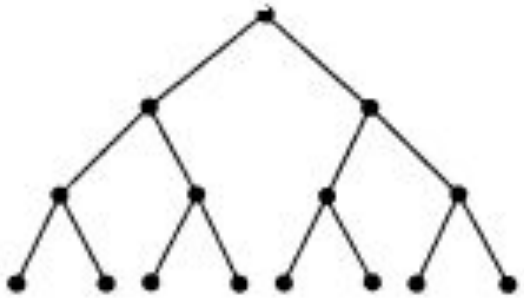
Деревья

Лектор: Завьялов Олег Геннадьевич
кандидат физико-математических наук, доцент

“*Нет, не увижу, это ясно, поэмы дерева прекрасней*”
Джойс Килмер (Joyce Kilmer, 1886 – 1918)

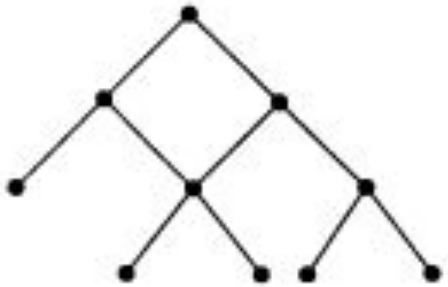
Дерево - граф без циклов.

Лес – граф, компоненты которого являются деревьями.

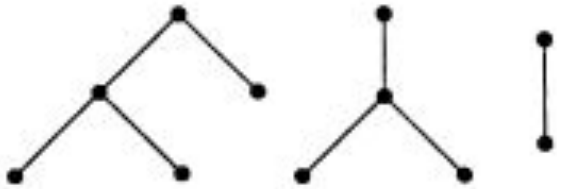


Дерево можно использовать для демонстрации результатов трехкратного подбрасывания монеты.

Не является деревом (содержит цикл):



Лес:



Пример
Дерево для университета

Определение

Ориентированное дерево T – свободный от петель ориентированный граф, соотнесенный граф которого является деревом. Если существует путь от вершины a к вершине b, то он единственный.

Если в ориентированном дереве имеется ребро (a, b), тогда не существует ребро (b, a), в противном случае путь aba был бы циклом, путь из a и b не был бы единственным.

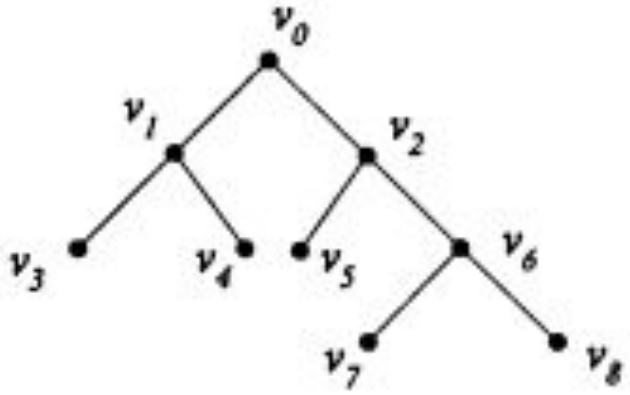
Множество E, которое для дерева представляет собой как конечное множество ребер так и отношение, обладает таким свойством, что если $(a, b) \in$ E, то $(b, a) \notin$ E.

Такое отошение называется ***асимметричным***.

Если неориентированное дерево имеет хотя бы одно ребро, оно имеет хотя бы две вершины со степенью 1.
Вершины степени 1 называются ***листьями*** (начинается в вершине a и заканчивается в вершине b).
Другие вершины называются ***внутренними вершинами***.
Максимальными путь не обязательно совпадает с самым длинным путем в дереве.

Пример.

Путь $v_0\ v_2\ v_5$ является максимальным путем.
Вершины v_3 , v_4 , v_5 , v_7 являются листьями.



Свойство деревьев

Теорема.
Для любых двух вершин a и b дерева T существует единственный путь из a и b.

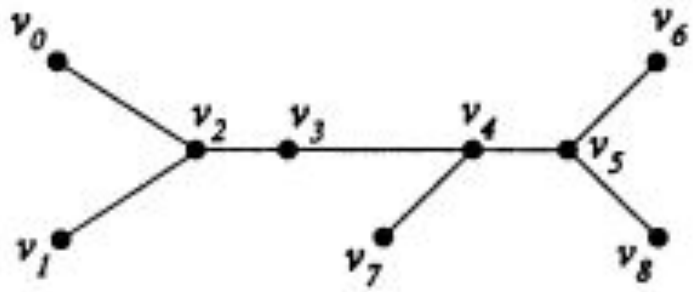
Доказательство.
Предположим, что для некоторых вершин a и b дерева T путь из a в b не является единственным, тогда T не является деревом:
Допустим существуют два различных пути $v_0 v_1 v_2 \ldots v_n$ длины n и $v`_0 v`_1 v`_2 \ldots v`_n$ длины m, где $a = v_0$ и $v_n = v`_m$.
В каждом пути должна существовать первая вершина, начиная с которой соответствующие вершины не совпадают, например $v_i \neq v`_l$ и в каждом из путей должна существовать точка, начиная с которой вершины опять одни и те же $v_j = v`_k$.
Тогда $v_{i-1}\, v_i\, v_{i+1}\, v_j\, v`_{k-1}\, v`_{k-2}\, v`_i\, v`_{i-1}$ является циклом ⇒ граф T не является деревом. Противоречие доказывает теорему.
Верна также обратная теорема.

