

Гл. 4: Теория графов

Лит-та: Харари Н. А. "ТГ" ДМ для программистов.
Емелин В. А. и др. Лекции по ТГ.
Иванов Б. Н. ДМ. Алгоритмы и пр-мы.

Часть 1. Характеристика графов

§ 1. Основное определение

Опр. Пусть V - мн-во, $V \neq \emptyset$

E - мн-во пар элементов из V , без повторений

Пара (V, E) наз-ся неориентированным графом (графом) $G(V, E)$.

Элементы мн-ва V - вершины

Элементы мн-ва E - ребра

(V) - мн-во вершин

(E) - мн-во ребер

$|V| = n$ - число вершин - порядок графа

$|E| = m$ - число ребер

Опр.: 1) Вершины u и v смежны, если пара $(u, v) \in E$ (uv - ребро).

2) Ребра e_1 и e_2 смежны, если у них одна вершина общая.

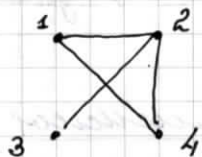
Опр.: Вершина u и ребро e инцидентны, если ребро e проходит $x-y$ верш. u .

Опр.: Множество смежности вершины v — это множество вершин, смежных с v .

$$S(v) = \{u \in V \mid (u, v) \in E\}$$

Опр.: Диаграмма графа — рисунок, на котором вершинам соотв. точки, ребрам — линии, соединяющие эти точки.

Пр.: $V = \{1, 2, 3, 4\}$
 $E = \{(1, 2), (2, 3), (2, 4), (1, 4)\}$

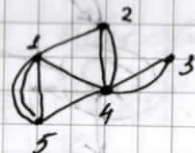


§2 Обобщение определения

Опр.: Если E является не множеством, а совокупностью, т.е.

может содержать несколько одинаковых элементов, то такие элементы называются кратными ребрами, а граф называется мультиграфом.

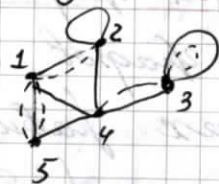
Пр.:



$$E = \{(1, 5), (1, 5), (1, 5), (1, 4), (1, 2) \text{ и т.д.}\}$$

Опр.: Если элементом множества совокупности (мультимножества) E может быть пара, в которой первой и второй элемент совпадают ($\exists v \in V$ т.е. $(v, v) \in E$), то такой элемент $u \in E$ называется петлей, а граф наз-ся псевдографом.

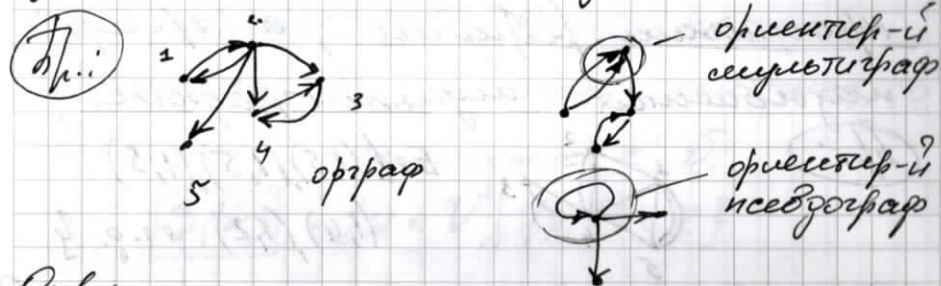
Пр.:



Опр.: Если элементами мн-ва E являются упорядоченные пары $(E \subseteq V^2)$, то граф наз-ся

ориентированным (орграфом).

В этом случае V - мн-во узлов; E - мн-во дуг.



Опр: Граф (или ориграф) наз-ся простым, если в нем нет кратных ребер (дуг) и петель.

Далее, если не оговорено иное, будем ∇ простые графы

§3 Изоморфизм графов

Теорема Число графов на n ^{различных} вершинах равно $2^{\frac{n(n-1)}{2}}$

Определим чему равно макс-мально возможное число ребер в графе с n вершинами $f(n)$

$f(n)$ равно числу способов выбрать две вершины из n . (Без учета порядка их след-я).

Первую вершину м-о выбрать n способами. Для выбора второй остается $(n-1)$ способов. Всего $n \cdot (n-1)$ вариантов, но здесь учит-ся порядок вершин. Без учета порядка число ребер р.б.

в 2 раза меньше \Rightarrow

$$f(n) = \frac{n \cdot (n-1)}{2} = C_n^2 = \frac{n!}{(n-2)! \cdot 2!}$$

Для графов на n вершинах отличаются друг от друга ребрами \Rightarrow каждой из этих графов однозначно опр-ся вектор из 0 и 1 длины $f(n)$:
 $\begin{cases} 1 - \text{соотв. ребро включено в граф} \\ 0 - \text{нет} \end{cases}$

Число таких векторов равно $2^{(n)} \Rightarrow$ и число графов на n вершинах равно $2^{f(n)} = 2^{\frac{n(n-1)}{2}}$.

Однако, ^{нашего} существенно бывает лишь то, что есть объекты (вершины графа) и связи м-у объектами (ребра графа). С этих позиций графы, которые получаются один из другого изменением наименования вершин, разумею не различать.

Опр.: Два графа $G_1(V_1, E_1)$ и $G_2(V_2, E_2)$

изоморфны $G_1 \approx G_2$, если

\exists биекция $\varphi: V_1 \rightarrow V_2$,

сохраняющая смежность:

$$(u, v) = e_1 \in E_1 \Rightarrow (\varphi(u), \varphi(v)) = e_2 \in E_2 \text{ и}$$

$$(u, v) = e_2 \in E_2 \Rightarrow (\varphi^{-1}(u), \varphi^{-1}(v)) = e_1 \in E_1$$

Само отображение φ наз-ся

изоморфизмом графов

Упр. 3. Изоморфизм графов — это отношение экв-тис.

1) рефл-ть: $G \approx G$; изоморфизм — тождеств. ф-я

2) сим-ть: $G_1 \approx G_2$ с уом. $\varphi \Rightarrow G_2 \approx G_1$ с уом. φ^{-1}

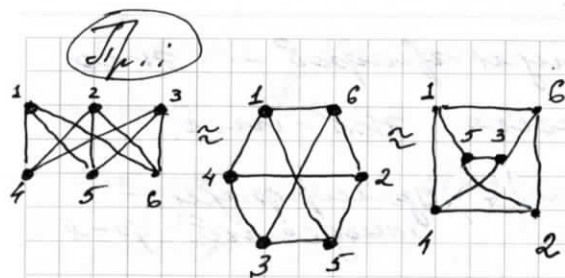
3) тр-ть: $G_1 \approx G_2$ с уом. φ_1 ; $G_2 \approx G_3$ с уом. $\varphi_2 \Rightarrow G_1 \approx G_3$ с уом. $\varphi_2 \circ \varphi_1$.

Из упр. следует, что мн-во всех графов разбивается на классы экв-ти так, что графы из одного класса попарно изоморфны.

Изоморфные графы будем считать совпадающими.

? Вопрос о том, являются ли два графа изоморфными в общ. случае является сложным.

пересор!



Опр.: Инвариант графа — числовая характеристика, одинаковая для всех изоморфных графов.

Примеры инвариантов:

- 1) мн-во вершин $n(G)$
- 2) мн-во ребер $m(G)$
- 3) и др.

Проблема: не известно ^{вычислимых до полиномиал. врем.} никакого набора инвариантов, определяющих граф с точностью до изоморфизма. (т.е. в известного набора инвариантов найдется хотя бы два неизоморфных графа, инварианты которых совпадают.)

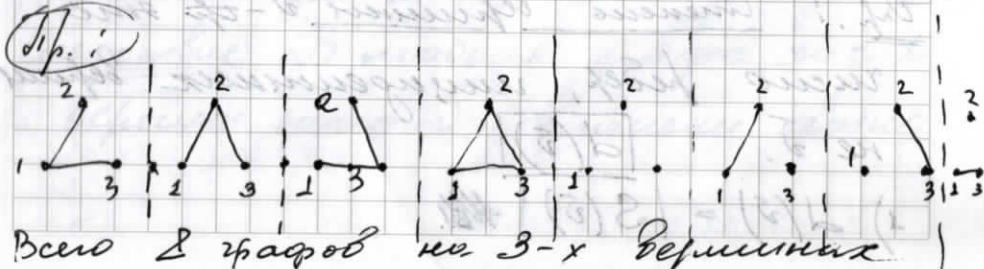
Опр. Если вершинам графа сопоставить метки (натуральные числа), то граф назовем помеченным.

Пр.:

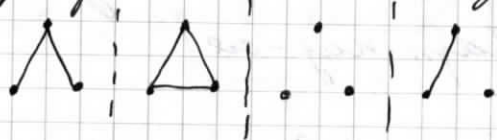
Как помеченные эти графы различны (это 3 изоморфных м-у собой графа) Как непомеченные они равны (изоморфны).

Формула Поля: Число всех непомеченных графов на n вершинах равно $\approx 2^{\frac{n(n-1)}{2}} / n!$

Т.е. число помеченных графов на n вершинах примерно в $n!$ раз больше числа непомеченных.



Если не рассматр. немощ.
графы, то получаем 4 графа



§4. Подграфы

Опр.: Граф $G_1(V_1, E_1)$ называется подграфом графа $G(V, E)$, если $V_1 \subseteq V$; $E_1 \subseteq E$.

$$G_1 \subseteq G$$

Опр. Остаточный подграф: подграф, т.е. $V_1 = V$.

Опр. Правильный подграф (или подграф, порожденный мн-м V_1): подграф, т.е.

$\forall u, v \in V_1$: если $(u, v) \in E \Rightarrow (u, v) \in E_1$.

§5. Степени вершин

Опр.: Степень вершины v -ой это число ребер, инцидентных вершине v . $d(v)$

$$1) d(v) = |S(v)|$$

$$2) \forall v \in V \quad 0 \leq d(v) \leq n-1 \quad (n=|V|)$$

-Если степени всех вершин графа равны, то граф называется регулярным.

-Если $d(v)=0$, то v - вислая вершина.
 $d(v)=0$, то v - узел-верш.

Для ориграфов степень узла складывается из двух величин:

1) $d^+(v)$ - полустепень захода - число входящих дуг

2) $d^-(v)$ - полустепень исхода - число исходящих дуг.

Теорема Эйлера или лемма о рукопожатиях Сумма степеней вершин графа четна и равна удвоенному числу ребер.

$\sum_{v \in V} d(v) = 2|E|$, т.к. каждое ребро в сумме учит-ся дважды.

Следствие В любом графе число вершин нечетной степени четно.

§6 Маршруты

Опр. Маршрут в графе - это чередующаяся пос-ть вершин и ребер, в которой любые два соседних элемента инцидентны.

$$v_0, v_1, v_2, v_3 \dots v_k, v_k \quad (e_i = (v_{i-1}, v_i))$$

Для простого неориентир-го графа достаточно указать только пос-ть смежных вершин или смежных ребер.

v_0, v_1, \dots, v_k - это (v_0, v_k) -маршрут.

Опр. Если $v_0 = v_k$, то маршрут замкнутой.

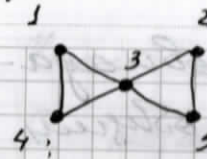
Опр. Цепь - это маршрут, все ребра которого различны.

Опр. Простая цепь - цепь, в которой все вершины различны.

Опр. Цикл - замкнутая цепь.

Опр. Простой цикл - замкнутая простая цепь.

Пр:



1, 3, 1, 4 - маршрут, но не цепь

1, 3, 5, 2, 3, 4 - цепь, но не простая цепь

1, 4, 3, 2, 5 - простая цепь

1, 3, 5, 2, 3, 4, 1 - цикл, но не простой цикл

1, 3, 4, 1 - простой цикл.

Опр. Ациклический граф - граф без циклов.

Опр. Для орграфов:

цепь — путь
цикл — контур

Опр. Вершина и достижима из вершины v , если $\exists (v, u)$ - цепь (путь).

Лемма 1 Любой (u, v) -маршрут ($u \neq v$) содержит простую (u, v) -цепь.

Лемма 2 Любой цикл содержит простой цикл.

Лемма 3 Объединение двух несовпадающих простых (u, v) -цепей содержит простой цикл.



Упр. 4 Если C и D два несовпадающих простых цикла с общей ребром e , то граф $(C \cup D) \setminus \{e\}$ также содержит простой цикл

$D \setminus e$ и $C \setminus e$
это
две несовп.



(u, v) -цели \Rightarrow по упр. 3 их одвержение содержит простой цикл.

Опр.: Две вершины в графе наз-ся связанными, если они соединены маршрутом / по упр. 1 — простой целью).

Опр.: Граф связный, если любые две его вершины связаны.

Опр.: Компонента связности графа — это любой максимальный связный подграф (максимальный в том смысле, что не содержится в связном подграфе с большим числом элементов).

Число компонент связности графа G обозначается $[k(G)]$. Т.о. граф G связный $\Leftrightarrow k(G) = 1$.

Опр. Если $k(G) > 1$, то граф G — несвязный

Метрические хар-ки графа

Опр.: Длина маршрута — это кол-во ребер в маршруте (с повтор.)

Опр.: Расстояние м-у вершинами u и v обозн $[d(u, v)]$ — это длина кратчайшей простой (u, v) -цели.

Самая кратчайшая сеть наз-ся геодезической. Если $\nexists (u, v)$ -цели, то $d(u, v) = \infty$.

Опр.: Эксцентриситет вершины u
 $[e(u)] = \max_{v \in V} d(u, v)$

Опр.: Диаметр графа G

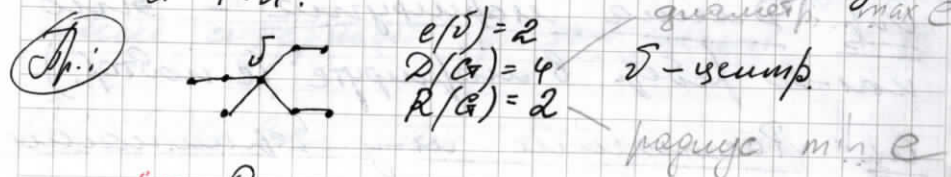
$$[D(G)] = \max_{u \in V} e(u) = \max_{u \in V} \max_{v \in V} d(u, v)$$

Опр.: Радиус графа G

$$[R(G)] = \min_{u \in V} e(u) = \min_{u \in V} \min_{v \in V} d(u, v)$$

Опр. Центр графа G - это вершина v т.ч. $e(v) = R(G)$.

