c e i f j

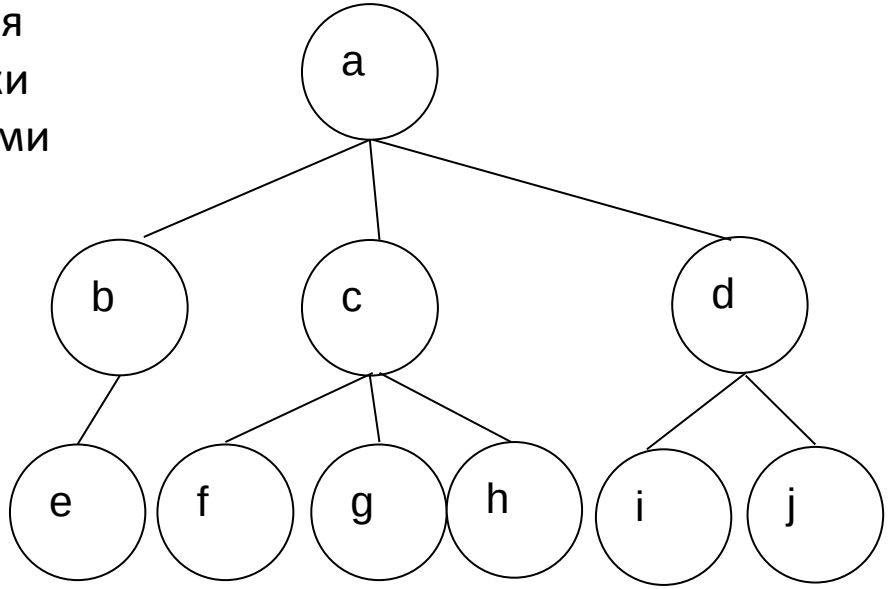
Результат обхода ЛПК:
g h d b i e j f c a

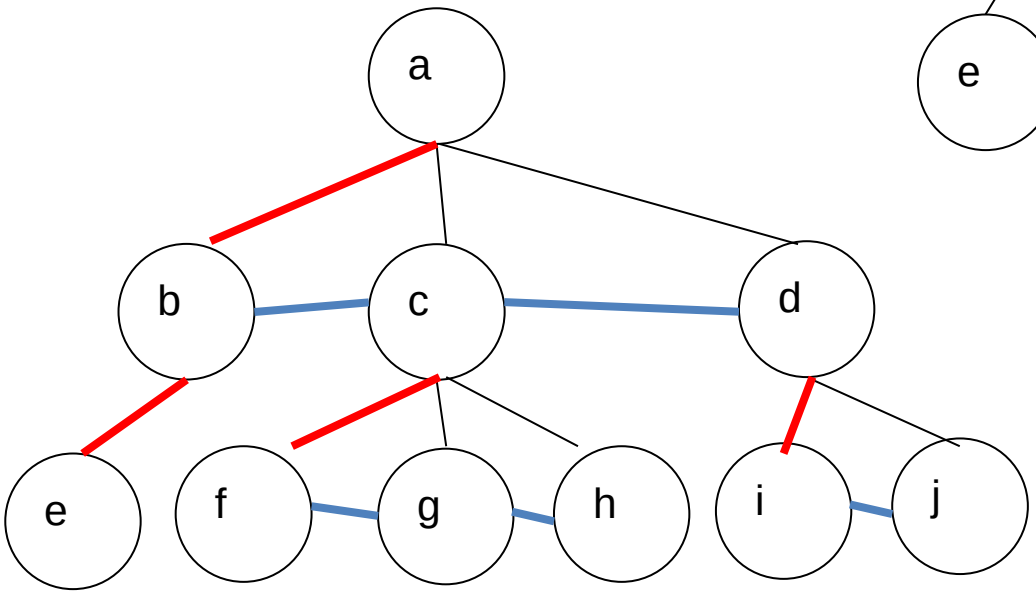
Результат обхода ЛКП:
g d h b a e i c f j



