Лекция 10

17.04.2018

(деревья)

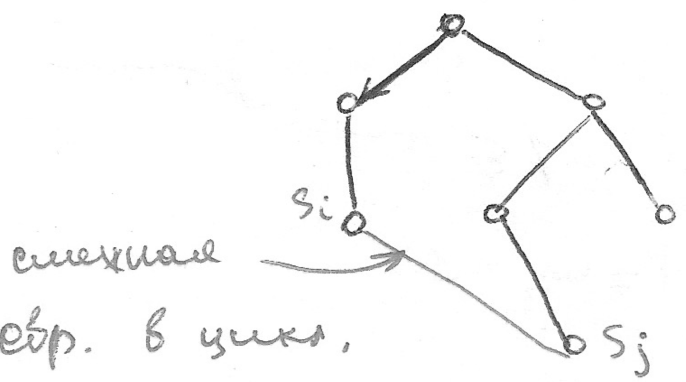
1) $m = n - 1$

~~Если s_i и s_j соед. единств. простая цепь, то~~

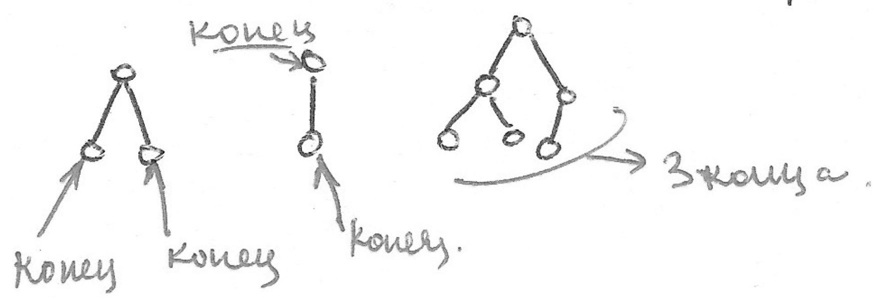
2) Если s_i и s_j – две разные вершины дерева, то их соединяет ! простая цепь.

То, что цепь простая след. из опр. дерева, а то, что цепь ! следует от того, что, если бы цепь была бы не единственной, то это был бы цикл. ($s_i, s_j \in S$, $s_i \neq s_j$)

3) $s_i\ s_j \in S$; s_i, s_j не смежные.



4) Дерево содержит по крайней мере две концевые вершины (вершины [illegible] 1)



5) Теорема (Кейли)

Число размеч. деревьев, кот. можно построить на n вершинах $= n^{n-2}$

(Многие из таких деревьев - изоморфны)

[Кол-во не изоморфных - вопрос нетривиальный (см. Теорема Пойа)]

Ориент. деревья

① Орграф $G(S, V)$ называют ориентир. деревом или ордеревом, если

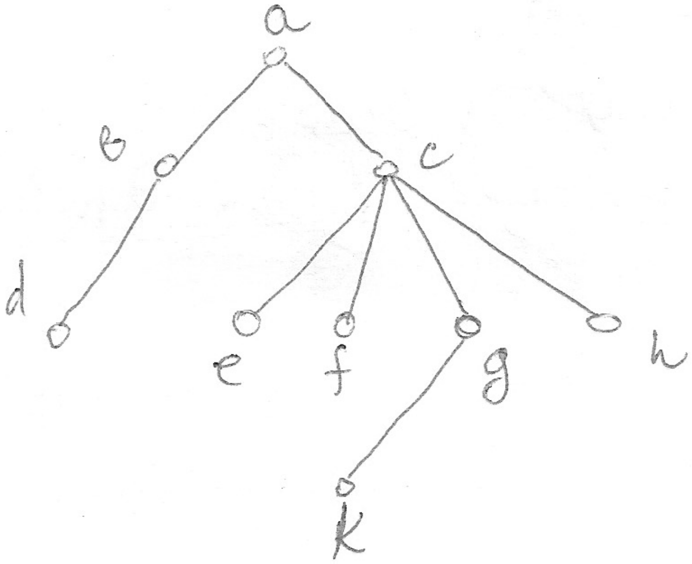
$\exists$ ① ровно одна вершина не имеющая конечных дуг $\Rightarrow p^+(s) = 0$ (это и называют корнем дерева)