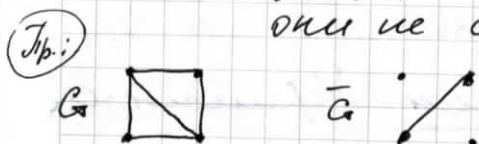
Задача нахождения центральных вершин часто возникает на практике. Например, населенные пункты соединены дорогами. Требуется оптимально (по времени или стоимости проезда, по длине дороги...) разместить больницу, школу и т.п.



§7 Операции над графами

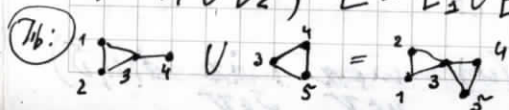
① Дополнение графа $G(V, E)$ - это граф $\bar{G}(V, E_1)$, т.ч.

$V_1 = V$; $E_1 = \bar{E}$ т.е. любые две вершины смежны $\bar{G} \iff$ они не смежны в G



② Объединение графов $G_1(V_1, E_1)$ и $G_2(V_2, E_2)$ это граф $G(V, E)$, где

$V = V_1 \cup V_2$; $E = E_1 \cup E_2$.

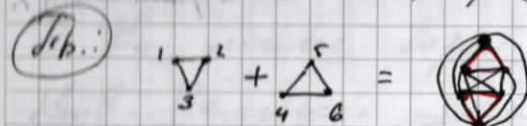


Пр.: $\bigvee_3^3 \cup \bigwedge_4^5 = \bigtriangleup - \#$

③ Соединение графов $G_1(V_1, E_1)$ и $G_2(V_2, E_2)$ это граф $G(V, E)$, где

$V = V_1 \cup V_2$

$E = E_1 \cup E_2 \cup \{e = (v_1, v_2) \mid (v_1 \in V_1) \wedge (v_2 \in V_2)\}$



④ Удаление вершины v из $G(V, E)$ есть граф $G(V, E) \setminus \{v\} = G_1(V_1, E_1)$, где

$V_1 = V \setminus \{v\}$

$E_1 = E \setminus \{e = (v_i, v_j) \mid (v_i = v) \vee (v_j = v)\}$
 удаляем все рёбра, инцидентные v .

⑤ Удаление рёбра e из $G(V, E)$

есть граф $G(V, E) \setminus \{e\} = G_1(V, E_1)$, где

$V_1 = V$

$E_1 = E \setminus \{e\}$

⑥ Добавление вершины $V_1 = V \cup \{v\}$
 $E_1 = E$

⑦ Добавление рёбра $V_1 = V$
 $E_1 = E \cup \{e\}$

⑧ Стенение подграфа K

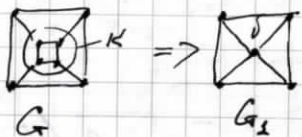
графа $G(V, E)$ есть граф $G_1(V_1, E_1)$, где

$$V_1 = (V \setminus K) \cup \{\delta\}$$

$$E_1 = (E \setminus \{e = (u, v) \mid (u \in K \wedge v \in K) \text{ или } (u \notin K \wedge v \notin K)\}) \cup \{e = (u, \delta) \mid u \in S\}$$

$\begin{matrix} \text{все ребра, инцидентные} \\ \text{вершинам из } K \end{matrix} \qquad \begin{matrix} \text{или-то} \\ \text{вершин} \\ \text{смежных} \\ \text{вершинам из } K \end{matrix}$

Пр.:



⑨ Расширение вершины δ графа

$G(V, E)$ есть граф $G_1(V_1, E_1)$, где

$$V_1 = V \cup \{\delta'\}$$

$$E_1 = E \cup \{e = (\delta, \delta')\} \cup \{e = (u, \delta') \mid u \in S(\delta)\}$$

или-то
вершины,
смежные
вершине δ

Пр.:



§8 Специальные виды графов

Опр.: Граф G называется полным, если любые две его вершины смежны.

K_n - полный граф на n вершинах.

Пр.:

K_2

K_3

K_4

K_5

Лемб. Число ребер в полном графе равно $\frac{n(n-1)}{2}$

■ ан. ①. \square число графов.

в K_n макс. возм. число ребер...

Лемб. Полный граф евл. регулярным.

■ Действительно, степень любой верш. равна $(n-1) \Rightarrow$ степени всех вершин равны \Rightarrow граф регулярный

Опр.: Граф G называется пустым, если в нем нет ребер.

$[P_n]$ - пустой граф на n вершинах.

α_2 - это трибнальный граф.

$[P_n]$ - простая цепь на n вершинах

$[C_n]$ - простой цикл на n вершинах (n -угольник)

Опр.: Двудольный граф - это граф $G(V, E)$ т.е. мн-во $V = V_1 \cup V_2$; $V_1 \cap V_2 = \emptyset$; $\forall e \in E$ инцидентно вершине из V_1 и вершине из V_2 . Мн-ва V_1 и V_2 называются долями графа. Если двудольный граф содержит все ребра, соединяющие мн-ва V_1 и V_2 , то он называется полным двудольным графом.

$[K_{n,m}]$ - полный двуд. граф
 $n = |V_1|$; $m = |V_2|$.

Пр.:



$K_{3,3}$



Двуд. граф

Теорема

Кенингс и критерий двудольности графа 1936г.

Граф является двудольным тогда и только тогда, когда все его циклы четной длины.

■ Необходимость: от противного.

Пусть $G(V_1, V_2; E)$ - двудольный граф; $v_1, v_2, \dots, v_{2k+1}, v_1$ - цикл нечетной длины. Пусть $v_1 \in V_1 \Rightarrow v_2 \in V_2 \Rightarrow \dots$
 $\dots v_{2k+1} \in V_2 \Rightarrow v_1, v_{2k+1} \in V_1 \Rightarrow$ в графе есть ребро, соединяющее две вершины из V_1 - противоречие с двудольностью.

Достаточность: Не ограничивая общности м-о считать, что G - связный граф (т.к. каждому компоненту связности м-о рассматривать отдельно). Пусть граф G с числом вершин $n > 1$ не имеет циклов нечетной длины.

Выберем вершину $v \in V$ произвольно. Построим разбиение $\{V_1, V_2\}$ мн-ва V ($V_1 \cup V_2 = V$; $V_1 \cap V_2 = \emptyset$) по след. правилу: произвольная вершина и графа G не имеют

в V_1 , если расстояние $d(u, v)$ четно,
и в V_2 , если это расстояние
нечетно.

Остается р-ть, что любые две
вершины в каждой из частей V_1 или
 V_2 не соединены ребром. Пусть
напротив, \exists две смежные вершины
 x и y , принадлежащие одному из
этих мн-в. Тогда ни одна из
них не совпадает с v , т.к. $v \in V_1$,
а все смежные ей вершины $\in V_2$
(как вершины, находящиеся
на расстоянии 1).

Пусть X - кратчайшая
 (x, v) -цепь; Y - кратчайшая
 (y, v) -цепь; v_1 - первая, считая
от v , из общих вершин этих
цепей (может оказаться, что
 v_1 совпадает с v). Тогда цепь
 XY состоит из двух подцепей

(x, v_1) и (v_1, v) , а цепь Y —
из (y, v_1) и (v_1, v) . Очевидно, что
длины цепей (x, v_1) и (y, v_1)
одного характера четности
(т.к. x и $y \in$ одному из мн-в
 V_1 или V_2). Но тогда объединение
цепей (x, v_1) , (y, v_1) и
ребра xy является
замкнутой четной длиной. \square

Следствие Граф двудольный \Leftrightarrow
все его простые циклы четной
длины.

Опр.: Звезда — полный двудольный
граф, в котором мощность
одной из частей равна 1.

$K_{1,n}$



Замечание: Скелет многогранника



тетраэдр



куб

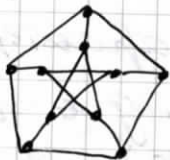
и др.

Численные графы:

Графы Куратовского:



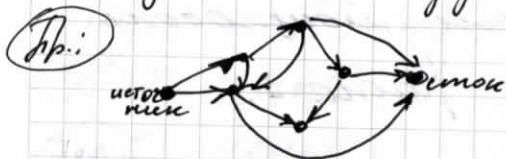
Граф Петерсона:



Граф Давида:



Опр.: Сеть — это ориграф, в котором имеется один источник, т.е. узел только с исходящими дугами и один сток, т.е. узел только с входящими дугами.



§9 Способы (представления) задания графов

При численном решении задач на вычислительных машинах граф должен быть представлен дискретным способом. Выбор конкретного представления

определяется требованиями конкретной задачи. Могут использоваться некоторые модификации или комбинации этих представлений.

Представления графов в памяти компьютера являются важным фактором в скорости выполнения операций над графами.

Далее будет рассмотрен граф (или ориграф) на n вершинах и m ребрах.

После некоторой модификации эти представления могут быть применены и для псевдо- и мультиграфов.

п.1 Матрица смежности.

Матрица смежности — это способ описать квадратная булева (состоит из 0 и 1) матрица размера $n \times n$, отражающая смежность вершин

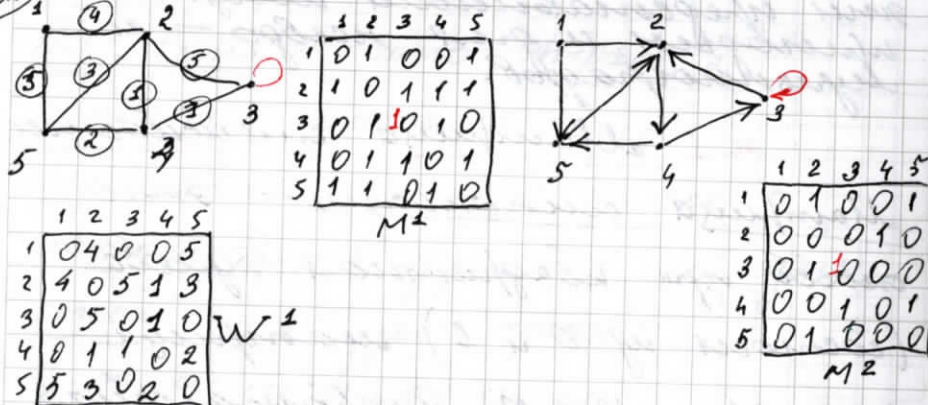
$$M_{ij} = \begin{cases} 1, & \text{если вершина } v_i \\ & \text{смежна с вершиной } v_j; \\ 0, & \text{если вершины } v_i \text{ и } v_j \\ & \text{не смежны.} \end{cases}$$

Объем памяти пропорц-н n^2

Для неориентированного

графа матрица смежности
симметрична относительно
главной диагонали \Rightarrow достаточно
хранить только числа на
главной диагонали и выше \Rightarrow
сокращаем объем памяти
почти вдвое.

Пр:



Если рассматривать взвеш. граф,
в котором каждому ребру сопост.
число, то он может быть
представлен матрицей весов W :
это обобщение матрицы
смежности: $W_{ij} = \begin{cases} \text{вес ребра,} \\ \text{если ребро } \exists; \\ 0 - \text{иначе} \end{cases}$

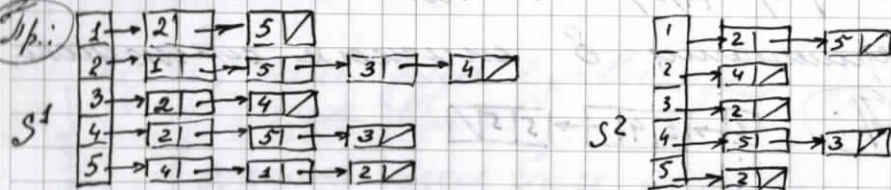
"+" : операция поиска ребра (i, j)
осуществляется за "5 шагов"
независимо от n и m .
- : большой объем памяти

Матрицы смежности удобнее
использовать для "плотных"
графов ($m \approx n^2$)

1.2. Списки смежности

Списки смежности — это
вектор (массив) S длины n ;
элементами массива являются
линейно связанные списки —
по одному на вершину. Для
каждой вершины $i \in V$ список
смежных вершин $S[i]$ содержит
в произвольном порядке все
смежные с ней вершины.

Пр:



Для ориентированных графов
сумма длин всех списков

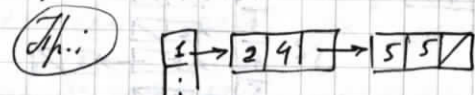
сменных узлов равна общему числу дуг. дуге (u, v) соответствует элемент в списке $S[u]$.

Объем памяти пропорц. $(n+m)$.

Для неориентир. графов эта сумма равна удвоенному числу ребер, т.к. ребро (u, v) порождает элемент в списке смежных вершин как для вершины u , так и для вершины v .

Объем памяти пропорц. $(n+2m)$.

Списки смежных вершин также удобны для хранения взвешенных графов. В этом случае вес ребра (u, v) хранится вместе с вершинами в списках смежности.



"+" : операции добавления и удаления ребра производится за константное число шагов, не завис. от n и m .

— : если надо узнать, есть ли в графе ребро (u, v) , то надо просмотреть весь список $S[u]$ в поисках вершины v .

Списки смежности удобнее использовать для "разреженных" графов ($m \ll n^2$)

2.3. Матрица инцидентности

Матрица инцидентности — это двема (а для ориграфов — составная из 0, 1, -1) матрица H размера $n \times m$, отражающая инцидентность вершин и ребер.

Строки — вершины
Столбцы — ребра

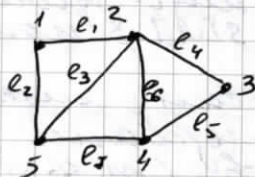
для ребер $H_{ij} = \begin{cases} 1, & \text{если вершина } v_i \text{ инцидентна ребру } e_j \\ 0, & \text{иначе} \end{cases}$

для ориграфов $H_{ij} = \begin{cases} 1, & \text{если дуга } e_j \text{ выходит из узла } v_i \\ 0, & \text{если узел } v_i \text{ и дуга } e_j \text{ не инцидентны} \\ -1, & \text{если дуга } e_j \text{ входит в узел } v_i \end{cases}$

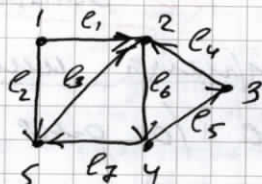
Объем памяти пропорц. $n \times m$

Пр.:

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
1	1	1	0	0	0	0	0
2	1	0	1	1	0	1	0
3	0	0	0	1	1	0	0
4	0	0	0	0	1	1	1
5	0	1	1	0	0	0	1



	e_1	e_2	e_3	e_4	e_5	e_6	e_7
1	1	1	0	0	0	0	0
2	1	0	1	1	0	1	0
3	0	0	0	1	1	0	0
4	0	0	0	0	1	1	1
5	0	1	1	0	0	0	1



1.4. Массив ребер

Массив ребер (дуг) (или список ребер (дуг)) — это вектор (массив) K длины m , элементами которого являются ребра (дуги) графа. При этом каждое ребро задается парой инцидентных ему вершин.