Преобразование дерева общего вида к бинарному

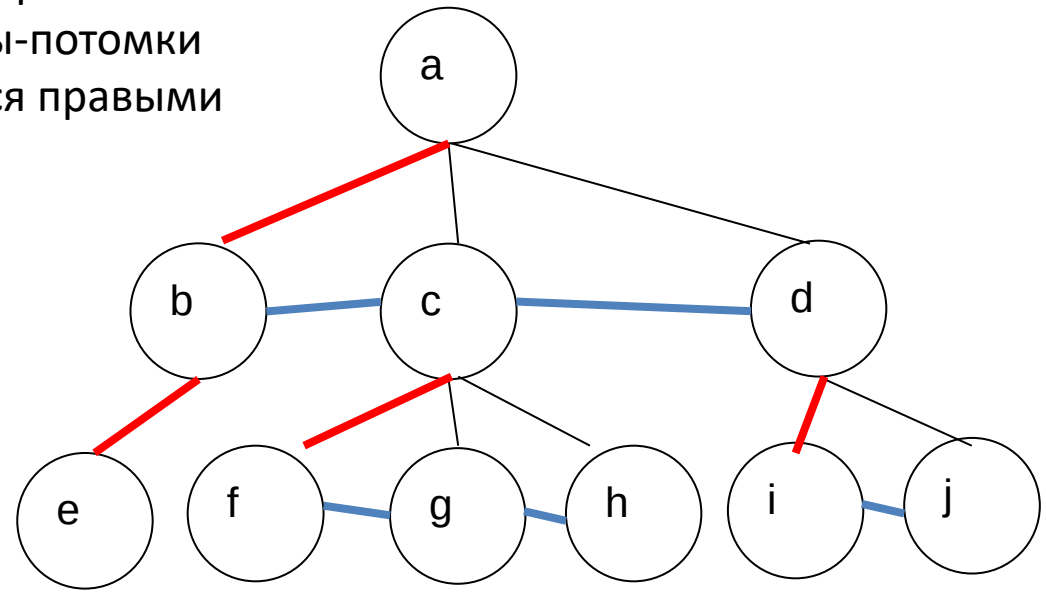
Для узлов дерева общего вида сохраняется самая левая (первая) связь, а узлы-потомки одного и того же узла соединяются правыми связями.

