1. *Основные понятия теории графов: неориентированные и ориентированные графы(276), цепи, пути, циклы, контуры, маршруты. Подграфы. Компоненты и бикомпоненты(286).*

~~~ ~~~

• - Неориентированный граф G задается двумя множествами $G = (V, E)$, где V – конечное множество вершин (узлов), E – множество *неупорядоченных* пар на V, элементы которого называют ребрами.

- Ориентированный граф G задается двмя множествами $G = (V, E)$, где V – конечное множество вершин (узлов), E – множество *упорядоченных* пар на V, элементами которого называют дугами.

- Цепь в *неориентированном* графе – последовательность вершин $v_0, v_1, \dots, v_n, \dots$ такая, что $v_i - v_{i+1}$ (вершины соединены ребром) для любого i, если v_{i+1} существует.

- Путь в *ориентированном* графе – последовательность вершин $v_0, v_1, \dots, v_n, \dots$ такая, что $v_i \to v_{i+1}$ (из вершины в вершину ведет дуга) для любого i, если v_{i+1} существует.

Простая – цепь, все вершины которой, кроме быть может первой и последней, попарно различны и все ребра попарно различны. Простой – путь, все вершины которого, кроме быть может первой и последней, попарно различны.

- Цикл (контур) – простая цепь (путь) ненулевой длинының совпадающими концами (началом и концом).

- Маршрутом в графе (ориентированном) $G = (V, E)$ называется последовательность вершин и рёбер (дуг) вида $v_0, e_1, v_1, e_2, \dots, v_{n-1}, e_n, v_n$, где $v_i \in V, i \in [0, n]$, $e_i \in E$, где ребра (дуги) e_i связывают вершины v_{i-1} и v_i. *//определение взято наполовину из вики, наполовину из здравого смысла, я вообще не уверен что Белоусов это давал…*

• Граф $G_1 = (V_1, E_1)$ называют подграфом графа $G = (V, E)$, если $V_1 \subseteq V,\ E_1 \subseteq E$.

• Неориентированный (ориентированный) граф называют связным, если любые две его вершины соединены цепью (для любых двух его вершин u,v вершина v достижима из u ИЛИ u достижима из v). Ориентированный граф называют сильно связным, если для любых двух его вершин u и v вершина v остижима из u И u достижима из v.

Компонента связности графа (любого) – его максимальный связный подграф. Бикомпонента ориентированного графа – его максимальный сильно связный подграф.

2. *Деревья и их классификация. Теорема о числе листьев в полном p-дереве.*

~~~ ~~~

• *Ориентированное* дерево – бесконтурный ориентированный граф с *одним входом* (вершиной с нулевой полустепенью захода), в котором для любой вершины, кроме входа, полустепень захода равна 1; вершины с полустепенью исхода равной 0 называются листьями дерева.

Неориентированное дерево – любой связный неориентированный граф, в котором нет циклов.

• Ориентированное дерево называют p-деревом, если полустепень исхода для каждой его вершины *не больше* p. Высота вершины h(v) – максимальная длина пути из этой вершины в лист; высота дерева – высота его корня; глубина вершины d(v) – длина пути из корня дерева в эту вершину. P-дерево называют полным, если полустепень исхода всех его вершин кроме листьев *равна* p, и уровни его листьев $\big(h(v) - d(v)\big)$одинаковы.

Теорема: В полном p-дереве высоты H число листьев равно p^H.

3. *Методы систематического обхода вершин графа: поиск в глубину. (319)*

~~~ ~~~

• Поиск в глубину. На вход подается граф $G = (V, E)$, заданный списками смежности, и начальная вершина v0. На выходе имеем множества древесных и обратных ребер (T и B), множество F_c фундаментальных циклов, массив D, содержащий номера вершин.

Вначале все вершины графа помечаются как «новые».
При достижении некоторой вершины v (при запуске алгоритма v=v0), с неё снимается метка «новая», ей присваивается D-номер, вершина v заносится в стек и просматриваются вершины из её списка смежности $L[v]$.
Если вершина w из этого списка «новая», то ребро $\{v, w\}$ помечается как древесное, после чего переходим к вершине w. Далее процесс повторяется – просматривается список смежности, выбирается первая «новая» вершина, анализ же остальных вершин из списка «откладывается на потом».
Если же вершина w пройденная, и ребро $\{v, w\}$ не является древесным, то $\{v, w\}$ помечается как обратное, а из вершин, находящихся в стеке (с верхшины стека до w) формируется фундаментальный цикл. После анализа всех вершин из списка смежности L[w], возвращаемся в вершину v и продолжаем анализировать список смежности L[v].
«Путешествие» прекратится, когда мы вернемся в исходную вершину v0 и окажется что либо все вершины перестали быть «новыми», либо остались «новые» вершины, но из v0 больше никуда перейти нельзя.

В случае ориентированного графа, в результате поиска в глубину получают также множество прямых дуг $F\ \big((u, v) \in F \text{ если } D[u] < D[v],\ v \notin Stack\big)$ и поперечных дуг $C\ \big((u, v) \in C \text{ если } D[u] > D[v],\ v \notin Stack\big)$.
~~~

```
begin                                    //вглубь
T,B,FC,Stack:=0; Count := 1
For all v ∈ V do New[v] := 1 end
For all v ∈ V while (∃v)(New[v] = 1) do
   Search_D(v) end
End

Proc Search_D(v)
New[v] := 0
D[v] := Count;  Count := Count + 1
v → Stack
For all (w ∈ L[v]) do
   If New[w] then begin
      {v,w} → T
      Search_D(w)
   End
   Else
      if {v,w} ∉ T then begin
         {v,w} → B
         Read(v..w) → F_c
      end
End for
Stack → v
End Search_D
```