Объем памяти пропорцион. $(2m)$.

Этот способ задания графа позволяет легко описать петли и кратные ребра.

Пр.:

1	12
2	15
3	52
4	23
5	34
6	24
7	45

1	12
2	15
3	52
4	32
5	43
6	24
7	45

5.10 Обходы графов

Опр.: Обход графа — это некоторое систематическое пере枚举ение его вершин (и/или ребер).

В основе многих алгоритмов на графах лежат алгоритмы, использующие локальную информацию (или-да смежности).

Алгоритмы поиска вершин, достижимых из S вершины S .

Поиск в ширину

Перебираются все достижимые из S вершины в порядке возрастания расстояния от S .

Здесь "расстояние" — это число ребер

Поиск в глубину

Стратегия такая: идти "зигзаг", пока есть непройденные исходящие ребра; возвращаться и искать новый путь, когда таких ребер нет.

n. l. Аморфизм

Дано: Граф $G(V; E)$ представляет
списком смежности S

Получить: Посл-ть вершин обхода Р.

V-мн-во Зерини градо

$\mu_{[1..n]}$ - вектор (масса) меток вершин

Т - стек или перегородка, куда падают некоторые вершины.

- ① а) Обнулим сектор (массив) меток —
в начале все значения не отличаются

- ② а) Выбираем из V произв. элементу s - начало отсчета;
б) Помещаем s в T ;
в) Помещаем s

- 3) циклы
- а) Извлекаем из **T** вершину ~~и~~ **u**.
 - б) Возвращаем эту вершину в качестве очередной — т.е. кладем **u** в **P**.
 - в) для всех вершин **x** из списка смежности **S[u]**:
 - если **x** не помечен:
 - помечаем **x** в **T**
 - помещаем **x**

пока $T \neq \emptyset$

Если T - очередь (FIFO), то алгоритм реализует поиск в ширину.



- $$f) T \rightarrow \boxed{\delta} \rightarrow$$

- 2) $P \square 3$

$$T \rightarrow [r] \rightarrow [w] \rightarrow$$

- 3) $p \mid \boxed{s} \boxed{w}$

T → x → t → \$

- 4) P

s	w	r
---	---	---

$T \rightarrow \boxed{v} \rightarrow \boxed{x} \rightarrow \boxed{t} \rightarrow$

- 5) P

s	w	r	t
---	---	---	---

T → 4 → 2 → X →

- 6) P

s	w	r	t	x
---	---	---	---	---

$T \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow$

- 7)

P	S	W	R	Z	X	U
---	---	---	---	---	---	---

$$T \vdash y \rightarrow \boxed{y} \rightarrow$$

- 2)

P	s	w	r	t	x	5	u
---	---	---	---	---	---	---	---

$$T \rightarrow [y] \rightarrow$$

- g)

P	s	w	r	t	x	j	u	y
---	---	---	---	---	---	---	---	---

$$T = \emptyset$$

Если T-стек (LIFO), то алгоритм реализует поиск в глубину

- $$1) T: S$$

- 2) $P: S$

- g) $P: \frac{3}{4}$

- 4) $P: \overline{S} \vee \overline{V}$

- 5) P: $S \sim \delta w$
T: $\begin{array}{|c|c|} \hline x & 1 \\ \hline \end{array}$

- 6) $P: \underline{sr} \vee w x$

- 7) $P: \underline{sr \vee w} \ x \ y$

- 2) $P: s r \sigma w x y u$

- g) $p: s r \sigma w x y t$

- [illegible]

п. 2. Д-во правильности ал-ма (обоснование)

Теорема Если граф G связен, то поиск в ширину и поиск в глубину обойдут все вершины по одному разу.

■ 1) Есть ли обход вершин Обходятся только вершины, попавшие в T . В T попадают только непосещенные вершины. При попадании в T вершина отмечается \Rightarrow любая вершина попадет в P (будет обойдена) не более 1 раза.

2) Конечность алгоритма. Всего в T может попасть не более n вершин (попадают непосещенные и сразу отмечаются). На каждом шаге 1 вершина удаляется из $T \Rightarrow$ алгоритм завершит работу не более чем через n шагов.

3) Обход ВСЕХ вершин. От противного. Пусть алгоритм завершил работу и вершина b не обойдена $\Rightarrow b$ не попала в $T \Rightarrow$ она не была отмечена \Rightarrow все вершины, смежные с b не были обойдены и отмечены (иначе бы b попала в T) \Rightarrow \forall вершины, смежные с непосещенными — непосещены. \neg G связен $\Rightarrow \exists (s, b)$ — путь \Rightarrow вершина s не отмечена — противоречие, т.к. s была отмечена на 1-м шаге. ■

п. 3. Время работы алгоритма (трудоемкость)

Опр. Время работы алгоритма — число "элементарных шагов". (элемент. шаг — команда, требующая фиксир. числа операций).

(Пр.: "поместить вершину s в T " — эл. шаг. (не завис. от размера графа))

Опр.: Если время X пропорционально y , т.е. $X \leq K \cdot y$, где K — const, то говорят, что $X = O(y)$.

Оценим время работы (или трудоемкость) нашего алгоритма.

- 1) Инициализация (обнуление массива меток) требует $O(n)$ шагов
- 2) Каждая операция с T требует фиксир. число шагов. Всего на операции с очередью уйдет $O(n)$ шагов (каждая вершина попадет в T 1 раз).
- 3) Список смежных вершин S просматривается лишь когда вершина убивается из T \Rightarrow каждый список $S[i]$ просматривается не более 1 раза. Сумма длин всех списков

равна $O(m)$ (m для оригра, $2m$ для неориент. графа) \Rightarrow всего на обработку S уйдет $O(m)$ шагов.



Общая трудоемкость

$$O(n) + O(n) + O(m) = \Theta(n) \\ = K_1 n + K_2 n + K_3 m = \boxed{O(n+m)}$$

Замечание Трудоемкость алгоритма может зависеть в завис. от способа представления исходных данных.

Часть 2. Деревья

§1. Описание деревьев

Деревья заслуживают отдельного и подробного описания по двум причинам:

1) деревья являются простейшим классом графов. Для них есть-ся много св-ва, которые не всегда выполняются для графов в общем случае. Применительно к деревьям много г-ва оказываются намного проще. Выводя какие-то теоремы при решении г-х теорем графов, целесообразно сначала их проверить на деревьях.

2) деревья являются самым распространённым классом графов применяемым в программировании, причем в самых разных ситуациях.

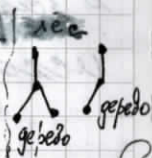
Опр Ациклический граф (лес) —

это граф без циклов.

Опр Дерево — связный ациклический граф.

Лем. Средиющие утверждение св-ва дерева эквивалентно:

- 1) Граф G — дерево
- 2) Любые две вершины графа G соединены единственной простой цепью



③ Граф G связный и удаление любого ребра ведет к нарушению связности.

④ Граф G связный и $m = n - 1$

⑤ Граф G ациклический и $m = n - 1$

⑥ Граф G связный и добавление любого ребра ведет к образованию ровно одного простого цикла

■ г-во см. учебника Новикова стр. 235 ■

Следствие В любом дереве ($n > 1$) имеются по крайней мере две висятые вершины.

■ Дерево — связный граф \rightarrow

$\forall v \in V \quad d(v) \geq 1$. Далее от противного. Пусть $\forall v_i \in \{v_1, \dots, v_{n-1}\} \quad d(v_i) \geq 2$. (т.е. висятые вершины)

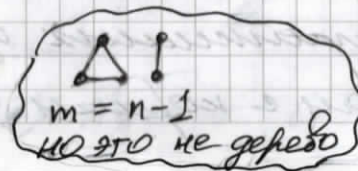
По лемме о рукопожатиях:

$$2m = \sum_{i=1}^n d(v_i) \geq 2(n-1) + 1 = 2n - 1.$$

Но т.к. в дереве $m = n - 1$, то

$$2m = 2n - 2 \rightarrow$$

$$2n - 2 \geq 2n - 1 \text{ — противоречие} \blacksquare$$



§2 Ориентированные деревья

Опр. Ориентированное дерево (ордередо)

это ориграф со следующими свойствами:

① $\exists!$ узел v т. е. $d^+(v) = 0$.

Узел v наз-ся корнем ордереда;

② $d^+(u) = 1 (\forall u \neq v)$

③ Каждый узел достижим из корня.

Утв. Ориентированное дерево
 с n - вер-нами
 ордереда обладает след. св-ствами:

① $m = n - 1$

② если в ордереве отменить ориентацию ребер, то получится дерево — свободное дерево

③ в ордереве нет циклов

④ для каждого узла $u \neq v$ путь, ведущий в этот узел из корня.

⑤ подграф, определенный $n-1$ узлами, достижимыми из v , является ордеревом с корнем v .

⑥ если в свободном дереве можно вершину назначить корнем, то получится ордередо.

■ 1) Каждая дуга входит в какой-то узел; из п.2 определения: $\forall u \neq v$ $d^+(u) = 1 \Rightarrow m = n - 1$.

2) Пусть G - ордередо. Граф G' получится из G отменой ориентации ребер. Тогда $\forall u_1, u_2 \in V$ в G' \exists простая (v, u_1) -цепь и \exists простая (v, u_2) -цепь $\Rightarrow \forall u_1, u_2 \in V$ в G' \exists простое (u_1, u_2) -цепь $\Rightarrow G'$ - свободное; кроме того, $m = n - 1 \Rightarrow$ по п.4. утв-е о св-вах дерева: G' - дерево.

3) следует из п.2

4) от противного: если в G \exists два пути из v в $u \Rightarrow$ в G' есть цикл.

5) Пусть G_u - подграф, опреде-

минимальной множества узлов,
 достигаемых из u . \Rightarrow
 $\forall \alpha \in X_{G_u} \setminus X^+ d_{G_u}^+(u) = 0$,
 иначе узел u был бы достигаем
 из какого-то узла v графа $G_u \Rightarrow$
 в G_u , а значит $u \in G$ имел бы
 контур, что противоречит п.3.
 Далее, $\forall w \in G_u \setminus \{u\} d^+(w) = 1$,
 т.к. $G_u \in G$. Все вершины G_u
 достигаемы из u по построению
 \Rightarrow по определению G_u — ордереб.
 6) Пусть вершина v назначена
 корнем и дуги последовательно
 ориентированы "от корня" входом
 в глубину. Тогда $d^+(v) = 0$ по
 построению; $\forall u \in G \setminus \{v\} : d^+(u) = 1$,
 т.к. входящая дуга появляется
 при первом посещении узла,
 а узлы посещаются по одному
 разу. Все вершины достигаемы

из корня, т.к. вход в глубину
 посещает все вершины связного
 графа. Т.о. по определению
 получаем ордереб. ▣

Опр. Вершина ордереба u т.ч.
 $d^-(u) = 0$, называется источником.
 Путь из корня в лист — это ветвь.
 Длина (число дуг) наибольшей
 ветви — высота ордереба.

Уровень узла — это расстояние
 от корня до этого узла.

Уровень корня равен 0.

Узлы одного уровня образуют уровень.

Наряду с "растительной"
 применяется и "генеалогическая"
 терминология.

Узлы, достигаемые из узла
 u , — потомки узла u . Если
 \exists дуга (u, v_1) , то u_1 — отец (роди-
тель) узла u_2 ; u_2 — сын узла u_1 .
 Сiblings одного узла — братья

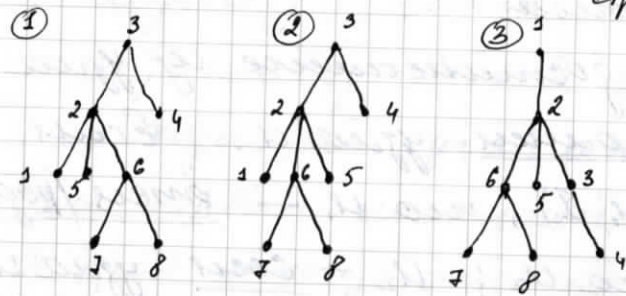
Опр (эквивалентное индуктивное определение порядка)

① 1 узел является ордеревом;
всё же — корень.

② Пусть δ — это узел; T_1, T_2, \dots, T_k — ордеревья с корнями $\delta_1, \delta_2, \dots, \delta_k$.
Можно построить новое ордерество, сделав δ родителем узлов $\delta_1, \dots, \delta_k$.
В этом случае δ — корень;
 T_1, \dots, T_k — поддеревья этого корня.

Опр Упорядоченное ордерество — ордерество, в котором отношение порядка поддеревьев фиксировано.

Пр. (обычно ориентация дуг в ордерестве выражается не пишется)



Как упорядоченные деревья все

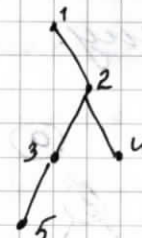
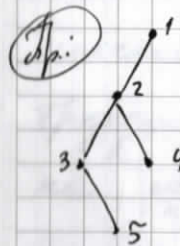
три графа различны.

Как ориентир. деревья $① \neq ② \neq ③$

Как свободные деревья все они

изоморфны: $① \approx ② \approx ③$

Опр. Бинарное (двоичное) дерево — это или пустое дерево, или ордерество, у которого любой узел или не имеет сыновей, или имеет либо левого сына, либо правого сына, либо обоих.



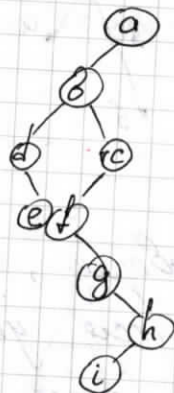
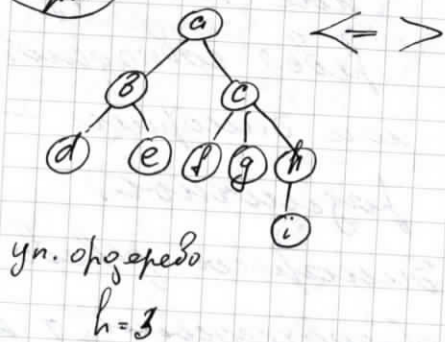
Как упорядоченные эти ордерества совпадают, как бинарные — различны.

В общем случае, бинарное дерево не является упорядоченным, т.к. каждый сын определен или как левый, или как правый.

§3 Бинарные деревья

- 1) Всеми свободные (неориентир.) дерево м-о ориентировать, назначив один из узлов корнем.
- 2) Всекое ордерено м-о произвольно упорядочить.
- 3) Всеми упорядоченные ордерено м-о представить бинарными деревьями: правая связь - к старшему брату; левая связь - к младшему сыну.