Теорема.

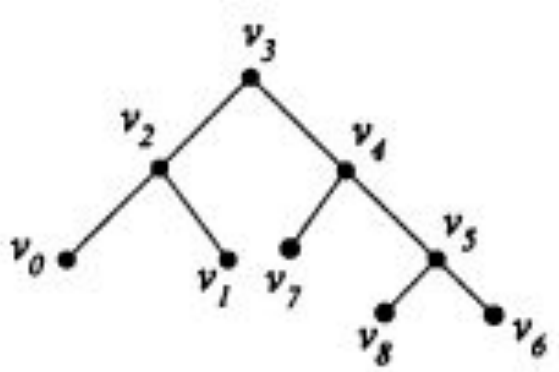
Если для любых двух вершин графа G существует единственный путь из вершины a в вершину b, тогда G – дерево.

Доказательство:
Предположим G не является деревом.
Тогда либо G не является связным, либо содержит цикл.
Если граф G не связный, то существуют вершины $a, b \in G$, для которых не существует пути из a в b.
Если G содержит цикл $v_0 v_1 v_2 \ldots v_{k-1} v_k$, то $v_1 v_2 \ldots v_{k-1} v_k v_0$ и $v_2 v_1 v_0$ являются путями из v_2 в v_0.
Полагая $a = v_2$ и $b = v_0 \Rightarrow$ путь между вершинами a и b не является единственным.
Противоречие доказывает теорему.

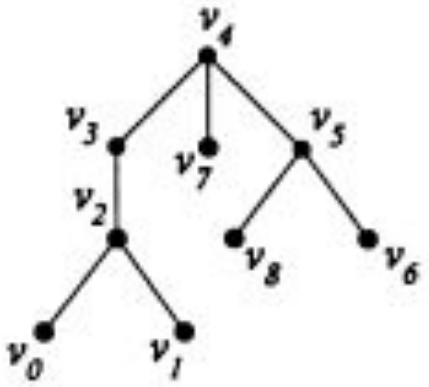
Пусть дерево представляет физический объект, подвижный в вершинах.
Подвешенное дерево за одну из вершин, остальная часть повиснет ниже



Подвешенное за вершину v_3

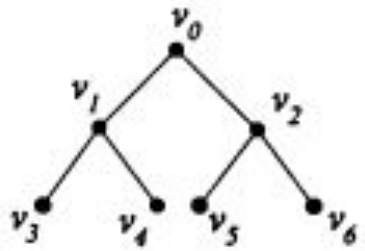


Подвешенное за вершину v_4

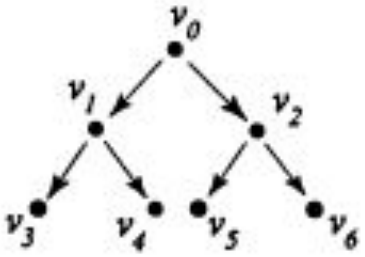


Определения

Вершина в самой верхней части каждого изображения называется ***корнем дерева***.
Если корень дерева определен, дерево называется ***корневым деревом.***
При необходимости можно заменить корневое дерево T на ориентированное T`, дерево



будет заменено деревом



Определения

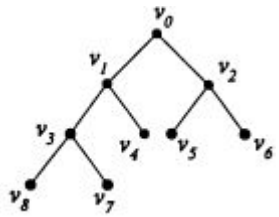
Такое дерево называется ***корневым ориентированным*** деревом. T` - порожденным деревом T.
Если корень выбран, уровень вершины *v* определяется длиной единственного пути из корня в вершину *v*.
Высотой дерева называется длина самого длинного пути от корня дерева до листа.
Если рассматривается корневое ориентированное дерево T `, порожденное данным корневым деревом T, тогда вершина *u* называется ***родителем*** вершины *v* , а *v* называется ***сыном*** вершины *u* , если существует ориентированное ребро из *u* и *v*.
Если *u* - родитель *v* и *v`*, тогда *v* и *v`* называются ***братьями***.
Если существует ориентированный путь из вершины *u* в вершину *v*, тогда *u* называется ***предком*** вершины *v*, а *v* называется ***потомком*** вершины *u*.

Определения

Если наибольшая из степеней выхода для вершин дерева равна *m*, тогда дерево называется *m* –***арным деревом***.
В частном случае (*m* = 2) дерево называется бинарным деревом.
В каждом бинарном дереве каждый сын родителя обозначается как ***левый сын*** или как ***правый сын*** (но не одновременно).

Пример

Граф – бинарное дерево.



Уровень вершины v_6 равен 2, уровень вершины v_8 равен 3.
Высота дерева равна 3 и не существует более длинного пути от корня к листу.
Вершина v_1 является родителем для v_3 и v_4.
Вершина v_3 и v_4, v_1 и v_2, v_5 и v_6, v_7 и v_8, - братья.
Вершина v_8 – левый сын вершины v_3,
v_4 – левый сын вершины v_1.