4. *Поиск кратчайших расстояний от фиксированной вершины: алгоритм волнового фронта и поиск в ширину в орграфе с числовыми метками дуг. (324)*

~~~ ~~~

• Алгоритм волнового фронта. На вход подается граф $G = (V, E)$, заданный списками смежности; v0 – начальная вершина. На выходе имеем массив M меток вершин.

Вначале всем вершинам графа присваиваются метки бесконечности, начальной вершине v0 M[v0] = 0. Достигнув некоторой вершины v (при запуске поиска, v=v0, и v0 сразу же заносится в очередь), просматриваем все вершины w из её списка смежности L[v]. Если метка M[w] равна бесконечности, то присваиваем вершине w метку M[w] := M[v]+1, и заносим вершину w в очередь. Просмотрев весь список L[v], выгружаем вершину v из очереди, и повторяем процес для следующей вершины в очереди. Алгоритм завершится, когда очередь опустеет.

В случае ориентированного графа $M[w] := M[v] + \phi(v, w)$, если $M[w] > M[v] + \phi(v, w)$; на выходе получают длины кратчайших путей из v0 в остальные вершины (+∞, если пути не существует).

```
For all v ∈ V do M[v] := +∞ end
Q := 0
v₀ → Q;  M[v₀] = 0
For all v ∈ V while Q ≠ 0 do
   For all w ∈ L[v] do
      If M[w] = +∞ then begin
         M[w] := M[v]+1
         w → Q
      End
    End
   Q → v
  End
```

5. *Алгоритм Дейкстры.*

~~~ ~~~

• На вход подается орграф $G = (V, E)$, функция разметки $\phi: E \to R_0^+$ и начальная вершина $v_0 \in V$. На выходе имеем длины кратчайших путей из v0 в остальные вершины.

При запуске алгоритма начальной вершине присваивается метка 0, остальным вершинам – метки бесконечности. Метки делятся на постоянные и временные.

На очередном шаге из всех вершин с не-бесконечными метками выбирают вершину v, которая имеет наименьшую временную метку среди вершин с временными метками, и меняют её метку с временной на постоянную. Далее для всех вершин $w \in L[v]$, если $M[w] > M[v] + \phi(v, w)$, то $M[w] \coloneqq M[v] + \phi(v, w)$. Алгоритм завершается, когда не остается временных меток, не равных бесконечности.

```
Begin
M[v0] := 0
For all (v ∈ V/{v₀}) do M[v] := +∞ end
S := ∅
While (v ≠ S) & (∃v ∈ V\S)(M[v] ≠ +∞) do
   w|min M[w] → S
   For all v ∈ L[w] ∩ (V\S) do
      M[v] := min(M[v], M[w]+ϕ(v,w))
End
```

6. *Поиск фундаментальных циклов в неориентированном графе. (319)*

~~~ ~~~

• В неориентированном графе фундаментальным циклом называется его подграф, являющийся циклом и содержащий только одно обратное ребро.

Фундаментальные циклы можно найти с использованием поиска в глубину. При последовательном обходе вершин, рассматриваем вершину $w \in L[v]$. Если w не новая и ребра (v,w) нет среди древесных, то из вершин, находящихся в стеке от вершины стека до вершины w, формируется список Cycle. После этого сформированный фундаментальный цикл добавляется к множеству фундаментальных циклов Fc.

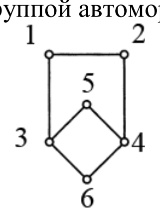
//суть там следующая - предположим у нас есть в стеке цепь 1 - 2 - 3 - 5 - 7; 1 мы считали вначале, 7 в конце, поэтому 7 будет верхушкой стека; и тут мы БАЦ, переходим из 7 кудато дальше и вновь натыкаемся на 2, а она у нас уже была, поэтому мы берем все что находится в стеке между вершиной стека (7) и найденым повторением (2), и заносим в цикл (2 - 3 - 5 - 7 - 2). Если сможете описать это в общем виде лучше чем я - дерзайте

7. *Изоморфизм графов. (342) Группа автоморфизмов графа и ее вычисление.*