Пр:

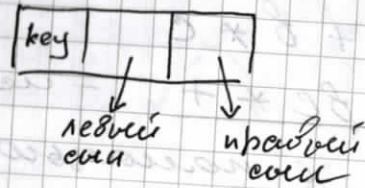


П.о. Наименование работы с деревьями м-о свести к работе с бинарными деревьями.

Чем хороши бинарные деревья?

- 1) Для их представления м-о использовать более экономичные стр-ры, например

а) Списочная структура



- 2) Большинство алгоритмов работы с бинарными деревьями имеют меньшую суммарную трудоемкость, чем алгоритмы на произвольных деревьях (структурах).

п.1. Обходы бинарных деревьев

Прямой обход

Понастоя 3 корень
обойти левое поддерево
обойти правое поддерево

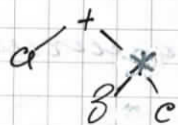
Симметричный обход

Обойти левое поддерево
Понастоя 3 корень
обойти правое поддерево

Концевой
обход

Обойти левое поддерево
Обойти правое поддерево
Понасиль в корень

Пр:



Прямой обход $+ a * b c$

Симметр. обход $a + b * c$

Концевой обход $a b c * +$ — исп-се

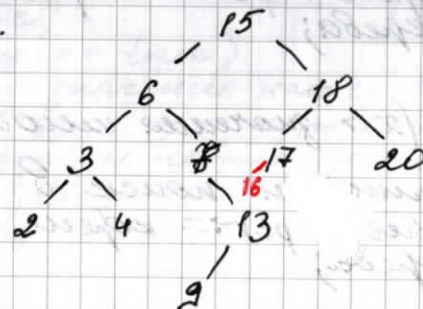
В трансляторах: с помощью
стека м-о вычислить за 1 проход.
Такое представление наз-ся
обратной польской записью.

1.2. Бинарное дерево поиска (дерево сортировки)

Опр.: Бинарное (двоичное) дерево
поиска (или дерево сортировки) —
это бинарное дерево, каждый узел
которого содержит ключ и облада-
ет следующими свойствами: значе-
ние ключа во всех узлах левого
поддерева меньше, а во всех узлах

правого поддерева больше, чем
значение ключа в узле.
(может исп-ся нестрогий
порядок).

Пр:



С-во:

симметричный обход
выдает все ключи в
возрастающем порядке.

Алгоритм поиска узла в дереве сортировки

Вход root — корень дерева
x — искомый ключ

Выход ptr — узел с ключом x
либо NULL, если
такого ключа в
дереве нет.

ptr := root; // в начале ptr совп. с корнем
key := true; // дерева


```

while (ptr != null) & (key == true)
{
    if (x < значение ключа в узле ptr)
    {
        продолжаем поиск в левом
        поддереве: ptr := корень левого
        поддерева;
    }
    else if (x > значение ключа в узле ptr)
    {
        продолжаем поиск в правом
        поддереве: ptr := корень правого
        поддерева;
    }
    else key := false // нашли!!!
}

```

Трудоемкость: т.к. длина
пути поиска не превосходит
высоты дерева h , то трудоемкость
поиска есть $O(h)$

Алгоритм вставки узла в дерево сортировки

Вход root — корень дерева
x — ключ добавляемого узла

Выход именованное дерево, если
узел с ключом x ранее
в дереве не было.

```

if (дерево пусто: root == null)
{
    создаем новый узел и делаем
    его корнем дерева
}
else
{
    ptr := root; // вначале ptr совпадает с корнем
    key := true;
    while (key == true)
    {
        if (x < значение ключа в узле ptr)
        {
            if (в узле ptr нет левого сына)
            {
                создаем новый узел с ключом x;
                присоединяем его к узлу ptr слева;
                key := false;
            }
            else
            {
                продолжаем поиск места для
                вставки слева: ptr := корень
                левого поддерева;
            }
        }
        else if (x > значение ключа в узле ptr)
        {
            if (в узле ptr нет правого сына)
            {
                создаем новый узел с ключом x;
                присоединяем его к узлу ptr справа;
                key := false;
            }
            else
            {
                продолжаем поиск места для
                вставки справа: ptr := корень
                правого поддерева;
            }
        }
        else key := false; // такой узел уже есть
    }
}

```

Обоснование!
Необходимо, что новый узел
подключается к существующему
узлу, имеющему свободную связь,
так, чтобы не нарушилось условие

дерева сортировки. Сложность вставки равна $[O(h)]$.

Алгоритм удаления узла из дерева сортировки.

Вход root — корень дерева
x — ключ удаляемого узла

Выход уменьшенное дерево

① Вызываем процедуру (метод) поиска узла с ключом x;
p₁ — найденный узел.

② if (удаленный узел p₁: p₁ ≠ null)
if (узел p₁ не имеет правого сына)

I: удаляем узел p₁ и подменяем его левое поддерево к его родителю с той же стороны.

else

① запоминаем в p₂ корень правого поддерева узла p₁;

② if (в узле p₂ нет левого сына)

① делаем левое поддерево узла p₁ левым поддеревом узла p₂;

II: удаляем узел p₁ и подменяем его правое поддерево к его родителю с той же стороны.

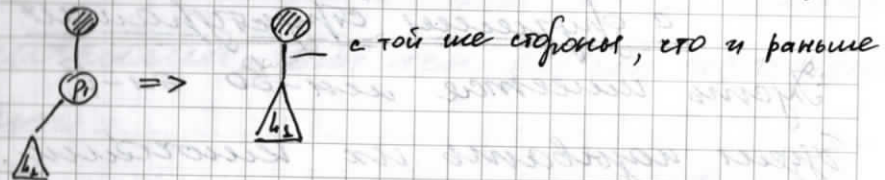
③ else

III: ① спускаемся по левому сыновьям от узла p₂ до узла p₃, левая свес которого пуста, p₄ — его родитель;

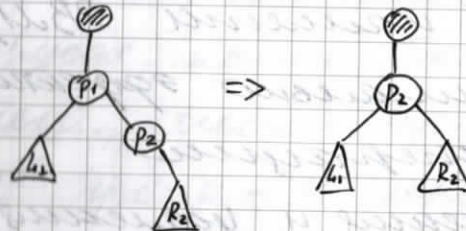
② в удаляемый узел p₁ переносим

свойство значения ключа из найденного узла p₃;
② удаляем узел p₃ и подменяем его правое поддерево к его родителю p₄ с той же стороны, то есть слева;

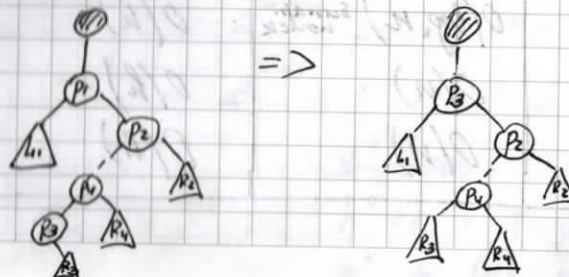
I: правая свес удаляемого узла пуста



II: правая свес удаляемого узла не пуста и ведет в узел, левая свес которого пуста



III: правая свес удаляемого узла не пуста и ведет в узел, левая свес которого не пуста



Трудность удаления узла $O(n)$
Обоснование не столь очевидно,
 как при поиске и вставке.

(самостоятельно; учтите, что
 $кноп(p_1) < кноп(p_2) < кноп(p_3)$)

Сравнение дерева сортировки с другими структурами

Пусть имеется мн-во n -в,
 будем называть их кнопками.

Число кнопок равно n .

Необходимо так представить
 это мн-во в памяти ЭВМ,
 чтобы максимально эффективно
 реализовать операции
 поиска, добавления и удаления

Кнопка	Неуп. массив	Упоряд. массив	Дерево сортировки
Найти	$O(n)$	$O(\log_2 n)$ <small>бинар. поиск</small>	$O(h)$
Добавить	$O(1)$	$O(n)$	$O(h)$
Удалить	$O(n)$	$O(n)$	$O(h)$

Дерево сортировки может быть
 неравновесно. Например, если при
 создании дерева исходные данные уже
 упорядочены, то построенное дерево
 будет право- или левостепенным.
 В этом случае $O(h) = O(n)$ — никакого
 выигрыша от исп. дерева сортировки
 нет.

Нижняя оценка для h :
 самое "узкое" бинарное дерево
 на n вершинах — т.е. на всех
 уровнях, кроме, быть может, послед-
 него, число вершин равно
 2^k , где k — № уровня. Высота
 такого дерева h^* :

$$2^0 + 2^1 + \dots + 2^{h^*} \geq n \geq 2^0 + 2^1 + \dots + 2^{h^*-1}$$

$$2^{h^*+1} - 1 \geq n \geq 2^{h^*} + 1$$

$$2^{h^*+1} - 1 > n - 1$$

$$2^{h^*+1} > n$$

$$h^* + 1 > \log_2 n$$

$$h^* \geq \log_2 n$$

высота дерева сортировки
зависит от диагонале.

$$\log_2 n \leq h < n$$

1.3. Другие специальные виды ордерированных.

Опр.: Сбалансированное дерево —
это бинарное дерево, для
любого узла которого высота
левого и правого поддеревьев



отличается не более чем на 1.

сбалансированное, но
не выровненное

Для сбалансир.
дерева:
(бу g-ва)

$$\log_2 n \leq h < 2 \log_2 n$$

Опр Выровненное дерево — это
ордерированное, все листья которого
расположены на двух сосед-
них уровнях

Выровненное дерево имеет
 $\min h$ для заданного n .

Для выровненного
бинарного дерева:
(бу g-ва)

$$\log_2 n \leq h < \log_2 (n+1)$$

Выровненные деревья сортировки
дают наименьший временной
затрат при поиске. Но добавление
или удаление узлов могут
потребовать перестройки всего
дерева \Rightarrow в худшем случае их
сложность этих операций $= O(n)$.

Сбалансированные деревья сорти-
ровки уступают по скорости
поиска выровненным деревьям
сортировки (менее чем в 2 раза).
Но их преимущество состоит
в том, что известные алгорит-
мы вставки и удаления
узлов в сбаланс. деревьях сортировки,
которые сохраняют сбаланси-
рованность и при перестройке
дерева затрачивают лишь
конечное число узлов \Rightarrow
сложность операций $O(h)$.

§3 Кратчайший остов

Опр.: Остовный подграф ^{связного} графа G — это ^{связный} подграф, содержащий все вершины графа G .

Опр. Остов — остовный подграф, являющийся деревом.

Пусть дан связный граф $G(V, E)$. Для каждого ребра графа задан неотриц. вес $w(u, v)$ (длина, цена, время и т.д.).

Задача: найти ^{связный} остовный подграф $G_T(V, E_T)$ т.ч.

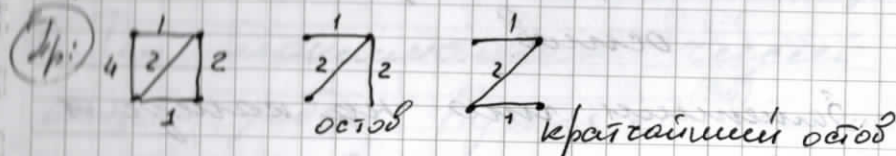
$$\sum_{(u,v) \in E_T} w(u, v) \text{ — миним.}$$

Утв. Искомый остовный подграф T является ^{остовным} деревом.

▣ T своден по построению.

Т.к. в любом цикле можно удалить одно ребро, не нарушая связности, но тем самым уменьшить

суммарный вес ребер, то T — минимален $\Rightarrow T$ — остов $\Rightarrow T$ — остов.



Алгоритм построения кратчайшего остова.

① Алгоритм Краскала

- 1) Выбираем ребро e_1 миним. веса и строим граф T_1 , полагая $V_{T_1} = V$; $E_{T_1} = \{e_1\}$
- 2) Если граф T_i уже построен и $i < \overset{\text{число вершин}}{(n-1)}$, то выбираем ребро e_{i+1} графа G , имеющее миним. вес среди ребер не входящих в T_i и не составляющих циклов с ребрами из T_i . Строим граф T_{i+1} , присоединив к T_i ребро e_{i+1} .

Обоснование.

Теорема Алгоритм Краскала строит крайний остов.

■ Заметим, что на каждом шаге i к уже построенному графу T_i присоединяется ребро, имеющее минимальный вес среди ребер, не входящих в T_i , и не образующее циклов. Пока $i < n-1$ граф T_i не связен \Rightarrow нужно ребро \exists и граф T_{i+1} можно построить.

Докажем, что граф T_{n-1} является остовом мин веса.

T_{n-1} содержит $n-1$ ребер и по построению ациклический \Rightarrow (по утв. о св-вах дерева, п.5)

T_{n-1} — дерево, т.е. остов.

Осталось показать, что этот

остов мин-го веса.

От противного: пусть T^* — остов мин-го веса, имеющий с T_{n-1} максимальное число общих ребер. Пусть $e_i = (a, b)$ — миним. по весу ребро среди ребер $\in T_{n-1}$ и $\notin T^*$ (это ребро было добавлено на шаге i). В дереве T^* есть простая (a, b) -цепь. Присоединим к ней ребро e_i — получим цикл. В этом цикле обязательно найтись ребро $e^* \notin T_{n-1}$. Заменяем в T^* ребро e^* на e_i — получим новый остов $T' = T^* - e^* + e_i$; $w(T') \geq w(T^*)$, т.к. T^* — кратчайший; с другой стороны на шаге i алгоритма мы могли бы выбрать ребро e^* (ребра e_1, \dots, e_{i-1}, e^* входят в T^* и значит не образуют

цикла). Но мы выбрали ребро $e_i \Rightarrow w(e_i) \leq w(e^*) \Rightarrow w(T') = w(T^*) - w(e^*) + w(e_i) \leq w(T^*) \Rightarrow w(T') = w(T^*) \Rightarrow T' - \text{основ миним. веса, имеющий } n-1 \text{ большее число ребер, чем } T^* \text{ — противоречие.}$

② Алгоритм Прима

Алгоритм Прима отличается от алгоритма Краскала тем, что на каждом шаге строится не просто ациклический граф, а дерево.

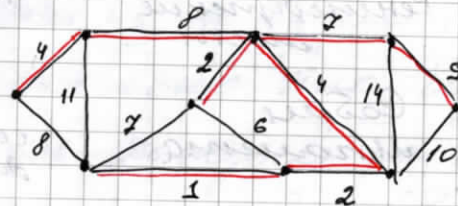
1) Выбираем ребро $e_1^{(a,b)}$ миним. веса и строим граф T_1 , полагаем $V_{T_1} = \{a, b\}$, $E_{T_1} = \{e_1\}$.