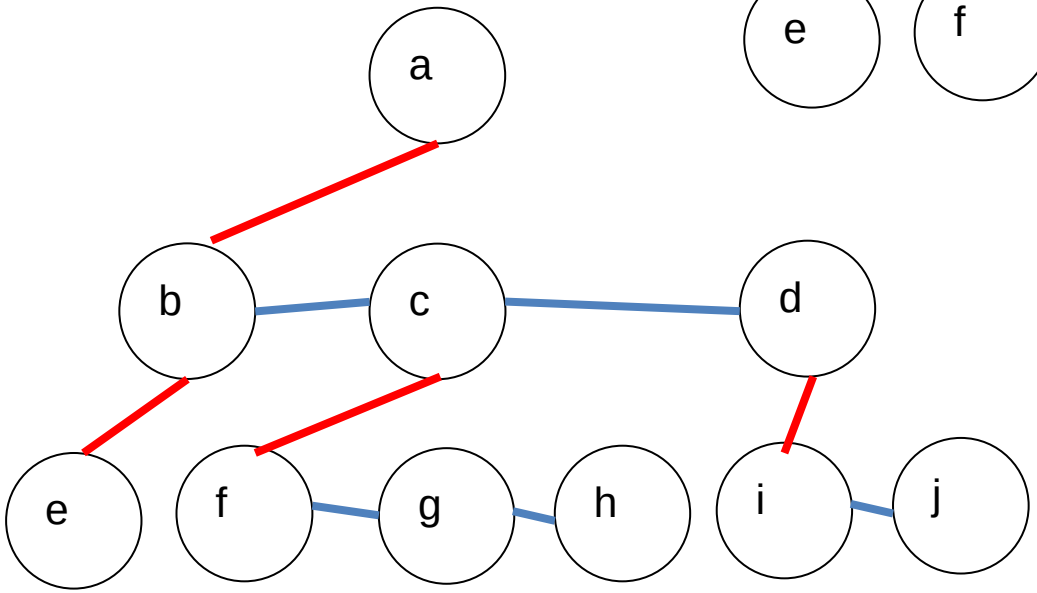




Преобразование дерева общего вида к бинарному

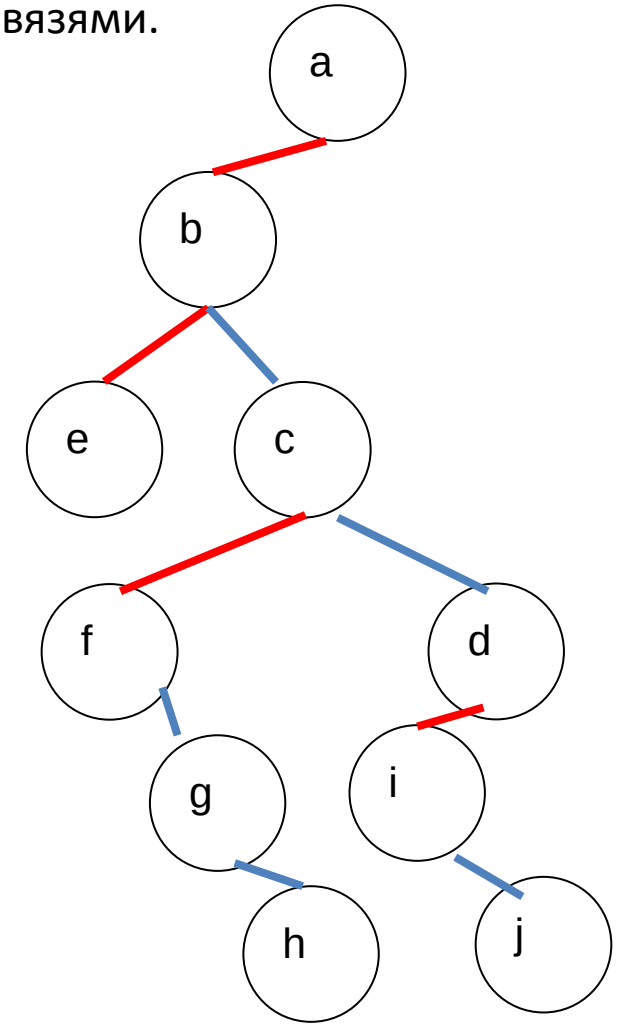
Для узлов дерева общего вида сохраняется самая левая (первая) связь, а узлы-потомки одного и того же узла соединяются правыми связями.