Теорема.

Если у дерева T имеется e ребер и v вершин, тогда $v = e + 1$.
Доказательство:
Рассмотрим ориентированное дерево T`, порожденное деревом T. У каждого ребра T одна и только одна конечная вершина. ⇒ Число ребер и вершин одно и то же, исключая коневую вершину.
Если учесть ориентированную вершину ⇒ вершин на одну больше, чем ребер. Ч.т.д.

Теорема.

Если G содержит цикл, если ребро $\{v_i, v_j\}$ входит в цикл ⇒ два пути из v_i в v_j. Т.е. ребро $\{v_i, v_j\}$ можно из цикла удалить, а путь из вершины v_i в v_j будет существовать.
Пусть a и b - любые точки в G.
G - связный граф ⇒ существует путь из a в b.
Если ребро $\{v_i, v_j\}$ удалено ⇒ существует путь из a в b, ребро $\{v_i, v_j\}$ можно заменить альтернативным путем из v_i в v_j.
Продолжают, пока все циклы не будут удалены. Получают связный граф G` без циклов.
G` - дерево, число вершин $v = e` + 1$, $e`$ - число ребер графа.
G`.
Ни одна вершина не выла удалена. Удалено n ребер,
тогда $e = e` + n$. $v = e + 1$ и $v = e` + 1$ ⇒ $e = e`$ и $n = 0$. ⇒ ни одно ребро не было удалено. ⇒ G - дерево.

Определения

Дерево G`, построенное из G в процессе доказательства остается, называется ***остовным*** (***каркасным***) деревом графа G.
Дерево T называется остовным деревом графа G, если T – подграф графа G и каждая вершина в G является вершиной в T.

Теорема.
У каждого связного графа существует подграф, который является остовным деревом.

Свойства деревьев

Теорема.

Следующие утверждения эквивалентны

А) Граф G - дерево.

Б) Граф G – связный и $v = e + 1$, v – количество вершин, e – количество ребер графа G.

В) Для каждой пары различных вершин a и b существует единственный путь из a в b .

Г) Граф G – ацикличный (не имеет циклов) и $v = e + 1$.

Определения.

Ориентированное *T*-дерево это ориентированный граф без петель, соотнесенный граф которого является деревом, так что если существует путь из вершины *a* в вершину *b*, то он единственный.

Ориентированное дерево *T* является ***корневым ориентированным деревом***, если существует единственная вершина v_0 такая, что существует путь из вершины v_0 в каждую другую вершину дерева *T*.

Теорема.

Для ориентированного дерева G следующие утверждения эквивалентны.

А) G – корневое ориентированное дерево.

Б) G имеет единственный такой элемент v_0, что для любой вершины a графа G существует единственный ориентированный путь из v_0 в a.

В) Соотнесенный граф графа G связен, и G содержит единственный элемент v' такой, что для любой другой вершины a графа G существует единственный путь из v_0 в a.

Рассматриваются только корневые ориентированные деревья.
Определения.

В ориентированном дереве ***уровень*** вершины *v* - это длина пути от корня до вершины *v*. ***Высота*** ориентированного дерева - это длина самого длинного пути от корня до листа.

m –арным ориентированным деревом называется такое дерево, в котором родитель имеет не более *m* сыновей.

Полным *m* –арным ориентированным деревом называется такое ориентированное дерево, в котором каждый родитель имеет в точности *m* сыновей.

m –арным ориентированным деревом высоты h называется ***сбалансированным*** (***полным***, или ***почти полным***), если уровень каждого листа равен *h* или *h* – 1.

Теорема.

Если полное m –арное ориентированное дерево имеет n вершин и i внутренних вершин, то $n = m\,i + 1$.

$$i = (n - 1)/\,m$$

Теорема.

Если полное m –арное ориентированное дерево имеет n вершин, i внутренних вершин и l листьев, то $l = (m - 1)\,i + 1$.

$$i = (l - 1)/(m - 1)$$

Теорема.

Полное m –арное ориентированное дерево высоты h имеет $(m^{h+1} - 1)/\,(m - 1)$ вершин и m^h листьев.

В частности, полное бинарное ориентированное дерево высоты h имеет $2^{h+1} - 1$ вершин и 2^h листьев.

Теорема.

а) Если полное m –арное дерево высоты h имеет l листьев, то $h = \log_m(l)$.

б) Если m –арное дерево имеет l листьев, то $h \geq \log_m(l)$.

в) Если полное бинарное дерево высоты h имеет v вершин, то $h = \log_2(v + 1) - 1$.

г) Если бинарное дерево высоты h имеет v вершин, то $h \geq \log_2(v + 1) - 1$.

Определение.

Функция f из графа $G(V,E)$ в граф $G'(V',E')$ называется ***гомоморфизмом*** из G в G', обозначается $f : G \to G'$, если она обладает следующими свойствами :

а) Если $e \in E$, то $f(e) \in E'$ ($f(E) \subseteq E'$).