2) Если дерево T_i с $(i+1)$ -ой вершиной уже построено и $i < n-1$, то среди ребер, соединяющих вершины этого дерева с вершинами

графа G , не входящими в T_i , выбираем ребро e_{i+1} миним. веса. Строим дерево T_{i+1} , присоединив к T_i ребро e_{i+1} (вместе с новой вершиной?).

Обоснование: самоот-но.

Пр.:



Задача Штейнера

Иногда, в некоторой практической задаче, сводящейся к нахождению кратчайшего остова, можно найти лучшее решение, если разрешить наращивать мощность множества вершин.



кратчайший осто



дерево Штейнера

Часть 3. Циклы

§3. Эйлеровы циклы

Опр. Эйлеров цикл — цикл, содержащий все рёбра графа.

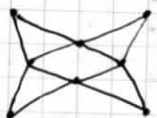
Опр. Эйлеров граф — связный граф, в котором есть эйлеров цикл.

Пр.



Кентсберские мосты

нет
эйл. ц.



Сабель
Могомета

есть
эйл. ц.

Теорема (Эйлер) Связный граф эйлеров \Leftrightarrow степени всех вершин чётные.

Необходимость. Пусть G — эйлеров граф. Эйлеров цикл, проходя через каждую вершину, входит в неё по одному рёбру, а выходит — по другому \Rightarrow каждая вершина инцидентна чётному числу рёбер эйлерова цикла. Т.к.

эйлеров цикл содержит все рёбра графа $G \Rightarrow$ степени всех вершин чётны.

Достаточность. Пусть G — связный, неориентированный граф, степени всех его вершин чётные. Начнём цикл P_1 из произвольной вершины b_1 и будем продолжать его настолько это возможно, выбирая каждый раз новое рёбро. Т.к. степени всех вершин чётны, то попав в очередную отличную от b_1 вершину, мы всегда будем иметь в распоряжении ещё не пройденное рёбро. Т.о. построение цикла P_1 может закончиться только в вершине $b_1 \Rightarrow P_1$ будет циклом.

Если окажется, что P_1 содержит все рёбра графа

G , то эйлеров цикл найден.

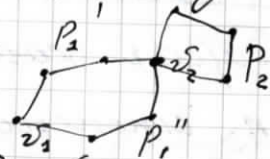
Иначе удалим из G все ребра цикла P_1 , получим граф G_1 .

Т.к. в G и P_1 все вершины четных степеней, то и все вершины графа G_1 имеют четную степень. Т.к. G связен $\Rightarrow P_1$ и G_1 имеют по крайней мере одну общую вершину — v_2 .

Теперь, начиная с v_2 , построим в графе G_1 цикл P_2 (по тому же принципу, что и P_1).

Обозначим через P_1' и P_1'' части цикла P_1 от v_1 до v_2 и от v_2 до v_1 . Тогда можно получить новый цикл $P_3 = P_1' \cup P_2 \cup P_1''$.

Если цикл P_3 не эйлеров, то продолжим построение.



В силу конечности множества E , этот процесс закончится построением эйлерова цикла. ■

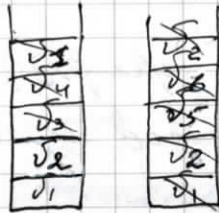
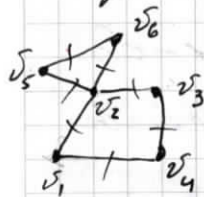
Замечание:

- ① В эйлеровом графе мн-во ребер м-о разбить на простые циклы.
- ② Д-во теоремы (достат-ны) дает основу для алгоритма поиска эйлерова цикла.

Алгоритм построения эйлерова цикла в эйлеровом графе.

Начиная с произвольной вершины v_1 , строим цепь, удаляя ребра и запоминая вершины в стеке, пока это возможно. В итоге вернемся в вершину v_1 (цепь P_1). Далее вершина v_1 выводится в качестве

первой вершины ^{удаленная из стека,} стека Эйлера цикла, а процесс повторится с вершиной, являющейся вершиной n -ой стека.



$v_1 v_4 v_3 v_2 v_6 v_5 v_2 v_1$

Оценка числа эйлеровых графов
Теорема: Эйлеровых графов почти нет.
 (Рейд)

Надо р-ть, что

$$\lim_{n \rightarrow \infty} \frac{|\text{мн-во эйлер. графов на } n \text{ верш.}|}{|\text{мн-во всех графов на } n \text{ верш.}|} = \lim_{n \rightarrow \infty} \frac{|\mathcal{E}(n)|}{|\mathcal{G}(n)|} = 0.$$

Пусть $\mathcal{E}'(n)$ - мн-во графов на n вершинах с четными степенями вершин \Rightarrow

$$\mathcal{E}(n) \subseteq \mathcal{E}'(n) \Rightarrow |\mathcal{E}(n)| \leq |\mathcal{E}'(n)|$$

любой граф из $\mathcal{E}'(n)$ м-о получить из некоторого графа

из $\mathcal{G}(n-1)$ (если добавить новую вершину и соединить её со всеми вершинами нечетной степени).

$$\Rightarrow |\mathcal{E}'(n)| \leq |\mathcal{G}(n-1)| = 2^{\frac{(n-1)(n-2)}{2}} \Rightarrow$$

$$|\mathcal{E}(n)| \leq |\mathcal{E}'(n)| \leq |\mathcal{G}(n-1)| = 2^{\frac{(n-1)(n-2)}{2}} =$$

$$= 2^{\frac{n^2-n-1}{2}} = 2^{\frac{n^2-n}{2} - (n-1)} =$$

$$= 2^{\frac{n(n-1)}{2}} \cdot 2^{-(n-1)} = |\mathcal{G}(n)| \cdot 2^{-(n-1)} \Rightarrow$$

число
всех графов
на n вершинах

$$\frac{|\mathcal{E}(n)|}{|\mathcal{G}(n)|} \leq \frac{1}{2^{n-1}} \Rightarrow \lim_{n \rightarrow \infty} \frac{|\mathcal{E}(n)|}{|\mathcal{G}(n)|} =$$

$$= \lim_{n \rightarrow \infty} \frac{1}{2^{n-1}} = 0.$$

§2 Гамильтоновы циклы

Опр.: Гамильтонов цикл — простой цикл, содержащий все вершины графа. (в кажд. верш. \neq раз!)

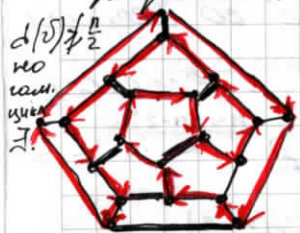
Опр.: Гамильтонов граф — связный граф, содержащий гамильтонов цикл.

Теорема Если в графе $G(V, E)$

Дирак $\forall v \in V \quad d(v) \geq \frac{n}{2}$, то

граф является
гамильтоновым.

(это лишь достат. условие)



От противного. Пусть G — не гамильтонов. Пусть v_1, \dots, v_n — вершины графа G . Добавим к G минимальное кол-во вершин u_1, \dots, u_k ($k > 0$) и соединим их со всеми вершинами v_1, \dots, v_n так, чтобы граф стал гамильтоновым. (Например, добавим $u_1, \dots, u_k \Rightarrow$ ребра (v_i, u_i) и (u_i, v_{i+1}) для $i = 1, \dots, n-1$ и ребра (u_n, v_1) и (v_n, u_n) и (u_n, v_1) дадут зам-з цикл) Получим гамильтонов граф $G' = G + u_1 + \dots + u_k$.

Пусть Z — гамильтонов цикл в графе G' . В нем обязательно

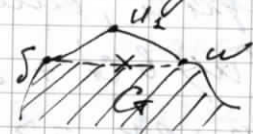
найдется ребро (v, u_1) т.е.

$(v \in G) \& (u_1 \notin G)$ (иначе бы G был гамильтоновым).

Тогда следующее ребро (u_1, w) т.е.

$w \in G$ и $w \notin \{u_1, \dots, u_k\}$ (иначе бы вершина u_1 была бы не нужна)

Кроме того, не \exists ребра (v, w) (иначе вершина u_1 была бы не нужна).



Далее, если в цикле $v, u_1, w, \dots, v', w', \dots, v$ есть вершина w' , смежная с u_1 , то вершина v' несмежна с v (иначе можно было бы построить гамильтонов цикл $v, v', \dots, w, w', \dots, v$ без вершины u_1) \Rightarrow число вершин графа G' , не смежных с v , не меньше числа вершин, смежных с w .

По условию $d(w) \geq \frac{n}{2}$ в $G \Rightarrow$ в G' $d(w) \geq \frac{n}{2} + k$. Аналогично,

в графе G' $d(v) \geq \frac{n}{2} + k$.

Общее число вершин, смежных и не смежных с v равно $(n+k-1)$.

$(n+k-1) =$ число вершин графа G' , не смежных с v + число вершин графа G' , смежных с v \geq

$$d(w) \text{ в } G' + d(v) \text{ в } G' = \frac{n}{2} + k + \frac{n}{2} + k = n + 2k$$

$\Rightarrow \underbrace{n+k-1}_{\geq n+2k} \geq n+2k \Rightarrow k+1 \leq 0 \Rightarrow k < 0$ — противоречие с тем, что $k \geq 0$.

§3 Задача коммивояжера

Имеется n городов, "расстояние" между которыми известно. Коммивояжер должен посетить n городов по 1 разу и вернуться в тот, с которого начал. Требуется найти такой маршрут движения, чтобы суммарное пройденное расстояние было минимальным.

математическая постановка

Необходимо найти кратчайший гамильтонов цикл в полном графе.

Точной алгоритмы полного перебора.

- 1) Рассмотреть все $n!$ возможных перестановок вершин полного графа — они дают $n!$ гамильтоновых циклов.
 - 2) Подсчитать для каждого циклов длину
 - 3) Выбрать кратчайший.
- Трудоёмкость $O(n!)$

$$\lim_{n \rightarrow \infty} \frac{n!}{n^k} = \infty; \quad \lim_{n \rightarrow \infty} \frac{n!}{2^n} = \infty$$

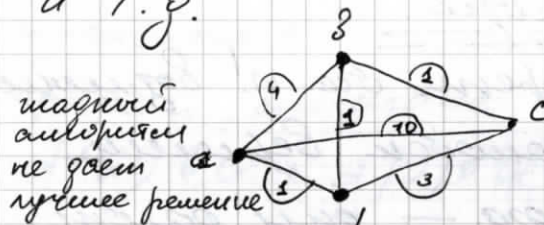
Не известен алгоритм решения с трудоёмкостью $O(n^{\text{const}})$ — полиномиальной. Также \exists та-кая трудоёмкость (экспоненциальная)

содержательная постановка

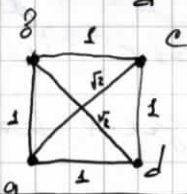
Приближенный шаговый алгоритм.

Граф имеет с произв. вершиной, выбираем самую "близкую"

и т.д.



вес ребра -
стоимость
прогн



шаговый
алгоритм
работает
точно

Гильбертов цикл

Этот цикл дост. усл.

Этот цикл дост. усл.
алгоритм
поиска Гильберта
цикла

Гильбертов цикл
встречается
редко

Гамильтонов цикл

Этот цикл дост. усл.

Заранее проверили
Э-и Гамильтоно-
вого цикла
есть ли трудной

Гамильтоновы
циклы встречаются
часто

$$\lim_{n \rightarrow \infty} \frac{|\delta(n)|}{|\gamma(n)|} = 1$$

Часть 4. Связность

§1. Компоненты связности

Связный граф имеет одну компоненту связности.

$$|k(G)| = 1$$

Любые две вершины связного графа соединены простой цепью.

Опр: Точка соприкосновения - вершина, удаление которой увеличивает число компонент связности графа.



Опр: Блок - связный граф, не имеющий точек соприкосновения.

Теорема Пусть k - число компонент связности \Rightarrow

$$n - k \leq m \leq \frac{(n-k)(n-k+1)}{2}$$

См. учебник Новикова ч. 8.

Следствие Если $m > \frac{(n-1)(n-2)}{2}$, то граф связен.

$$\frac{(n-1)(n-2)}{2} < m \leq \frac{(n-k)(n-k+1)}{2}$$

$$k=1 \Rightarrow 0 < m \leq \frac{(n-1) \cdot n}{2}$$

$k=2 \Rightarrow$ От противного. Пусть $\frac{(n-1)(n-2)}{2} < m$ и $k > 1 \Rightarrow$

$$\text{по теореме } m \leq \frac{(n-k)(n-k+1)}{2} \leq \frac{(n-2)(n-1)}{2}$$

$$\Rightarrow \frac{(n-1)(n-2)}{2} < m \leq \frac{(n-1)(n-2)}{2}$$

противоречие. \blacksquare

Опр. Мост — ребро, удаление которого увеличивает число компонент связности.

Опр. Вершинная связность графа G — это наименьшее число вершин, удаление которых приводит к несвязному или тривиальному графу.

$$\kappa(G)$$

Если $\kappa(G) = k$, то говорим, что граф G k -связен.

Опр. Реберная связность графа G — наименьшее число ребер, удаление которых приводит к несвязному или тривиальному графу.

$$\lambda(G)$$

Минимальную степень вершины в графе G обозначим $\delta(G)$.

$$\text{Теорема } \kappa(G) \leq \lambda(G) \leq \delta(G)$$

\blacksquare (1) Д-м, что $\kappa(G) \leq \lambda(G)$

Если $\lambda = 0 \Rightarrow$ граф несвязен $\Rightarrow \kappa = 0$.

Если $\lambda = 1 \Rightarrow$ есть мост \Rightarrow либо есть (.) сечение, либо $G = K_2$ (\rightarrow) $\Rightarrow \kappa = 1$.

Если $\lambda \geq 2$. Удалим $(\lambda - 1)$ ребро \Rightarrow Получим граф G_1 , имеющий мост (u, v) . Для каждого u удалим ребро, инцидентно ему вершине, отлич-

ную от u и v . Возможны 2 случая:

1) граф стал несвязным $\Rightarrow \lambda \leq \lambda - 1 < \lambda$.

2) граф остался связным \Rightarrow удалим u или v — граф станет несвязным $\Rightarrow \lambda \leq \lambda$.

② Докажем, что $\lambda \leq \delta$.

Если $\delta = 0 \Rightarrow \lambda = 0$. Пусть v т.е. $d(v) = \delta$ — вершина v имеет наименьшую степень. Удалим все ребра, инцидентные $v \Rightarrow \lambda \leq \delta$.

§2. Теорема Менгера

Заметим, что чем более связен граф, тем больше λ цепей соединяющих две вершины с другой.