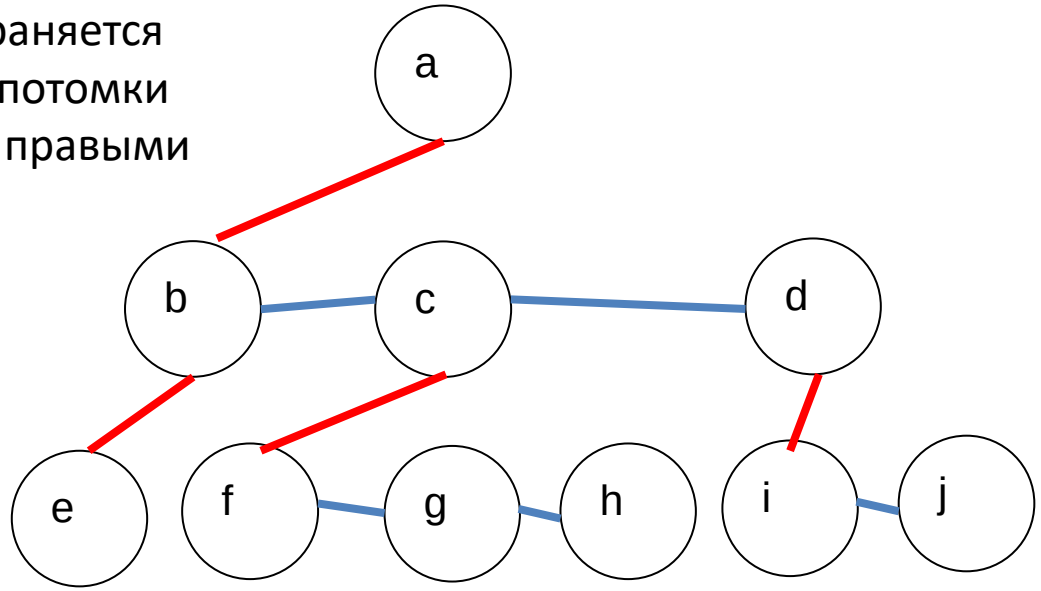




Преобразование дерева общего вида к бинарному

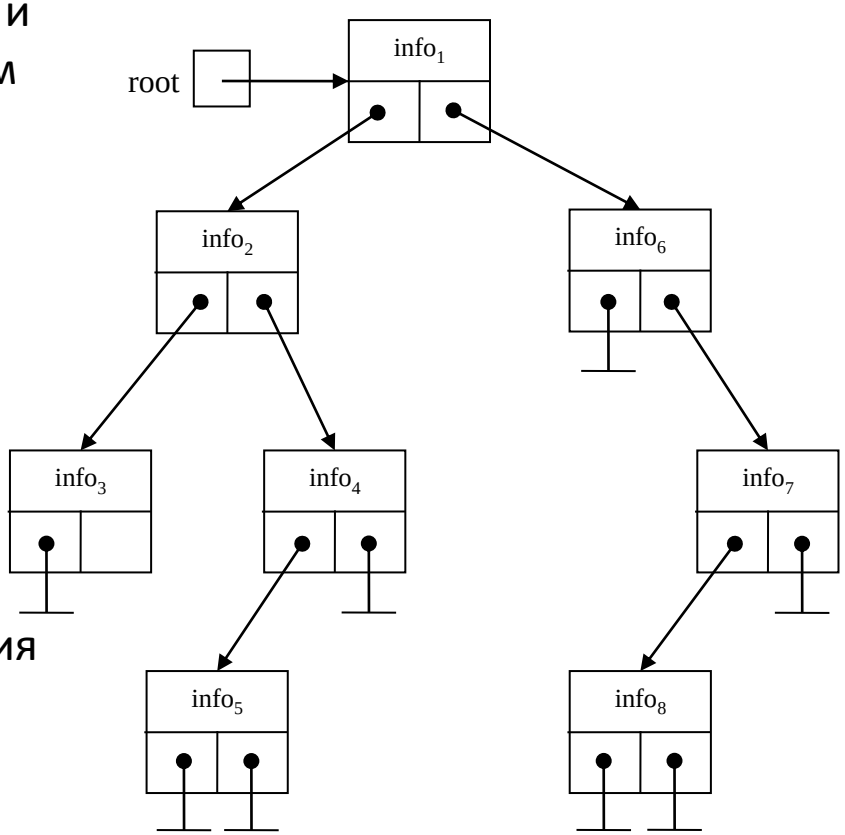
Для узлов дерева общего вида сохраняется самая левая (первая) связь, а узлы-потомки одного и того же узла соединяются правыми связями.





Хранение бинарных деревьев в ОП

Хранение бинарных деревьев может быть организовано, как последовательным, так и связным способом. При последовательном хранении могут быть использованы, как массивы, так и способы, допускающие представление деревьев общего вида, например, уровневое или скобочное представления. Нас, однако, будет интересовать связный способ хранения бинарных деревьев. При таком способе каждый узел бинарного дерева представляет собой структуру, имеющую два поля связи, используемые для хранения указателей на левое и правое поддеревья узла, соответственно.