б) Если $v \in V$, то $f(v) \in V'$ ($f(V) \subseteq V'$).

в) Если вершины u и v инцидентны ребру e в G, то $f(u)$ и $f(v)$ инцидентны ребру $f(e)$ в G'.

Определение.

а) Если $e \in E$, то $f(e) \in E'$ ($f(E) \subseteq E'$).

б) Если $v \in V$, то $f(v) \in V'$ ($f(V) \subseteq V'$).

в) Если вершины u и v инцидентны ребру e в G, то $f(u)$ и $f(v)$ инцидентны ребру f(e) в G'.

Определение.

Гомоморфизм $f : G \to G'$ является ***изоморфизмом***, если $f : V \to V'$ и $f : E \to E'$ являются взаимно однозначными соответствиями.

Если $f : G \to G'$ изоморфизм, то G и G' ***изоморфны***.

Определение.

Два корневых бинарных дерева $T(E,V)$ и $T'(E', V')$ изоморфны, если существует изоморфизм f из T в T' такой, что

а) v_i - левый сын вершины v_j тогда и только тогда, когда $f(v_i)$ - левый сын вершины $f(v_j)$.

б) v_i - правый сын вершины v_j тогда и только тогда, когда $f(v_i)$ - правый сын вершины $f(v_j)$.

в) f отображает корень r дерева T в корень r' дерева T'.

Теорема.

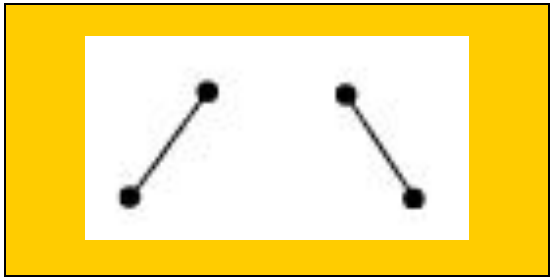
Число неизоморфных корневых бинарных деревьев с n вершинами равна числу Каталана C_n.

Доказательство:

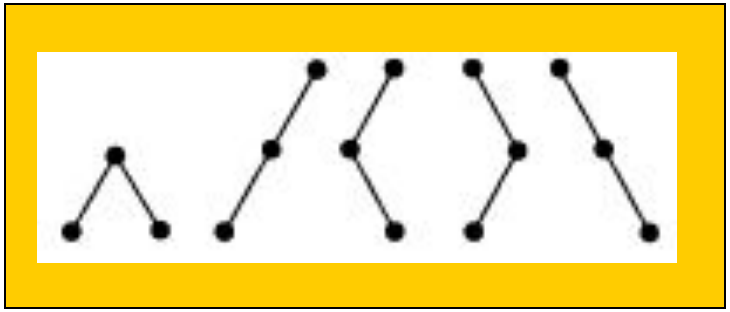
Пусть l_n – число изоморфных корневых бинарных деревьев с n вершинами. Если n=0, дерево пустое и $l_0 = 1$.

Если n=1, одна вершина, одно дерево и $l_1 = 1$.

Если n=2, два корневых бинарных дерева.



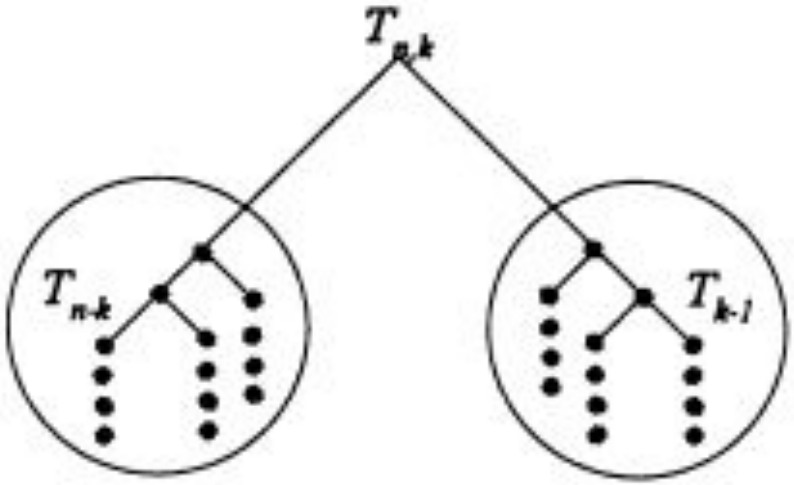
Если n=3, пять корневых бинарных дерева.



Если $n > 3$, выбираем k, что $1 \le k \le n$.

Пусть T_n обозначает дерево с n вершинами и пусть $T_{n,k}$ - дерево с n вершинами, определенное следующим образом.

Одна вершина является конем, пусть имеется правое поддерево T_{n-1} с $k - 1$ вершиной и левое поддерево T_{n-k} с $n - k$ вершинами.



По определению число способов, которыми можно построить дерево T_{k-1}, а число способов, которыми можно построить дерево T_{n-k}, равно I_{n-k}.

Т.е. число способов построения $T_{n,k}$ равно $I_{k-1} \cdot I_{n-k}$.