② $p^+(s_i) = 1$, $s_i \in S / s$

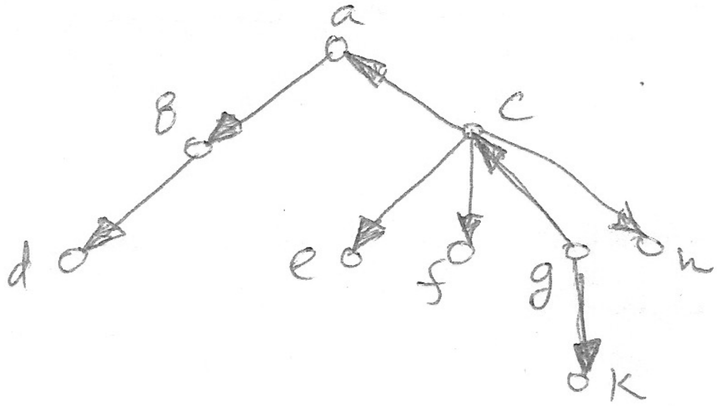
(всех остальных вершин $= 1$)

Очевидно: любое неориентиров. дерево можно превратить в ориентированное, назначив любую вершину корнем.

#

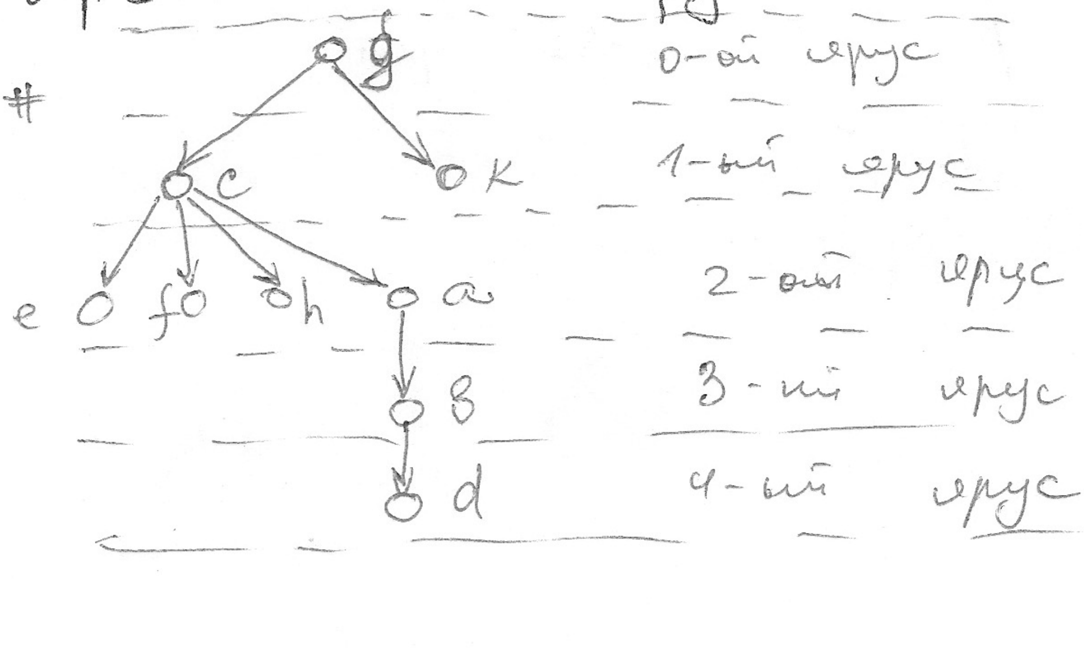


Пусть g - корень



Принято ориентир. деревья обозначать по ярусам так, что корневая вершина - 0-ой ярус и находится вверху.

#



Θ Концевая вершина — **лист**.