~~~ ~~~

• Пусть даны графы $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$. Отображение $h: V_1 \to V_2$ называют изоморфизмом, если оно биективно и сохраняет отношение смежности, $(\forall\, u, v \in V)\big(u\, \rho_1\, v \iff h(u)\, \rho_2\, h(v)\big)$. Изоморфизм графа на себя называют автоморфизмом.

• Композиция любых двух автоморфизмов графа есть автоморфизм; подстановка, обратная к автоморфизму также является автоморфизмом. Таким образом, множество всех автоморфизмов графа образует группу по операции композиции, называемую группой автоморфизмов графа.



Для графа, изображенного на рисунке, группу автоморфизмов образуют подстановки e, (3 4)(1 2), (5 6) и их композиция (3 4)(1 2)(5 6).

8. *Задача о путях в ориентированном графе, размеченном над полукольцом (326)и ее решение с помощью алгоритма Флойда — Уоршелла — Клини(339). Задача о достижимости и поиске кратчайших расстояний между двумя узлами графа. (327)*

~~~ ~~~

• Размеченным ориентированным графом называют пару $W = (G, \phi)$, где $G = (V, E)$ – обычный ориентированный граф, $\phi: E \to R$ – функция разметки со значениями в некотором идемпотентном полукольце $\mathcal{S} = (S, +, *, \mathbb{O}, \mathbf{1})$, $(\forall e \in E)(\phi(e) \neq 1)$.

Если задать орграф с помощью матрицы смежности $A = (a_{ij})$, $a_{ij} = \begin{cases} 1, & (v_i, v_j) \in E \\ 0, \text{иначе} & \end{cases}$, то задача о достижимости сводится к вычислению матрицы достижимости графа $C = (c_{ij})$, $c_{ij} = \begin{cases} 1, & v_i \to^* v_j \\ 0, \text{иначе} & \end{cases}$.

Если задать орграф с помощью матрицы меток дуг $A = (a_{ij})$, $a_{ij} = \begin{cases} \phi(v_i, v_j), \text{если } (v_i, v_j) \in E \\ 0, \text{иначе} \end{cases}$, то задача о поиске кратчайших расстояний между двумя узлами графа сводится к вычислению матрицы кратчайших расстояний $C = (c_{ij})$, $c_{ij} = \begin{cases} \text{длине кратчайшего пути из } v_i \text{в } v_j, \text{если } v_i \Rightarrow^* v_j \\ +\infty, \text{иначе} \end{cases}$.

• Обе задачи можно решить с помощью алгоритма Флойда – Уоршелла – Клини. В случае задачи о достижимости, в качестве полукольца $\mathcal{S}$ выбирают полукольцо $\mathbb{B} = (\{0,1\}, max, min, 0, 1)$, а в случае задачи о поиске кратчайших расстояний: $\mathcal{R}^+ = ([0, +\infty], min, +, +\infty, 0)$. После этого матрица C путем решения системы уравнений $c_{ij}^{(k)} = c_{ij}^{(k-1)} + c_{ik}^{(k-1)} c_{kj}^{(k-1)}$, где $c_{ij}^{(0)} = \begin{cases} a_{ij}, & i \neq j \\ 1 + a_{ij}, & i = j \end{cases}$, a k – максимальный номер вершины, в которую разрешено заходить по пути из vi в vj.

\- - - - - - - - - - - - - -
~~~

1. *Алфавит, слово, язык. (462) Операции над языками(469), полукольцо всех языков в заданном алфавите и его замкнутость.*

~~~ ~~~

• - Алфавит – произвольное непустое конечное множество $V = \{a_1, \dots, a_n\}$, элементы которого называют буквами или символами.

- Словом или цепочкой в алфавите V называют произвольный кортеж из множества V^k для различных k; при k=0 получаем пустой кортеж, называемый пустым словом λ.

- Языком в алфавите V называется произвольное подмножества множества всех слов языка V^*.

• - Над языками допустимы все теоретико-множественные операции: объединение, пересечение, разность, симметрическая разность, дополнение ($\bar{L} = V^* \backslash L$).

- Соединением языков L1, L2 называют язык L1L2, состоящий из всех возможных соединений слов xy, в которых слово x принадлежит первому, а слово y = второму языку; $L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$

- Итерацией языка L называют объединение всех его степеней, $L^* = \{\lambda\} + L + L^2 + L^3 + \cdots$.

• Алгебра $\mathbb{L}(V) = \left(2^{V^*}, \cup, *, \emptyset, \{\lambda\}\right)$ является замкнутым полукольцом. Аксиомы полукольца проверяются непосредственно; замкнутость полукольца следует из существования объединения любого семейства множеств, служащего точной верхней гранью этого семейства (относительно операции включения), а также из тождеств $L(\bigcup_{i \in I} P_i) = (\bigcup_{i \in I} LP_i)$, $(\bigcup_{i \in I} P_i)L = \bigcup_{i \in I} P_i L$.

2. *Регулярные языки и регулярные выражения. (490)*