Суммируя по k, получается число всевозможных способов построения дерева T_n.

$$I_n = \sum_{k=1}^{n} I_{k-1} \cdot I_{n-k}$$

Это совпадает с определением числа Каталана C_n.

$$I_n = C_n = \frac{1}{n+1} \cdot \frac{(2n)!}{n! \cdot n!}$$

Бинарные деревья поиска

Бинарное корневое дерево (просто бинарное дерево) обеспечивает метод организации данных, при котором любые конкретные данные можно легко найти или установить их отсутствие.

Первый способ – просмотр всех данных. Не эффективен.

Второй способ – бинарное дерево. Требование – введение на данных некоторого линейного порядка (алфавитный или числовой) . Линейный порядок может быть установлен в файле, указателе или некотором другом ключе.

Определение.

Бинарные деревья поиска – это прежде всего бинарное дерево, в каждом узле которого находится имя (или другой ключ).

Алгоритм вставки имени в дерево поиска, за исключением размещения имени в корне дерева.

Применим для любого упорядочения.

Вставка(элемент)

(1) Начинаем с корня.
(2) Если элемент < объекта в вершине, переходим к левому сыну.
(3) Если элемент > объекта в вершине, переходим к правому сыну.
(4) Повторяем шаги 2 и 3, пока не достигнем вершины, которая не определена.
(5) Если достигнутая вершина не определена, то определяем вершину и вставляем элемент.

Алгоритм поиска имени в дереве списка.

Поиск(элемент)

(1) Начинаем с корня.

(2) Если элемент < объекта в вершине, идем к левому сыну.

(3) Если элемент > объекта в вершине, идем к правому сыну.

(4) Если элемент = объекту в вершине, то имя найдено; выполняем соответствующие действия и выходим.

(5) Повторяем шаги 2, 3 и 4, пока не достигнем вершины, которая не определена.

(6) Если достигнутая вершина не определена и в дереве нет имени, то выполняем соответствующие действия и выходим.

Пример удаления вершины из дерева.

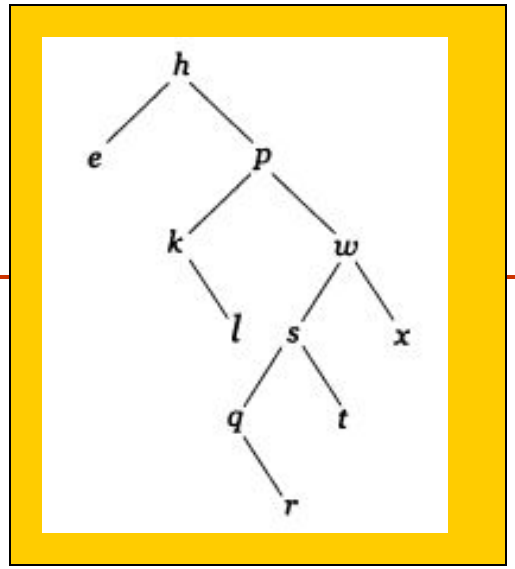
Удаление(элемент)

(1) Если вершина v_0 не имеет сыновей, просто удаляем ее.

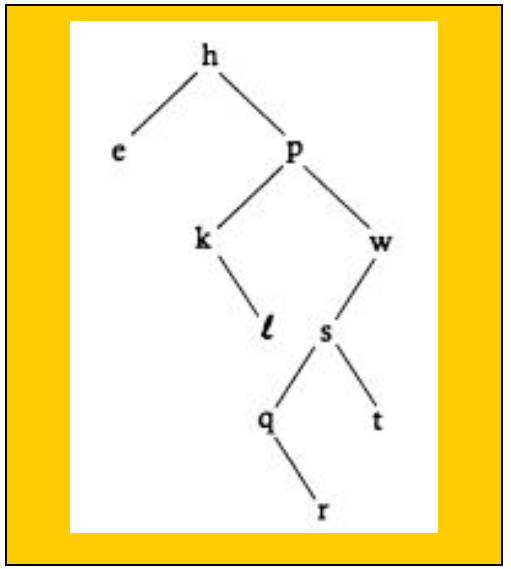
(2) Если вершина v_0 имеет одного сына, удаляем v_0 и заменяем ее сыном.

(3) Если v_0 имеет двух сыновей, находим правого сына v_1 вершины v_0, а затем находим левого сына вершины v_1 (если он существует). Продолжаем выбирать левых сыновей каждой найденной вершины, пока не найдется такая вершина v, у которой не будет левого сына. Заменим v_0 на v и сделаем правого сына вершины v левым сыном родителя вершины v.