Путь из корня в лист — **ветвь**

Длина наибольш. ветви — **высота дерева**

\# ветвь g – c – a – b – d — самая длинная

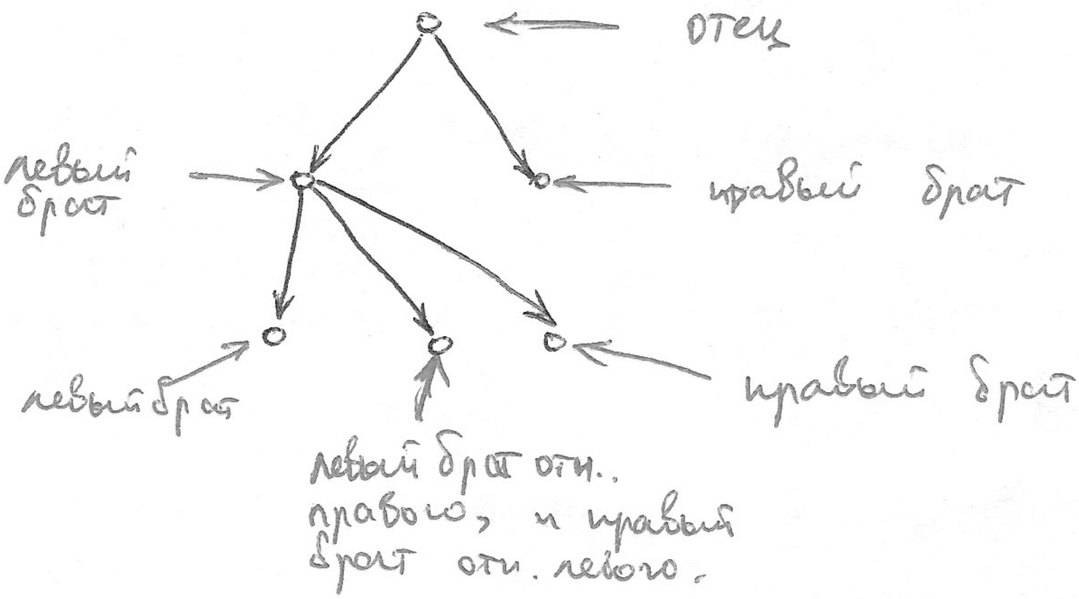
$0 \to 1 \to 2 \to 3 \to 4$

(длина = 4)

Θ Вершины одного уровня образуют **ярус**

расстояние от корня до любой вершины — **уровень вершины**

- Это всё растительная система опр., а есть генеологическая:

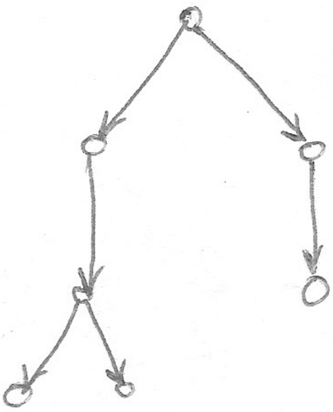


Ориентиров. дерево — бинарное (двоичное), если $p^-(s_i) \le 2$, $\forall s_i \in S$. (иное лишь писать словами)

p^-(лист) $= 0$

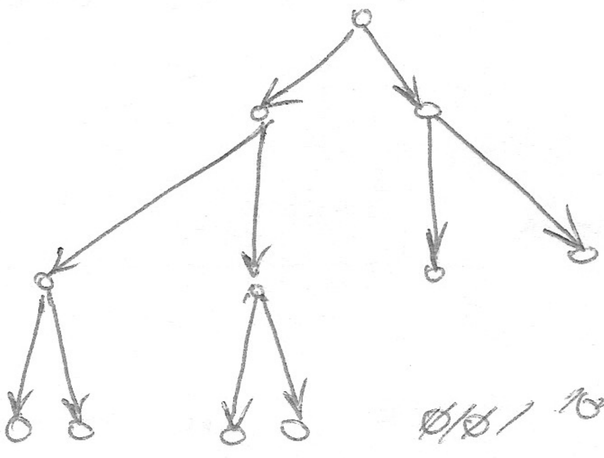
p^-(корня) $= 1$, $= 2$

#



Θ бинарное дерево — полное, если $p^-(s_i) = 2$, $s_i \in S$, кроме листьев

Доделаем дерево из пред. # до полного

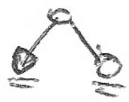


ø/ø/ 100

Θ Сбалансированное = ? (разобраться самому)

По высоте бинарного дерева (используя индукцию), можно показать, что в полном бинарном дереве высоты = h имеется ровно 2^h листьев.

$2^0 = 1$ – только корень = аксиом.

$2^1 = 2$ – 2 листа: 

~~Теорема~~ (просто

Пусть полное бинарное дерево имеет высоту K, тогда число листьев = 2^K и при увеличении высота дерева на 1 (высота дерева = K + 1) → получим число листьев в два раза больше (число листьев = $2^K \cdot 2 = 2^{K+1}$)

Следствие

В произвольном бинарном дереве высоты h маленькое число листьев $\leq 2^h$.