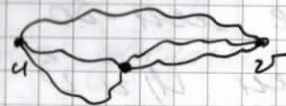
Опр. Две (u, v) -цепи наз-ся вершинно непересекающимися, если у них нет общих

вершин, отличных от u и v .

Опр. Две (u, v) -цепи называются реберно непересекающимися, если у них нет общих ребер.

Вершинно непересекающ. цепи \Rightarrow

Реберно непересекающ. цепи



Мн-во вершинно непересекающихся простых (u, v) -цепей обозначается $P(u, v)$. $|P(u, v)|$ — число цепей в $P(u, v)$.

Опр. Мн-во Разделяющее мн-во вершин где вершины u и v — это мн-во вершин, удаление которых у графа G приводит к тому, что u и v разл. в разных компонентах связности.

Опр. Разделяющее мн-во ребер (или разрез) где вершины u и v

это мин-во ребер, удаление которых из графа G приводит к тому, что u и v окажутся в разных компонентах связности.

Разделяющее мин-во вершин для вершин u и v обозначается $[S(u, v)]$. $|S(u, v)|$ — число вершин в $S(u, v)$.

Теорема Пусть u и v — две несмежные вершины в вершинной форме графа G . Тогда $\max |P(u, v)| = \min |S(u, v)|$.

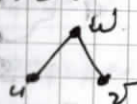
■ Сначала докажем, что $\max |P| \leq \min |S|$. Пусть P и S — произвольны. Любая (u, v) -цепь проходит через S . Если две $|P| > |S|$, то в S \exists -на две вершины, через которую прошло бы более 1 (u, v) -цепи —

противоречие с тем, что цепи из P вершинами не пересекаются.

$$\Rightarrow \forall P, \forall S \quad |P| \leq |S| \Rightarrow \max |P| \leq \min |S|.$$

Теперь покажем, что $\max |P| = \min |S|$, т.е. $\exists P$ и $\exists S$, на которых достигается равенство.

Д-во проведем ^{индукцией} совместно по числу вершин n и числу ребер m .

База: 

$$P(u, v) = \{(u, w, v)\}$$

$$S(u, v) = \{w\}$$

$$|P| = |S| = 1.$$

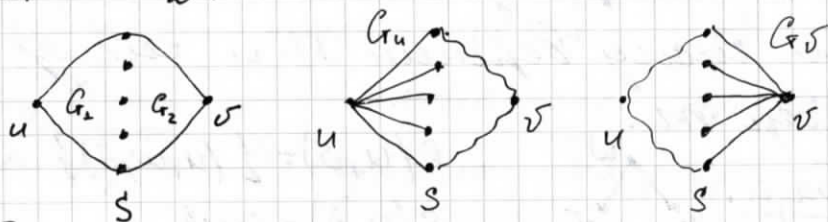
Инд-е пред-е: Пусть формула верна для всех графов с числом вершин меньшим n и/или числом ребер меньшим m .

Инд-е переход: Рассмотрим граф G с n вершинами и m ребрами.

Вершины u и v несмешные;
 S — наименьшее мн-во вершин,
 разделяющее u и v . Пусть
 $|S| = k$. Рассмотрим 3 случая.

① В S есть вершина несмешная
с u и v .

Тогда граф $G \setminus S$ состоит из
 двух нетривиальных графов
 G_1 и G_2 .

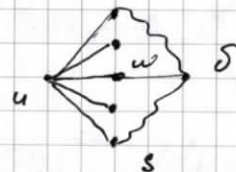


Рассмотрим два графа G_1 и G_2 ,
 полученные из G путём стягн-
 вания нетривиальных графов
 G_1 и G_2 к вершинам u и v .

П.к. G_1 и G_2 нетривиальные, то
 G_1 и G_2 имеют меньше
 число вершин и/или ребер,
 чем $G \Rightarrow$ для этих графов

верно индуктивное предполож.
 \Rightarrow в G_1 и в G_2 имеется
 к вершинам непересекающихся
 простых (u, v) -цепей.
 Набрав для отрезков этих
 цепей из G_1 в G_2 и из G_2 в G_1
 получаем к вершинам непересе-
 кающихся простых (u, v) -цепей
 в G .

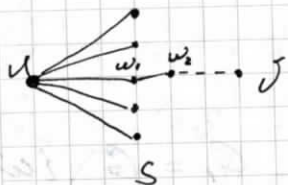
② Все вершины S смежны
с u или с v (не ограничивая
общности будем считать,
что с u); и среди вершин S
есть вершина w , смежная
одновременно с u и v .



Рассмотрим граф $G_1 = G \setminus \{w\}$.
 В нем разделяющим мн-вом для

и v является $S \setminus \{w\}$. По индуктивному предположению в этом графе имеется $|S \setminus \{w\}| = k-1$ вершинно-непересекающихся простых (u, v) -цепей. Добавим к ним цепь (u, w, v) . Она простая и вершинно не пересекается с остальными \Rightarrow в G имеется k вершинно-непересекающихся простых (u, v) -цепей.

③ Все вершины S смежны с u или с v (для u, v , с u) и среди вершин S нет вершин, смежных одновременно с u и v .



Пусть (u, w_1, w_2, \dots, v) — кратчайшая

(u, v) -цепь. ($w_1 \in S, w_2 \neq v$).

Рассмотрим граф G_1 , полученный из G склейкой вершин w_1 и w_2 в w_2 . Т.к. $w_2 \neq v$ (иначе цепь (u, w_1, \dots, v) была бы более короткой), то в G_1 цепь-во S по-прежнему является наименьшим разложением. Т.к. G_1 имеет по крайней мере $k-1$ ребро меньше, то по индуктивному предположению в G_1 \exists k вершинно-непересекающихся простых (u, v) -цепей. Но эти цепи не пересекаются и в $G \Rightarrow \blacksquare$

Упр. 1) Сформулировать (Т) Милера в реберной форме.

2) Сф-ть (Т) Милера как необх. и дост. усл. вершинной k -связности графа.

§3 Токны в сетях


Существует сеть, авторизованная
 $C(u, v)$ - какой-то автомобиль, который может проехать по дороге (u, v) за единицу времени.
 ? Как найти макс. какой-то автомобиль, который может проехать из A в B без обрат. пробок.
 ? какие дороги надо расширить (и т.д.), чтобы этот макс. увеличился.

- ① Пусть $G(V, E)$ - сеть с источником s и стоком t . Каждой дуге сети приписывается неотриц. веществ. число, т.е. заданы ф-ция
- $$C: E \rightarrow \mathbb{R}^+$$
- Если (u, v) - это дуга, то $C(u, v)$ - пропускная способность дуги (u, v) .

- ② Пусть $f: E \rightarrow \mathbb{R}$

Дивергенция ф-ции f в вершине v - это число

$$\boxed{\text{div}(f, v)} = \sum_{\{u \in V \mid (v, u) \in E\}} f(v, u) - \sum_{\{u \in V \mid (u, v) \in E\}} f(u, v)$$


то, что ушло
то, что пришло

Дивергенция источника > 0

- ③ Функция $f: E \rightarrow \mathbb{R}$ наз-ся поток в сети G , если

1) $\forall (u, v) \in E \quad 0 \leq f(u, v) \leq C(u, v)$

2) $\forall v \in V \setminus \{s, t\} \quad \text{div}(f, v) = 0$
 дивергенция потока во всех вершинах кроме источника и стока равна нулю.

- ④ Если $f(u, v) < C(u, v)$, то дуга (u, v) наз-ся ненасыщенной

- ⑤ Если $f(u, v) = 0$, то дуга (u, v) наз-ся ненапряженной

- ⑥ Величина потока f - это число $\text{div}(f, s) =: \boxed{\omega(f)}$

- ⑦ Пусть $P \subseteq E$; P - (s, t) -разрез. Разрез можно определить подмножеством вершин S и T на которые разбивается множество всех вершин V :
 $S, T \subseteq V$; $S \cap T = \emptyset$; $S \cup T = V$; $s \in S$, $t \in T$.
 В разрез P входят все дуги, соединяющие S и T .

Разрез P можно разбить на два подмножества P^+ и P^- ,

где $P = P^+ \cup P^-$;

P^+ — дуги из S в T ;

P^- — дуги из T к S .

② Сумма потоков через дуги

из P : $|F(P)| = \sum_{e \in P} f(e)$

③ Пропускная способность

разреза P : $|C(P)| = \sum_{e \in P^+} c(e)$

Теорема Форда-Фалкерсона Максимальный поток в сети равен миним. пропускной способности разреза, т.е.

∃ поток f^* :

$$W(f^*) = \max_u W(f) = \min_P C(P)$$

■ (Т) Ф.-Ф. м.б. построена из (Т) Менгера. Это г-во будет не конструктивным. (самостоятельно — см. Новиков „ДМ для проф.“)

Конструктивное г-во:

Лемма 1 $\{W(f), F(P^+) - F(P^-)\} \mid f \text{ — поток}\}$

Рассмотрим величину

$$W = \sum_{\delta \in \delta} \text{div}(f, \delta)$$



Пусть в сети есть дуга (u, v)

1) $u, v \in S \Rightarrow$ в сумму W попадут два слагаемых для этой дуги: $+f(u, v)$ для вычисления $W(f, u)$ и $-f(u, v)$ для вычисления $\text{div}(f, v) \Rightarrow$ всего 0.

2) $u \in S; v \in T \Rightarrow$ в сумму W попадёт одно слагаемое $f(u, v)$ для вычисления $\text{div}(f, u)$.

Все такие дуги образуют $P^+ \Rightarrow$ все такие слагаемые дадут сумму потоков $F(P^+)$.

3) $u \in T; v \in S \Rightarrow$ в сумму W попадёт одно слагаемое $-f(u, v)$ для вычисления $\text{div}(f, v)$. #

Все такие дуги образуют $P^- \Rightarrow$

все такие слагаемые дают
сумму потоков $-F(P^-)$

$$W = F(P^+) - F(P^-)$$

$$\text{По } W = \sum_{s \in S} \text{div}(f, s) = \overbrace{\text{div}(f, s)}^{= c(f)} + 0, \text{ т.к.}$$

f - поток и дивергенция во
всех вершинах, кроме источника
и стока, равна нулю. \Rightarrow

$$W(f) = \underbrace{F(P^+)}_{\substack{\text{величина} \\ \text{потоков} \\ \text{через } P^+}} - \underbrace{F(P^-)}_{\substack{\text{величина} \\ \text{потоков} \\ \text{через } P^-}}$$

Лемма 2 $\{\text{div}(f, s) = -\text{div}(f, t)\}$ (f - поток)

Рассмотрим разрез P , т.е.

$$S = V \setminus \{t\}; T = \{t\}. \Rightarrow$$

$$\text{div}(f, s) = w(f) \stackrel{A1}{=} F(P^+) - F(P^-) =$$

$$= 0 - \text{div}(f, t).$$

Лемма 3 $W(f) \leq F(P^+)$

$$\# w(f) \stackrel{A1}{=} F(P^+) - \underbrace{F(P^-)}_{\geq 0} \leq F(P^+) \quad \#$$

Лемма 4 $\max_f w(f) \leq \min_P C(P)$

$$\# \text{ По (1) 3: } w(f) \leq F(P^+) \Rightarrow$$

$$\max_f w(f) \leq \min_P F(P^+).$$

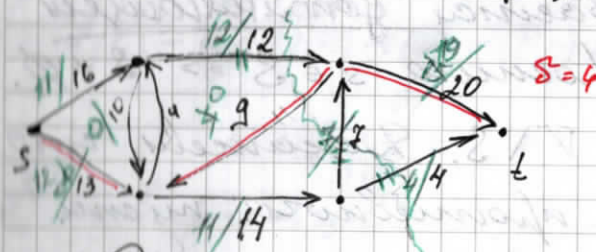
Из определения следует, что

$$F(P^+) \leq C(P) \Rightarrow$$

$$\min_P F(P^+) \leq \min_P C(P) \Rightarrow \quad \#$$

2-я теорема:

Пусть f - максимальный
поток. Докажем, что f разрез
 P , т.е. $w(f) = C(P)$.



$$F(P) = 12 + 7 + 4 + 9 = 32$$

$$C(P) = 12 + 7 + 4 = 23$$

$$w(f) = 11 + 12 = 23$$

Рассмотрим граф G' , полу-
ченный из сети G отменой
ориентации ребер.

Построим множество вершин
 S по следующему правилу:

$$S := \{u \in V \mid \exists (s, u)\text{-цены в } G'\}$$

\forall ребра $(u_i, u_{i+1}) \in (S, u)$ -цены:

$$1) f(u_i, u_{i+1}) < c(u_i, u_{i+1}), \text{ если } (u_i, u_{i+1}) \in E$$

$$2) f(u_{i+1}, u_i) > 0, \text{ если } (u_{i+1}, u_i) \in E$$

Иными словами, все (s, u) -цены

дуги в направлении от s к u

ненасыщены, а дуги против

направления имеют ненулевую

цену. Такая цена называется дополняющей

По построению $s \in S \Rightarrow S \neq \emptyset$.

Пусть $T := V \setminus S$. Докажем, что

$t \in T$. От противного: пусть

$t \in S \Rightarrow \exists$ дополняющая (s, t) -цена

Назовем ее D . Пусть

$$A(e) = \begin{cases} c(e) - f(e) > 0, & \text{если } e \text{ ориентировано} \\ & \text{от } s \text{ к } t, \end{cases}$$

$$\begin{cases} f(e) > 0, & \text{если } e \text{ ориентировано} \\ & \text{от } t \text{ к } s. \end{cases}$$

Пусть $\delta = \min_{e \in D} A(e)$.

Тогда можно увеличить величину потока на δ , изменив поток для всех дуг дополняющей цены по следующему правилу:

$$f_1(e) := \begin{cases} f(e) + \delta, & \text{если } e \text{ ориент.} \\ & \text{от } s \text{ к } t \\ f(e) - \delta, & \text{если } e \text{ ориент.} \\ & \text{от } t \text{ к } s \end{cases}$$

Проверим, что f_1 — поток:

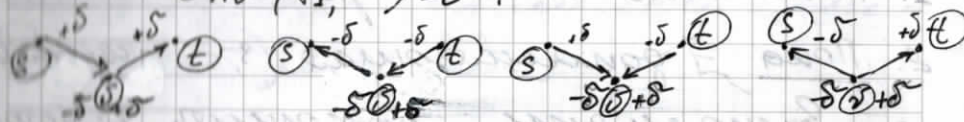
$$1) f(e) + \min_e (c(e) - f(e), f(e)) \leq f(e) + c(e) - f(e) = c(e)$$

$$f(e) - \min_e (c(e) - f(e), f(e)) \geq f(e) - f(e) = 0$$

$$\Downarrow \\ 0 \leq f_1(e) \leq c(e)$$

2) дивиденды f_1 могут измениться только в вершинах дополняющей (s, t) -цены. Но тогда, очевидно, $\forall v \neq s, t$:

$$\text{div}(f_1, v) = 0$$



Т.е. $w(f_1) > w(f)$ на величину δ —

противоречие с максимальностью
потока $f \Rightarrow t \in T$ и $T \neq \emptyset$.