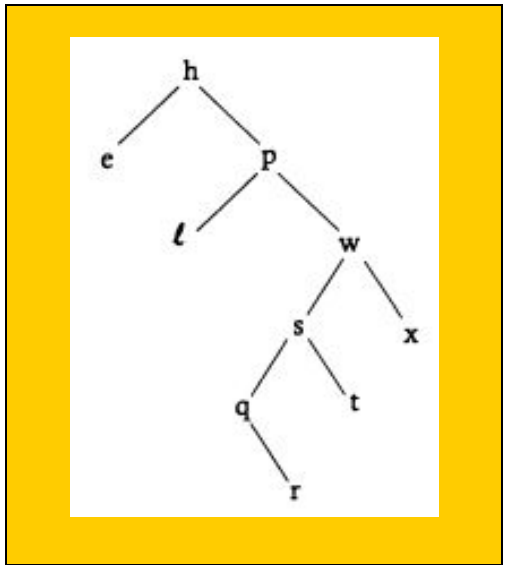
Пример. Дерево:



Если удалить вершину *x*



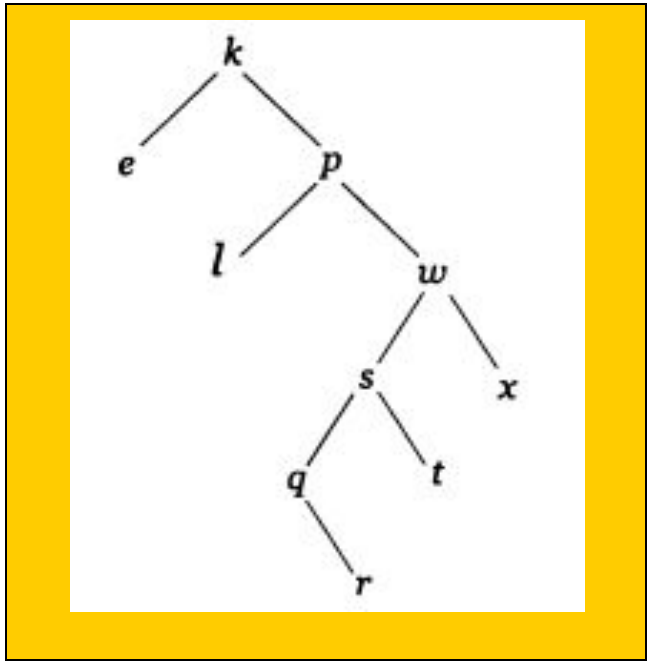
Если удалить вершину *k*



Пример (продолжение).

Если удалить вершину *h,* нужно спуститься к правому сыну *p*, затем к левому сыну *k*.

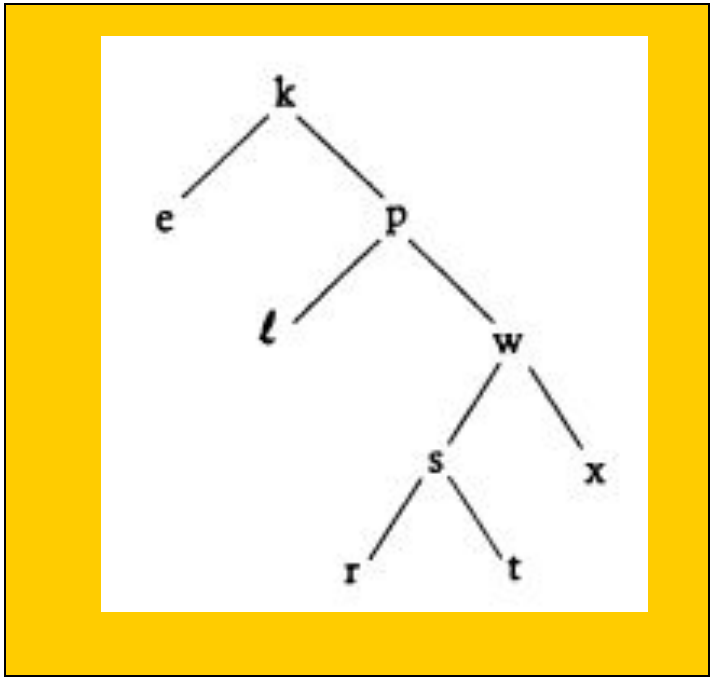
k не имеет левого сына, это искомый элемент для замены. Меняем *h* на *k* и делаем *l* левым сыном *p*.



Пример (продолжение).

Если удалить вершину *p* , следует идти вправо к вершине *w*, затем влево к *s*, затем влево к *q*.

q не имеет левого сына. Меняем p на *q* и делаем *r* левым сыном вершины *s*.



Взвешенные деревья

В компьютере все буквы и другие символы хранятся в виде строк из 1 и 0.

Если данных достаточно много, всегда желательно провести компактификацию.

Проблема: если строки, представляющие разные символы, имеют разную длину, то как узнать, где заканчивается строка одного символа и начинается строка другого.

Определение.

Однозначно декорируемый код для языка как множество, что каждая строка в языке может быть задана однозначно как конкатенация элементов.

В этом случае строки из единиц и нулей, представляющие элементы из *A*, будут кодом.