Из этого следует теорема:

Теорема:

Бинарное ориентир. дерево с n листьями имеет высоту h не меньшую чем $\log_2 n$ ($h \geq \log_2 n$)

(Теорема имеет прямое отн. к задаче сортировки произв. структ. данных)

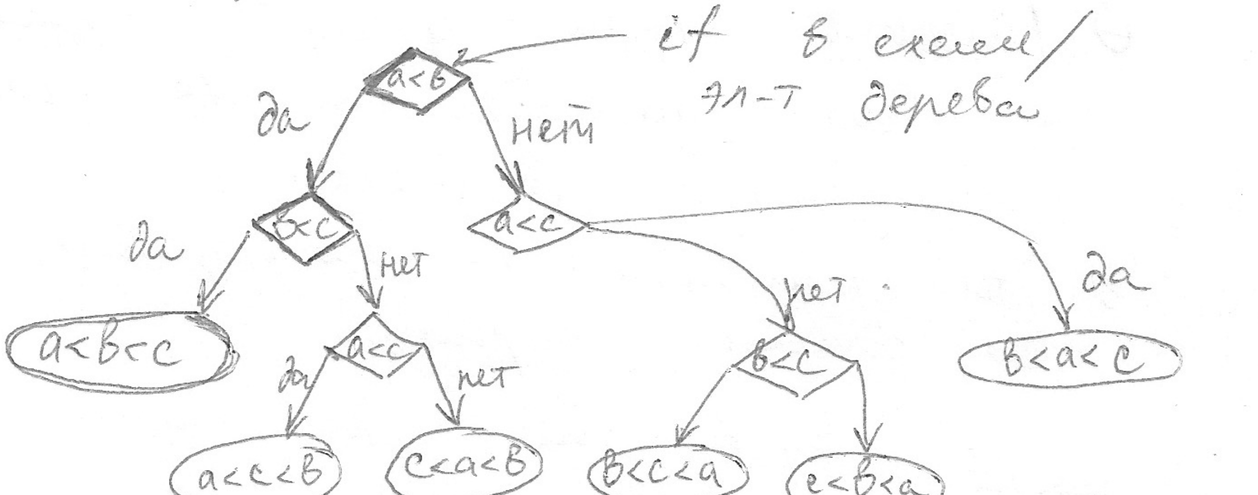
Считаем, что структ. данных сост. эл-ты конечного, линейно упоряд. мн-ва $\{a_1, a_2, a_3, \dots, a_n\}$.

Алг. сортировки должен найти i-ую перестановку эл-тов: $\{a_{i_1}, a_{i_2} .. a_{i_n}\}$, такую, что на эл-тах исходн. мн-ва вводится отношение линейного порядка. $(A, \le)$.

Вычислит. сложность процесса = ?

Подход к оценке сложности алгоритмов сортировки и ряда других алгов принято называть построение дерева решения.

\# $A = \{a, в, c\}$



⇒ Если мн-во $= n$ эл-тов, то сложность равна $n!$ (число листьев при линейной

Сортируя исходное мн-во алгоритм пройдет по какому-то одному пути из корня в лист. $\Rightarrow$ число шагов алгоритма будет пропорционально высоте дерева решений и являться величиной не меньше чем $\log_2(n!)$ (на основании форм. выше теоремы).

Формула Стирлинга для вычисления:

$n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$ $\Rightarrow$ при подстановке получим:

Сложность (высота дерева решений) не может быть ниже чем $n \log_2 n$.

Θ Удаление корня ордерева приводит к его распадению на несколько компонентов связности

Θ Эти компоненты - поддеревья, кажд. со своим корнем.

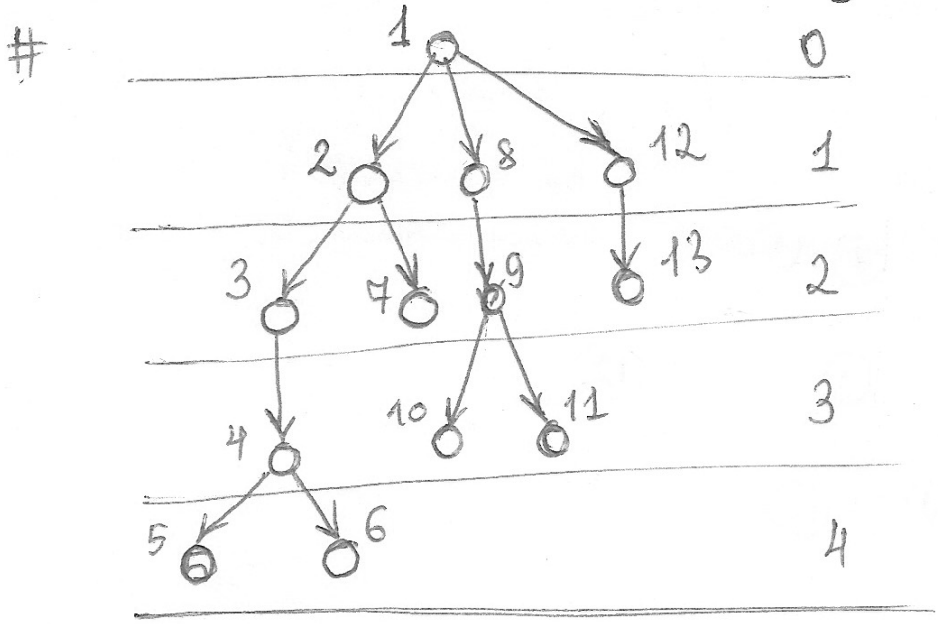
Замечание: Ордерево не явл. сильно связным орграфом, хотя явл. связным.