Итак, S и T определяют разрез
 $P = P^+ \cup P^-$. В этом разрезе

1) все дуги e^+ из P^+ насыщены
($f(e^+) = c(e^+)$)

2) все дуги e^- из P^- ненапряжены
($f(e^-) = 0$)

(иначе S можно было бы расширить)

$$w(f) = F(P^+) - \underbrace{F(P^-)}_0 = C(P) \Rightarrow$$

P - искомый разрез. \blacksquare

Алгоритм построения максимального потока

(Метод Форда - Фалкерсона)

1. Положить поток f равным 0
2. Пока \exists дополнительная (s, t) -сеть D
3. Выполнить: дополнить D поток
 \downarrow вдоль D на δ .

Продолжительность $O(n^3 m)$

§4 Выделение компонент связности.

① В неориентированном графе
две вершины либо связаны,
либо нет.

② В ориентированном графе G
две дуги v_1 и v_2

(2.1) сильно связаны, если
 \exists путь из v_1 в v_2 и путь
из v_2 в v_1 .

(2.2) односторонне связаны, если
 \exists путь из v_1 в v_2 или \exists путь
из v_2 в v_1 .

(2.3) слабо связаны, если эти
вершины связаны в графе G' ,
полученном от исходной ориен-
тации ребер.

Опр. Компонента сильной
связности (КСС) орграфа G -

это максимальный сильно связный подграф.

① Каждая вершина графа G принадлежит только одной компоненте связности.

② Каждый узел графа G принадлежит только одной КСС и только одной компоненте связности.

Алгоритм выделения компонент связности в графе

Работа алгоритма направлена на формирование вектора M длины n — это вектор меток вершин графа G .

Элементу $M[v]$ присваивается номер той компоненты связности, которой принадлежит вершина v .

Пусть граф задан списком смежности S .

- 1) Загрузить массив M ;
- 2) $count := 0$;
- 3) цикл по всем вершинам $v \in V$

если $M[v] = 0$ то
 $count := count + 1$;
 component($v, count$);

кв
кв

основная работа выполняется этой процедурой, которая передает себе модифицируемую переменную $count$ в память.

component($v, count$)

$M[v] := count$;

цикл по всем $u \in S[v]$

если $M[u] = 0$ то
 component($u, count$);

кв

Алгоритм выделения КСС в графе

Идея алгоритма: строится "конденсат" графа G , путем сближения в один узел каждой КСС.

Основная процедура этого алгоритма также рекурсивна. Она использует стек T для организации поиска в глубину.

- ① помещаем произв. вершину в стек T и отмечаем эту вершину
- ② ②.1 берет вершину с верха-б
- ②.2 идем по всем вершинам u , смежным с v
- ②.3 если u не отмечена то
 - а) отмечаем u
 - б) помещаем u в стек
 - в) идем на шаг 2.1 (рекурсия)
- иначе u - отмечена
 - а) достаем вершину из стека до тех пор, пока не встретим u
 - б) все эти вершины "отмечаем" в одну (или) "параллельную" смежность; u и оставшиеся в стеке;
 - в) идем на шаг 2.1 (рекурсия)

ксс

квз

§5 Кратчайшие пути

Пусть задан ориграф $G(V, E)$. Каждой дуге (i, j) приписан вес:

$$C_{ij} = \begin{cases} 0, & \text{если } i=j \\ < \infty, & \text{если есть дуга } (i, j) \\ = \infty, & \text{если нет дуги } (i, j) \end{cases}$$

по аналогии с матрицей смежности

Алгоритм Форамана проверки F-я (и S)-путей

Дано: S - матрица $n \times n$

$$S_{ij} = \begin{cases} \text{true}, & \text{если } C_{ij} < \infty \\ \text{false}, & \text{если } C_{ij} = \infty \end{cases}$$

матрица смежности

матрица F - матрица $n \times n$

$$F_{ij} = \begin{cases} \text{true}, & \text{если } F(i, j) \text{ - путь} \\ \text{false}, & \text{иначе} \end{cases}$$

матрица достижимости

for ($i=1$; $i \leq n$; $i++$)

{ for ($j=1$; $j \leq n$; $j++$)

for ($k=1$; $k \leq n$; $k++$)

$$F_{jk} := S_{jk} \vee (S_{ji} \& S_{ik})$$

есть путь (j, k) u_j в u_k можно добраться через i по дугам, следовательно можно из вершин $1, 2, \dots, (i)$



$$S := F;$$

строится на основе алгоритма Форсманна

Алгоритм Флойда

Этот алгоритм находит кратчайшие пути между всеми парами вершин в графе. Эта информация хранится в матрицах H и T

$H_{ij} = \begin{cases} k, & \text{если } k - \text{первый узел,} \\ & \text{распадающийся на} \\ & \text{кратчайшие пути из } i \text{ в } j; \\ 0, & \text{если из } i \text{ в } j \text{ пути нет.} \end{cases}$

$T_{ij} = \text{вес пути из } i \text{ в } j$

Чтобы восстановить путь надо последовательно перебирать вершины:

$K_0 = i, K_1 = H_{ij}; K_2 = H_{K_1 j}; K_3 = H_{K_2 j}$ и т.д.

Дано: матрица весов C .

Получить матрицы H и T .

for ($i := 1; i \leq n; i++$) // инициализация

for ($j := 1; j \leq n; j++$) $T := C$

{ $T_{ij} := C_{ij}$
 if ($C_{ij} \neq \infty$) $H_{ij} := 0$ // нет пути из i в j
 else $H_{ij} := j$ // есть путь из i в j }

for ($i := 1; i \leq n; i++$)
 for ($j := 1; j \leq n; j++$)
 for ($k := 1; k \leq n; k++$)
 if ($(i \neq j) \wedge (T_{ij} \neq \infty) \wedge (i \neq k) \wedge (T_{ik} \neq \infty) \wedge ((T_{jk} = \infty) \vee (T_{jk} > T_{ji} + T_{ik}))$)
 { $H_{jk} := H_{ji}$ // новый путь
 $T_{jk} := T_{ji} + T_{ik}$ // длина нового пути }



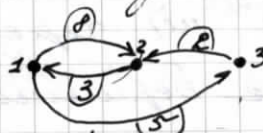
Обоснование (по индукции)

База индукции ($i=0$): в H и T — информация о кратчайших путях, не проходящих ни через какие промежуточные вершины.

Инд. предп.: перед i -й итерацией в H и T — информация о кратчайших путях, которые могут содержать промежуточные вершины $1, 2, \dots, i-1$. T_{jk} — длина кратчайшего пути от j до k ; H_{jk} — первая вершина этого кратчайшего пути.

Цикл переход: если при добавлении вершины i к диагоналю промежуточных вершин находится более короткий путь, или если это первый найденный путь, то он записывается. Т.о. после окончания цикла по i матрицы H и T содержат информацию обо всех кратчайших путях.

Пр.:



T/H	1	2	3
1	0	8/2	5/3
2	3/1	0	∞ /0
3	∞ /0	2/2	0/3

T/H	1	2	3
1	0	8/2	5/3
2	3/1	0	8/1
3	∞ /0	2/2	0/3

Сложность алгоритма Флойда $O(n^3)$

Замечание

Алгоритм Флойда используется для поиска центра графа:

$$\min_i \left(\max_j (T_{ij}) \right)$$

по строкам по столбцам

T/H	1	2	3
1	0	8/2	5/3
2	3/1	0	8/1
3	5/2	2/2	0/3

$i=2$

T/H	1	2	3
1	0	7/3	5/3
2	3/1	0	8/1
3	5/2	2/2	0/3

$i=3$

Алгоритм Дейкстры

Находит кратчайший путь между двумя данными вершинами в графе. Веса дуг должны быть неотрицательными.

Пусть необходимо найти кратчайший путь из s в t .

Алгоритм Дейкстры просматривает граф в ширину, начиная с вершины s . При этом метками отмечаются вершины, для которых рассчитаны от s (т.е. длины кратчайших путей) уже известные.

Метки могут быть двух типов: временные и окончательные. В качестве вершины s присваивается окончательная метка 0 (нулевое расстояние до самой себя). Кансрой из

оставшихся $(n-1)$ -й вершине присваивается временная метка ∞ . На каждом шаге одной вершине присваивается окончательная метка и поиск продолжается. $\{O(n)\}$

На каждом шаге метки меняются след. образом:

① каждой вершине j , не имеющей окончательной метки, присваивается новая метка =

$$= \min \{ \text{временная метка для } j, \\ c_{ij} + \text{окончат. метка для } i \}$$

где i — вершина, которой присвоена окончат. метка на предыдущем шаге. $\{O(n)\}$

Замечание 1: очевидно, что метка может помещаться только для вершины, в которую ведет дуга

из i .

Замечание 2: для возможности восстановления кратчайшей пути для каждой вершины j полезно запоминать i (если метка была изменена).

Замечание 3: каждая метка — это длина пути из s в j через вершины с окончательными метками.

② определяется наименьшая из всех временных меток, которая и становится окончательной меткой своей вершины. (в случае равенства меток выбирается любая).

③ Циклический процесс 1-2 продолжается до тех пор, пока вершина t не получит окончательной метки.

Очевидно, что этот процесс конечен.

Трудоемкость. $O(n^2)$

Обоснование. Достаточно г-ть, что окончательная метка каждой вершины — это расстояние (длина крайнего пути) из s до этой вершины. По индукции:

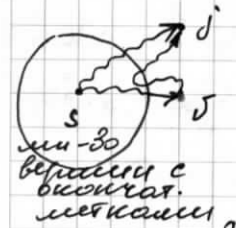
База: метка для s равна 0.

инд. предп.: пусть окончат. метки всех вершин, номер. на шагах $1, 2, \dots, k-1$ — это расстояние от s .

инд. переход. пусть на шаге k окончат. метка присво. вершине j . Обозначим эту метку t_j .

Пусть $t_j > d(s, j) \Rightarrow \exists$ кратчайший путь из s — с

от s вершина, не имеющая окончательной метки. Тогда



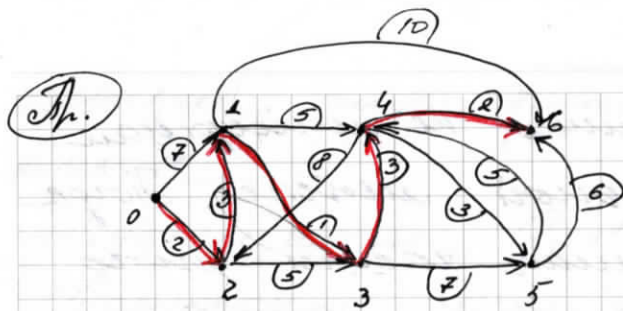
длина части этого пути от s до j также меньше t_j :

$$d(s, j) < t_j.$$

Но по замеч. 3 $d(s, j)$ должна совпасть с меткой t_j для вершины $j \Rightarrow t_j > t_j \Rightarrow$ на пути к окончат. метка р.д. присвоена вершине j — противоречие.

Замечания:

1. Алгоритм Дейкстры относится к классу "жадных" алгоритмов и является точным.
2. Алгоритм Дейкстры в отн. сн алгоритма Флойда не м.б. модифицирован на случай отриц. весов.



	0	1	2	3	4	5	6
0	0	∞	∞	∞	∞	∞	∞
1		7	2	∞	∞	∞	∞
2		5		7	∞	∞	∞
3				6	10	∞	15
4				9	13	15	
5					12		

$0 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 6$

Часть 5. Раскраска графов

Опр.: Раскраска графа — это такое приписывание цветов (натур. чисел) ко вершинам, что никакие две смежные вершины не получают одинаковый цвет.

Опр.: Хроматическое число — это наименьшее возможное число цветов в раскраске графа.

$$\chi(G)$$

Пример: v_1, \dots, v_n — работы; ребра — зав. то, что треб. одних ресурс. Работы, соотв-е вершинам одного цвета можно выполнять одновременно.

Очевидно, что $\chi(G) \leq n$, где n — число вершин в графе. Все-го вершин, покрашенных в один цвет, называется орнаментальным классом и является независимым мн-м вершин.

§1. Планырные графы

Опр.: Граф укладывается на некоторой поверхности, если его можно нарисовать на этой поверхности так, чтобы рёбра графа не пересекались.

Опр.: Планный граф — граф, который можно уложить на плоскости.

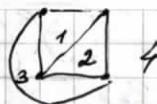
Опр.: Плоский граф — диаграмма графа, уже уложенного на пл-ть.

Опр.: Грани — область, ограниченная рёбрами плоского графа и не содержащая внутри себя вершин и рёбер. Число граней: $\chi(G)$

плоский
граф



укладка
на
плоскости



$$v = 4$$

Теорема В связном планарном графе
Формула Эйлера $n - m + v = 2$

Указание по числу ребер m .

$$m = 0 \Rightarrow n = 1, v = 1 \Rightarrow 1 - 0 + 1 = 2$$

Пусть для всех графов с m ребрами $n - m + v = 2$.

Добавим ещё одно ребро.

Возможны 2 случая:

1) добавленное ребро соединит существующие вершины:

$$n' = n; m' = m + 1; v' = v + 1 \Rightarrow$$

$$n' - m' + v' = n - m - 1 + v + 1 = n - m + v = 2.$$

2) добавленное ребро соединит существующую вершину с новой:

$$n' = n + 1; m' = m + 1; v' = v \Rightarrow$$

$$n' - m' + v' = n + 1 - m - 1 + v = n - m + v = 2.$$

Следствие 1 Если G связ. планар. и $n > 3$, то

$$m \leq 3n - 6$$

Заметим, что каждая грань ограничена по крайней мере 3-мя ребрами. Каждое ребро ограничивает не более 2-х граней \Rightarrow

$$3 \cdot v \leq 2 \cdot m \quad \leftarrow \quad \frac{3 \cdot v}{2} \leq m$$

сумма ребер по всем граням

$$2 = n - m + v \leq n - m + \frac{2}{3} m \Rightarrow$$

$$6 \leq 3n - 3m + 2m \Rightarrow$$

$$6 \leq 3n - m \Rightarrow m \leq 3n - 6.$$

Следствие 2 K_5 и $K_{3,3}$ не планарны.

$$1. K_5: n = 5, m = \frac{n \cdot (n-1)}{2} = \frac{5 \cdot 4}{2} = 10$$

Если K_5 планарен, то по след. 1:

$$m \leq 3n - 6 \Rightarrow 10 \leq 3 \cdot 5 - 6. \text{ — противор.}$$

$$2. K_{3,3}: n = 6, m = 9.$$

В этом графе нет треугольников \Rightarrow если $K_{3,3}$ планарен, то в плоской

укладке каждая грань ограничена как минимум 4-мя ребрами \Rightarrow

$$4v \leq 2m \Rightarrow 2v \leq m. \text{ По формуле Эйлера } n - m + v = 2 \Rightarrow$$

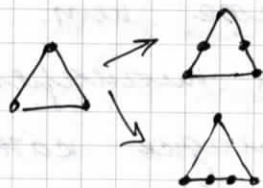
$r = 2 - 6 + 9 = 5 = > 2r = 10 \leq m = 9$
 — противоречие.