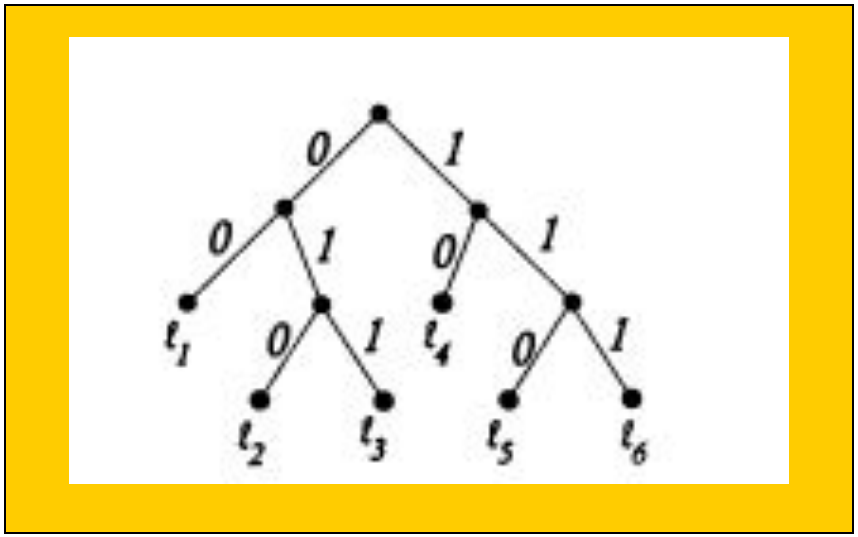
Эти строки образуют однозначно декодируемый код. Разделяя строки на элементы, представляющие *A*, знаем, что представление однозначно. Декодированные слова будут правильные.

Код *C* ***префиксный***, если он обладает свойством, что никакой элемент кода не может быть начальной строкой другого элемента кода.

Конкатена́ция (сцепле́ние) — операция склеивания объектов линейной структуры, обычно строк. Например, конкатенация слов «микро» и «мир» даст слово «микромир».

Пример.

Дерево с листьями



Пути к листам $v1$, $v2$, $v3$, $v4$, $v5$, $v6$ обозначают 00, 010, 011, 10, 110, 111.

Строку, соответствующую данному листу, называют ***путевым кодом.***

Теорема.

В любом бинарном дереве путевые коды для листьев дерева являются префиксным кодом.

Чем меньше вес дерева, тем в большей степени достигнута цель.

Чтобы найти наилучший код для минимизации данных, необходимо найти код с минимальным весом.

Процесс построения такого дерева называется ***алгоритмом Хаффмана***.

Код, приписываемый символам согласно их путевому коду, называется ***кодом Хаффмана***.

Пример.

Имеются буквы и их частоты

символ	частота
A	15
B	25
C	10
D	30
E	15
F	5

Бинарное дерево, что бы наиболее часто используемые элементы были возможно ближе расположены к корню.

Требуется найти такое дерево, что вес дерева

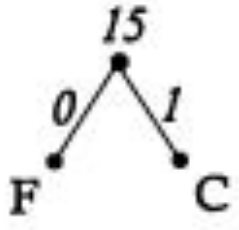
$$w = l_1 f_1 + l_2 f_2 + l_3 f_3 + \ldots + l_n f_n = \sum_{i=1}^{n} l_i f_i$$

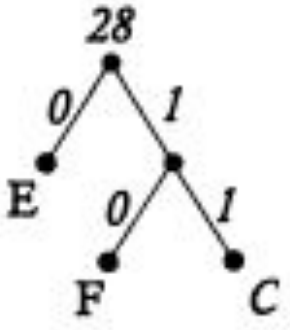
Пример (продолжение).

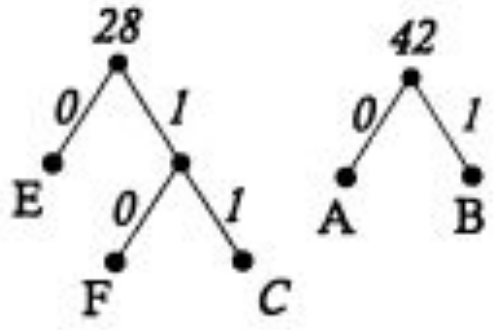
l_i и f_i - уровень и частота заданного элемента.

Упорядочим частоты списке частот (5, 10, 13, 17, 25, 30).

Деревья:







Списки частот: (13, 15, 17, 25, 30)

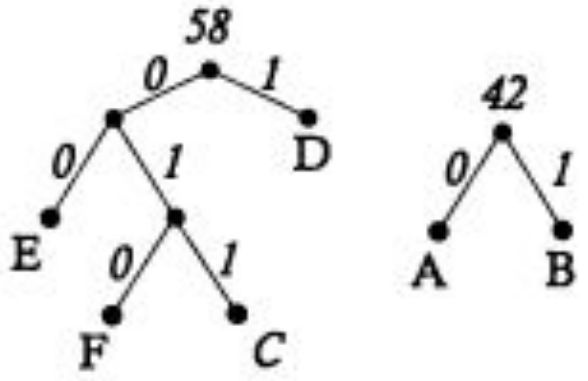
(17, 25, 28, 20)

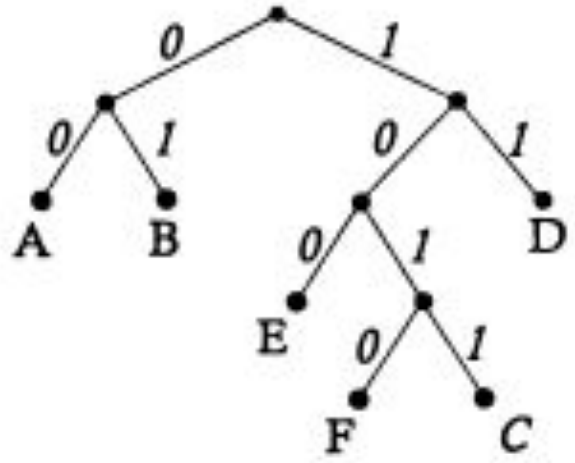
(28, 30, 42)

(42, 58)

Пример (продолжение).