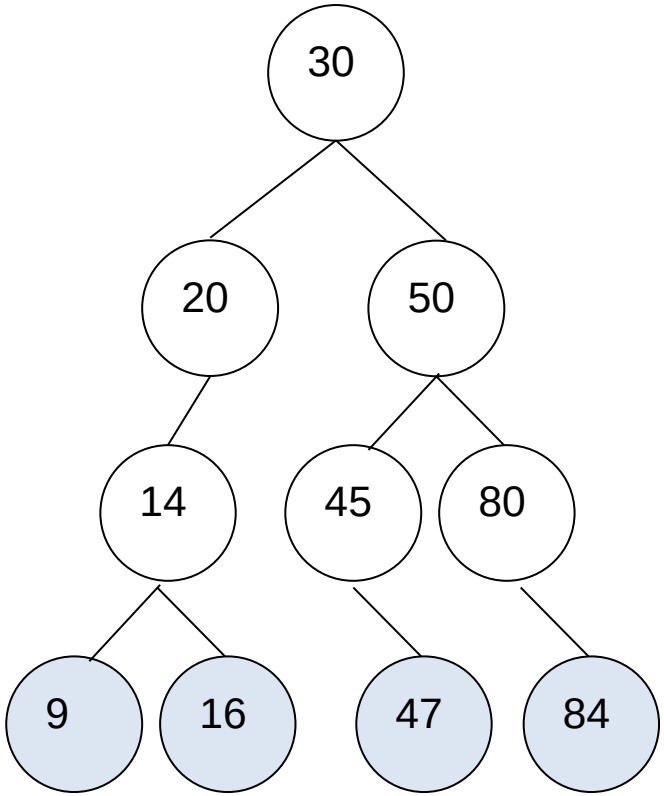



```
struct TreeElm {
   T info;
   TreeElm *left, *right;
};
TreeElm  *root;
```

Деревья поиска

Деревом поиска (***дихотомическим*** деревом) называют такое дерево, для любого узла t которого:
- значения ключевых полей всех узлов левого поддерева узла t меньше, чем у узла t,
- значения ключевых полей всех узлов правого поддерева узла t больше, чем у узла t,

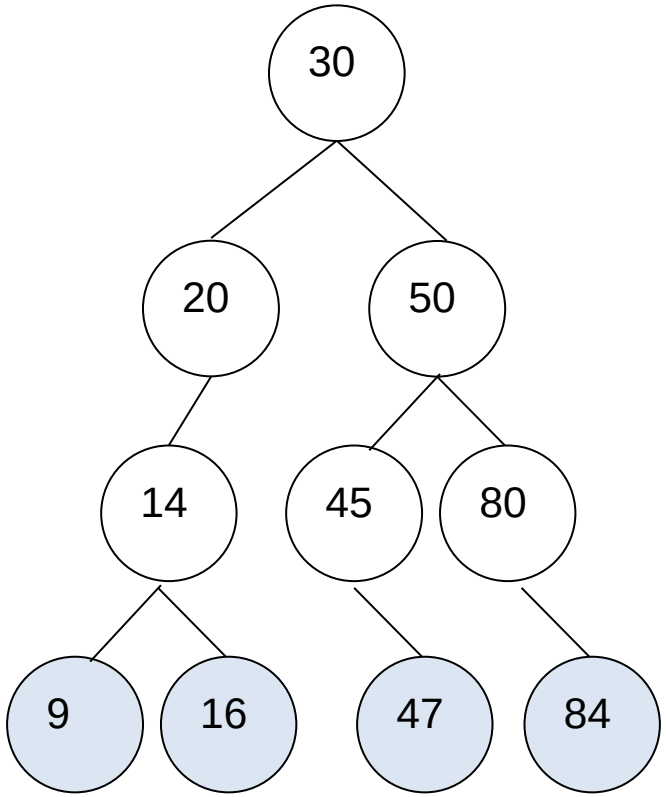
Дерево поиска позволяет быстро находить узел дерева с заданным значением ключа.
Поиск в дереве поиска начинают с корня дерева и на каждом шаге сравнивают искомое значение с ключевым полем рассматриваемого узла. В том случае, если значения совпадают, узел найден. В том случае, если искомое значение меньше значения ключевого поля, то поиск продолжают в том же порядке в левом поддереве узла, иначе – в правом поддереве. В том случае, если при поиске осуществляется переход к пустому поддереву, констатируют, что значение не найдено.