Задача обхода

Нужно обойти все вершины дерева, вернуться в корень, начать из корня.

Есть два базовых алг. обхода дерева: в глубину / в ширину

Применим к ордереву:



Поиск в глубину:

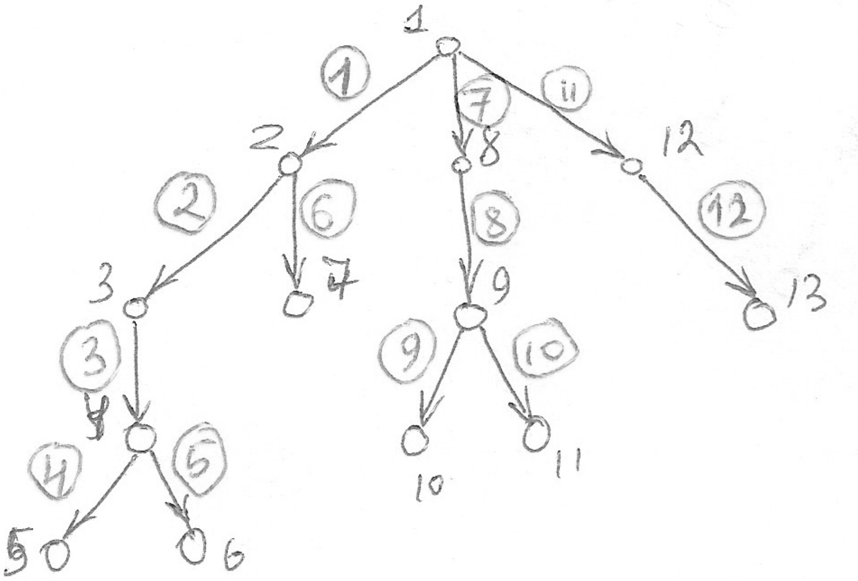
- Текущей вершиной считаем корень
- номер яруса i считаем $= 0$
- номер ребра K считаем $= 0$

1. Выбираем вершину смежную текущей (из яруса $i+1$), объявл. её текущей, полученое ребро помечаем $K = K+1$, соответственно: $i = i+1$.
2. Для текущей вершины выполняем п. 1.
3. Если текущей вершиной стал лист, то возвращ. в смежн. ей вершину ярусом ниже $\Rightarrow$ $i = i-1$ и переходим к п. 1.

4. Е. для тек. вершины нет ни одного инцидентного ~~смежного~~ ей непомеченного ребра, возвращаемся в смежную ей вершину ярусом выше ($i = i+1$)

5. Алгоритм заканчивается, когда помечены все ребра, и $i = 0$.

Квинтэссенция: номера ребер покажут рез-ты обхода дерева в глубину.



Алгоритм в ширину аналогичен, только сначала перебираем все ~~ребра~~ вершины одного яруса, и только тогда переходим на ярус ниже.

Вычислит. сложность обоих одинаково.

- Если дерево достаточно широкое, а листья расположены на достаточно близких ярусах, то надо исп. алгоритм в глубину
- Если наоборот узкое, высокое, листья отстают далеко, то в ширину
- Не всегда требуется обход полный (все вершины), а только поиск опр. вершины, то тоже опираемся на топологию, аналогично рассм. ранее.

/* дискретная математика для не математиков. - Интересная книжка! (Упорядочить кухонную утварь по принципу бинарного дерева */

Остовы

Θ Есть $G(S, U)$, можно рассмотреть подграф $G'(S', U')$, $G' \subset G$.
G' - остовный подграф, если $S' = S$, а $U' \subset U$

Θ Остовный подграф, кот. явл. деревом называют остовным деревом, или просто остовом, проще говоря остов графа G - дерево построенное на мн-ве его вершин, ребра дерева явл. подмн-вом мн-ва графа G.

Изначально поиск ~~граф~~ остова применяется к неориентированным графам.

\# Пример на след. лекции.