Объединяем два дерева и формируем исходное дерево





Лемма.

Для заданного множества из *n* символов и их частот существует бинарное дерево минимального веса с символами в качестве листьев.

Доказательство:

Существует только четное число бинарных деревьев с *n* листьями. Для каждого

$$w = l_1 f_1 + l_2 f_2 + l_3 f_3 + \ldots + l_n f_n = \sum_{i=1}^{n} l_i f_i$$

и выберем дерево, которое обеспечивает наименьшее значение. Ч.т.д.

Лемма.

В дереве с минимальным весом на максимальном уровне листья присутствуют в парах, т.е. всюду, где есть сын, имеется и правый и левый сын и наоборот.

Лемма.

В дереве с минимальным весом два символа с минимальными частотами расположены на максимальном уровне.

Лемма.

Существует дерево с минимальным весом, в котором два символа с наименьшими частотами имеют одного родителя.

Теорема.

Для заданного множества символов с соответствующими частотами дерево Хаффмана является деревом с минимальным весом.

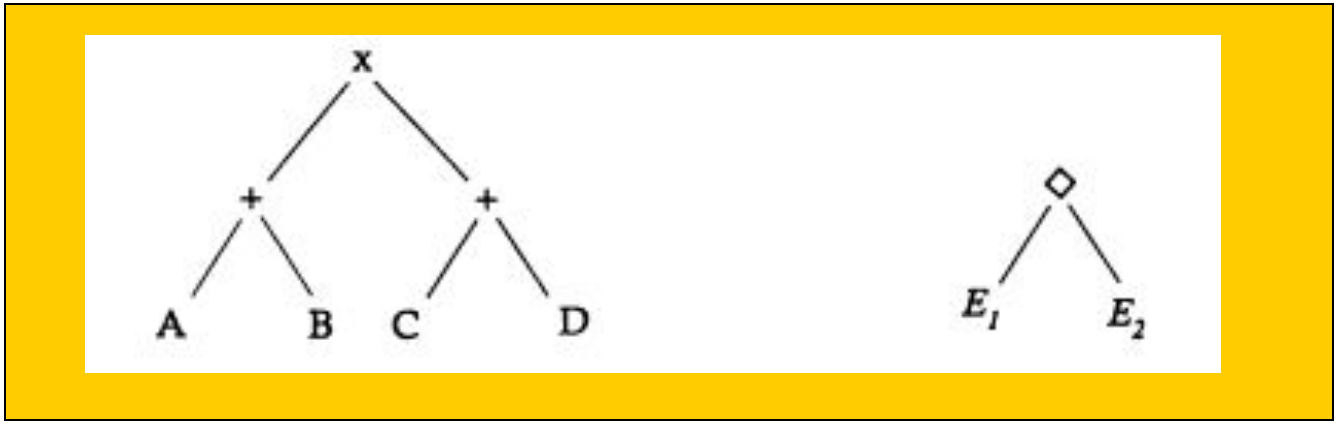
Обход бинарных деревьев.

Рассмотрим способ обхода бинарного дерева

Используем команду ***обработать*** (*n*), где *n* – узел.

Обход дерева в центрированном порядке.

Обращаем процесс, использованный при создании дерева. Если дерево:



- бинарная операция над выражениями *E*1 и *E*2, обрабатываем (печатаем) это как $E_1 \diamond E_2$

Получается выражение в ***инфиксной записи***.

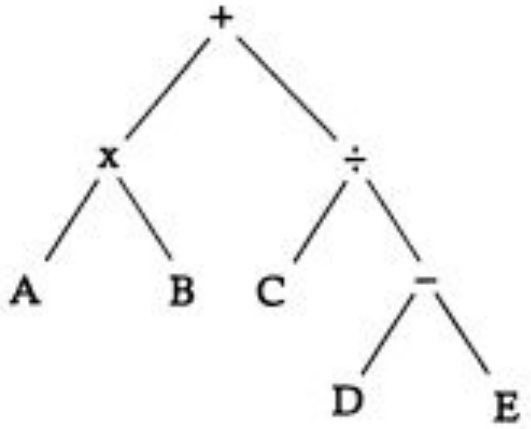
Алгоритм обхода дерева в центрированном порядке (ОПД).

лс (v) – левый сын вершины v .

пс (v) – правый сын вершины v .

Алгоритм обхода дерева в центрированном порядке — ОЦП (*корень*)

(1) Если лс(*корень*) существует, то ОЦП(лс(*корень*)).
(2) Обработать(*корень*).
(3) Если пс(*корень*) существует, то ОЦП(пс(*корень*)).



Последний слайд лекции

!!