Деревья поиска

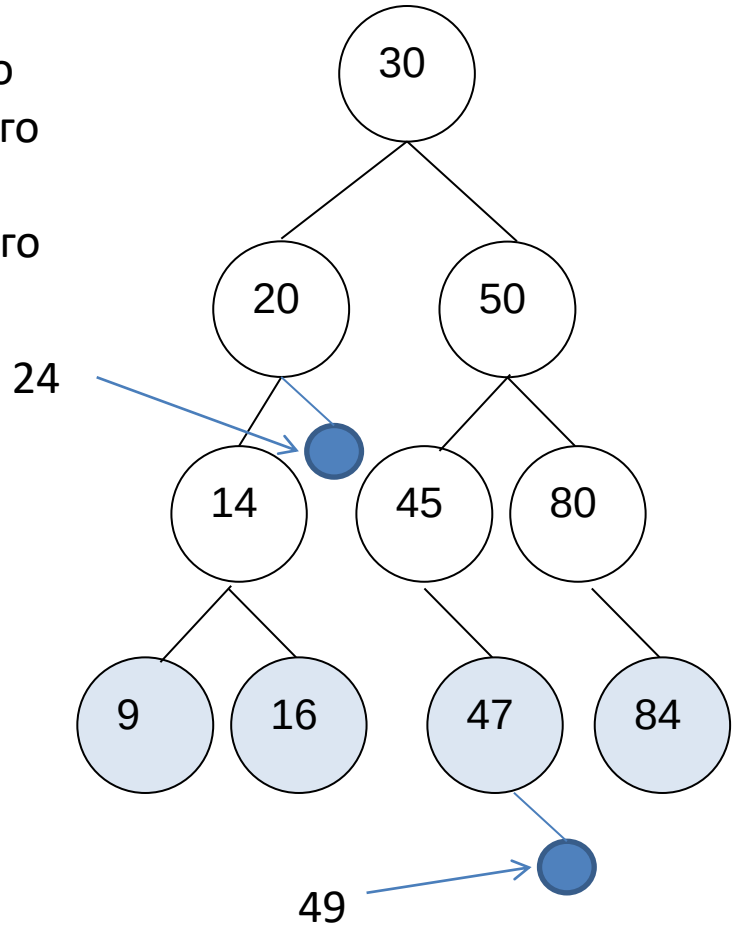
Ключевые поля узлов дерева поиска, распечатанные в симметричном порядке, образуют возрастающую последовательность.

ЛКП:
9 14 16 20 30 45 47 50 80 84



Добавление узлов

В тех случаях, когда к дереву поиска не предъявляется требований по сбалансированности, добавляемый в дерево поиска узел помещается на место того пустого поддерева, к которому привел поиск узла (поиск по ключевому значению добавляемого узла).



Удаление узлов

Удаление узла из дерева поиска осуществляется следующим образом.

1) Если удаляемый узел является листом дерева, то удаляемый узел замещается пустым поддеревом.
2) Если удаляемый узел имеет одного потомка, то удаляемый узел замещается своим потомком.
3) Если удаляемый узел имеет двух потомков, то в удаляемый узел помещаются данные из узла, непосредственно предшествующего удаляемому узлу по значению ключевого поля (самый правый узел левого поддерева), либо данные из узла, непосредственно следующего за удаляемым узлом по значению ключевого поля (самый левый узел правого поддерева). После этого узел, из которого пересылались данные, удаляется. Удаление такого узла выполняется несложно, так как такой узел будет иметь не более одного потомка.