~~~ ~~~

• Среди всех языков в алфавите V выделяют множество регулярных языков. База: языки $\emptyset, \{\lambda\}, \{a\}$ – регулярные. Если L_1, L_2 – регулярные языки в алфавите V, то регулярны также будут их объединение, соединение и итерация. Других регулярных языков в алфавите V нет.

Регулярное выражение – слово, обозначающее регулярный язык. Регулярные выражения $\emptyset, \lambda, a$ обозначают соответственно языки $\emptyset, \{\lambda\}, \{a\}$. Если $\alpha \mapsto K, \beta \mapsto K$, то $(\alpha \mapsto \beta) \mapsto K \cup L$; $(\alpha\beta) \mapsto K * L$; $\alpha^* \mapsto K^*$, $\alpha^+ \mapsto K^+$.

3. *Понятие конечного автомата (КА) и языка, допускаемого КА(502,8). Анализ и синтез КА. (518)*

~~~ ~~~

• Пусть V – некоторый алфавит. Конечным автоматом называется орграф (Q,E), размеченный над полукольцом R(V), при этом:

1) на функцию разметки ϕ: $E \to R(V)$ наложены ограничения: $\phi(e) \neq \emptyset$; $\phi(e) = \{\lambda\}$ или $\phi(e) \subseteq V$

2) задана вершина $q_0 \in Q$, называемая входной

3) задано множество вершин F, называемых заключительными

КА допускает цепочку x, если она читается на некотором пути, ведущем из начальной вершины в одну из заключительных. Язык, допускаемый КА – множество всех допустимых им цепочек.

• Для КА существует две задачи – анализа (для данного КА найти допускаемый им язык) и синтеза (для данного регулярного языка построить допускающий его КА).

4. *Теорема Клини о совпадении класса языков, допускаемых КА и класса регулярных языков. (514)*

~~~ ~~~

• **Теорема:** Пусть $V = \{a_1, \dots, a_n\}$ – произвольный алфавит. Язык $L \subseteq V^*$ является элементом полукольца R(V) (то есть является регулярным языком) тогда и только тогда, когда он допускается некоторым конечным автоматом.

5. *Детерминизация и минимизация КА(521,31). Регулярность дополнения регулярного языка и пересечения двух регулярных языков(526). Проблемы пустоты и эквивалентности. (531)*

~~~ ~~~

• КА называют детерминированным, если: в нем нет -переходов, и из каждого состояния по любому символу возможен переход только в одно состояние. КА M1 и M2 называют эквиввалентными, если они допускают один и тот же язык.

Для любого конечного автомата может быть построен эквивалентный ему детерминизированный КА. Детерминизация происходит в два этапа – на первом происходит удаление из автомата лямбда-переходов, на втором непосредственно детерминизация (например с помощью метода вытягивания).

В свою очередь, для любого детерминизированного КА может быть построен ему детерминизированный КА с минимальным числом состояний. Минимизация происходит за счет «группировки» состояний в классы эквивалентности.

• Пусть даны два регулярных языка L1 и L2. Тогда регулярными языками также будут их дополнения ($\bar{L} = V^*/L$), пересечение $\left(L1 \cap L2 = \overline{\overline{L1} \cup \overline{L2}}\right)$, разность $(L1/L2 = L1 \cap \overline{L2})$ и симметрическая разность $\left(L1 \Delta L2 = (L1 \backslash L2) \cup (L2 \backslash L1)\right)$

• Проблема пустоты: дан КА, необходимо выяснить, не является ли пустым язык, который он допускает. Проблема эквивалентности: для двух заданных КА M1 и M2 необходимо проверить, являются ли они эквивалентными, т.е. совпадают дли допускемые ими языки.

6. *Лемма о разрастании для регулярных языков. (538)*

~~~ ~~~

• **Теорема:** Если L – регулярный язык, то существует натуральная константа K_L, зависящая от L, такая, что для любой цепочки $x \in L$, длина которой не меньше K_L, x допускает представление в виде $x = uvw$, где $v \neq \lambda$ и $|v| \leq K_L$, причем для любого $n \geq 0$ цепочка $x_n = uv^n w \in L$.