Лемма 3 В любом планарном графе Γ вершина степени ≤ 5 .

От противного. Пусть $\forall v \in V$
 $d(v) \geq 6 \Rightarrow 6n \leq \sum_{v \in V} d(v) = 2m \Rightarrow$
 $3n \leq m$. Но по сл. 1: $m \leq 3n - 6$
 $\Rightarrow 3n \leq 3n - 6$ — противоречие.

Теорема Граф планарен \Leftrightarrow
 Гамильтонов — не содержит подграфов,
 гомеоморфных K_5 или $K_{3,3}$.

(два графа гомеоморфны, если
 их можно получить из одного графа
 с помощью последовательного
 подразделения ребер)



Теорема Любой планарный
 граф можно раскрасить
 5-ю красками.

Достаточно рассмотреть
 связные графы.

Индукция по n .

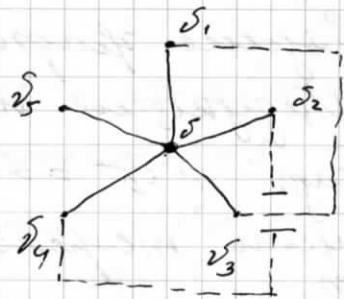
База: $n \leq 5 \Rightarrow \chi \leq n \leq 5$.

Пусть теорема верна для всех
 связных планарных графов с n
 вершинами.

Рассмотрим граф G с $(n+1)$ -й
 вершиной. По сл. 3: $\exists v \in V: d(v) \leq 5$.
 По индуктивному предположению
 $\chi(G-v) \leq 5$. Нужно раскрасить
 вершину v .

1. Если $d(v) < 5$, то в 5-раскраске
 графа $(G-v)$ v свободна для v .
2. Если $d(v) = 5$, а для $S(v)$ исполь-
 зованы не все 5 цветов, то в
 5-раскраске графа $(G-v)$ v свободна для v .
3. Если $d(v) = 5$ и все 5 цветов
 использованы для $S(v)$ (v_i —

покрашена в i -й цвет), тогда рассмотрим правильный подграф G_{13} графа $(G-v)$, порожденный всеми вершинами, покрашенными в цвета 1 или 3 в 5-раскраске графа $(G-v)$.



1) Если v_1 и v_3 принадлежат разным компонентам связности графа G_{13} , то в той компоненте, в которой находится вершина v_1 , произведем перекраску $1 \leftrightarrow 3$. При этом получим 5-раскраску графа $(G-v)$, но цвет 1 будет свободен для v .

2) Если v_1 и v_3 принадлежат 1 компоненте связности \Rightarrow

1 простая цепь, соединяющая вершины v_1 и v_3 и состоящая из вершин, покрашенных в цвета 1 и 3. В этом случае вершины v_2 и v_4 принадлежат разным компонентам связности подграфа G_{24} (т.к. G - планарен). Перекрасим вершины $2 \leftrightarrow 4$ в той компоненте связности графа G_{24} , которой принадлежит вершина v_2 . Получим 5-раскраску графа $(G-v)$, в которой цвет 2 свободен для v . ▣

§2 Задача о раскраске карты.

Задача о раскраске карты эквивалентна задаче о раскраске графа:

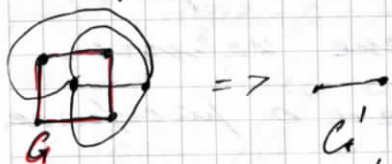
Теорема Карта G является k -раскрашиваемой \Leftrightarrow планарно-двойственный ей граф G'

является (вершинно) k -раскрасимым.

Алгоритм построения двойственного графа

- 1) поставим внутри каждой грани 1 вершину.
- 2) через каждое ребро e g -го пройти ребро двойственного графа, соединяющее вершины грани ребра e .

Заметим, что G' могут появиться кратные ребра. Для получения $\chi(G')$ кратные ребра можно сократить до одного.



§3 Алгоритм раскраски

① Точный алгоритм

Процедура $P(G)$ // G - граф.

1. Выбрать в графе G максимальное по мощности мн-во попарно не смежных вершин M (независимое мн-во вершин).
2. Покрасить вершины мн-ва M в очередную цвет..

3. Рекурсивно вызвать $P(G-M)$.

Обоснование. М-о g -то, что если $\chi(G)=k$, то, используя процедуру P , м-о получить k -раскраску графа. $\{k$ -раскрасок м.б. несколько $\}$ Алгоритм строит одну

Сложность $O(n2^n)$ ан. далее §6.2

② Приближенный алгоритм

"-": приближенный алгоритм не всегда находит точное решение.

"+": прикл. алл. более эффективен (сложность полином-на)

Приближенный алгоритм раскраски применяет ЭВРИСТИКУ (нет строгих обоснований, нет гарантии оптимальности):

начинать раскрашивать следует с вершины наибольшей степени, т.к. если такие вершины раскрасить в конце, то более вероятно, что для них не найдется свободного цвета, и придется задействовать ещё один.

① Упорядочить вершины по неубывающей степени.

② Красим, всё, что можно, в цвет 1. Среди оставшихся вершин всё, что можно, красим в цвет 2, и т.д.

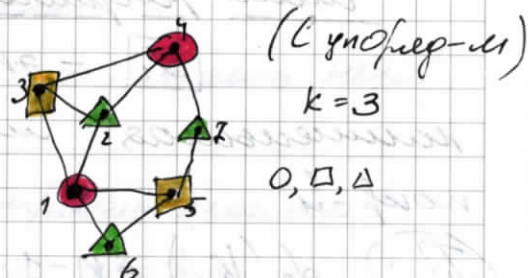
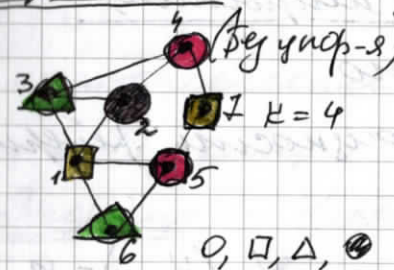


- 1) Этот алгоритм относится к "жадным" алгоритмам.
- 2) Алгоритм правильно раскрашивает граф, но может получиться, что число цветов в раскраске $\chi > \chi(G)$.

Сложность $O(n^3)$ (без учета сортировки)

Это значит, что красим всё, что можно $O(n)$ → (простоты не имеет, если нет цвета i) \Rightarrow 5 красок в i

Пример



Часть 6 Независимые и покрывающие ми-ва

Опр. Вершина ^{§1} определяется покрывает все инцидентные ей ребра. И наоборот: ребро покрывает все инцидентные ему вершины.

Опр. Ми-ва вершин, покрывающих все ребра, называется вершинным покрытием. Ми-ва ребер, покрывающих все вершины,

называется рёберным покрытием.

Опр Число вершинного покрытия

α_0 - это

наименьшая мощность вершин.
покр-я.

Число рёберного покрытия

α_1 - это

наименьшая мощность рёбер.
покр-я.

Пр. $\alpha_0(K_n) = n-1$; $\alpha_1(K_{2n}) = n$;
 $\alpha_0(K_{m,n}) = \min(m, n)$; $\alpha_1(K_{2n+1}) = n+1$.

Опр. Независимое мн-во вершин —
мн-во вершин, в котором никакие
никакие две не смежны.

Опр. Независимое мн-во рёбер —
мн-во рёбер, в котором
никакие два не смежны.

(паросочетание)

другое
название



преим

критер

Пример

Опр. Вершинное число мн-ва

β_0 - это

наибольшая мощность независ.
мн-ва вершин.

Рёберное число мн-ва

β_1 - это

наибольшая мощность незав.
мн-ва рёбер.

Теорема В неориентированного
связного графа

$$\alpha_0 + \beta_0 = n = \alpha_1 + \beta_1$$

Пример Вершинный процесс;

рёбра соединяют вершины, требующие
общий ресурс. Тогда β_0 —
это макс кол-во возможных
независимых процессов.

§2 Поиск наибольшего

независ. мн-ва вершин

(ал. точный ал. раскраски графа)

Мощность этого мн-ва — β_0 .

В силу конечности вершинности,
можно использовать алгоритм "полного перебора".

Пусть $B(V)$ - булеан мн-во V
 $|B(V)| = 2^n$

$\beta_0 := 0$

for $\forall \subseteq B(V)$

{ if (\forall - независ. мн-во) & ($|\forall| > \beta$)
 { $\beta_0 := |\forall|$; $X := \forall$; }
 }

Сложность этого ал. $O(2^n)$

При решении переборных задач
 большое значение имеет
 способ организации перебора.

§3. Доминирующие мн-во вершин.

Опр.: S - доминирующее мн-во вершин, если $\forall v \in V$; либо
 $v \in S$, либо v смежна с верш. из S .

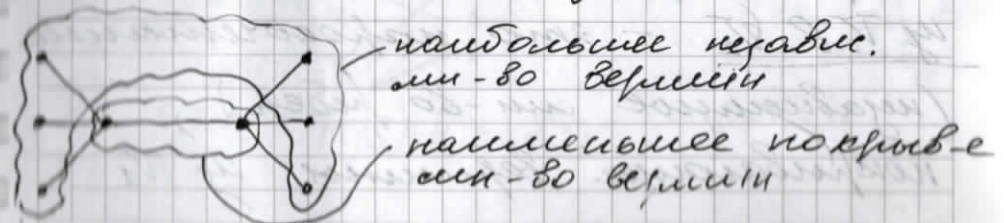
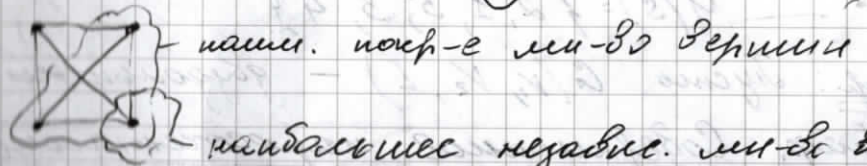
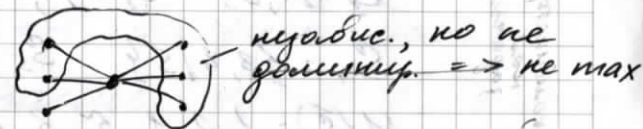
Теорема Независ. мн-во вершин
 макс. \Leftrightarrow оно доминирующее.

\Rightarrow S - максим-ное мн-во вершин. Пусть оно не доминир. \Rightarrow
 $\exists v \in V \setminus S$, т.е. v не смежна
 ни с одной вершиной из $S \Rightarrow$
 S можно расширить - противор.

\Leftarrow S - доминир. мн-во, оно не
 независ. мн-во вершин. Пусть
 оно не макс $\Rightarrow \exists v \in V \setminus S$ т.е.

v не смежна ни с оди. верш. из $S \Rightarrow$
 S не доминир. — противор.

Примеры



(Связь с комбинаторикой)

§4 Трансверсали

Пусть E — некоторое конечное множество. $S_1, S_2, \dots, S_k \subseteq E$.

$\mathcal{S} = \{S_1, \dots, S_k\}$ — семейство подмножеств E .

($S_i \cap S_j$ не обязательно \emptyset)

Опр. Трансверсали (система различных представителей)

семейства \mathcal{S} — это мн-во

$$M(\mathcal{S}) := \{s_1, s_2, \dots, s_k \mid s_i \in S_i, s_i \neq s_j\}$$

Пример $E = \{1, 2, 3, 4, 5\}$

комбинаторная задача

$$S_1 = \{2, 3\} \quad S_2 = \{1, 2, 4\} \quad S_3 = \{1, 2, 5\}$$

$$S_4 = \{3, 4, 5\} \quad S_5 = \{3, 4, 5\}$$

$$M(\mathcal{S}) = \{2, 1, 5, 3, 4\}$$

Опр. Пусть $G(V_1, V_2, E)$ — двудольный граф. Совершенное паросочетание

из V_1 в V_2 — это паросочетание

(независимое мн-во ребер), покрывающее вершины из V_1

Замечание Совершенное паросочетание экв. максим-м.

Оказывается, что задача поиска трансверсали сводится к задаче поиска совершенного паросочетания некоторого двудольного графа!

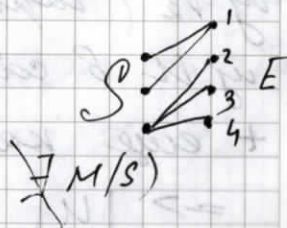
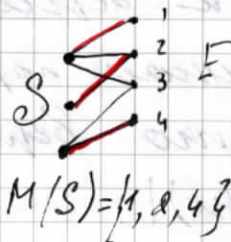
V_1 — это $\{S_1, S_2, \dots, S_k\}$

V_2 — это E

В этом двудольном графе ребро $S_i j \in E \iff j \in S_i$.

Лемма Полученный двудольный граф имеет совершенное паросочетание из V_1 в $V_2 \iff$ Семейство \mathcal{S} имеет трансверсаль.

Пример



Теорема Холла (I) В комбинаторной постановке:

Семейство S имеет трансверсаль $\Leftrightarrow \forall$ подсемейства $\{S_{i_1}, \dots, S_{i_p}\} \subseteq S$ выполняется неравенство $|\bigcup_{j=1}^p S_{i_j}| \geq p$, т.е. кол-во элементов в объединении любых p попарно-д.д. не менее p .

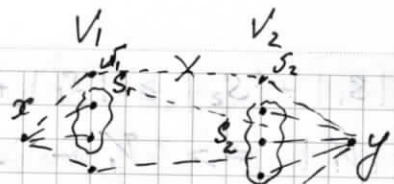
(II) В матричной постановке: Пусть $G(V_1, V_2, E)$ - двудольный граф. Совершенное паросочетание $u \in V_1, v \in V_2 \exists \Leftrightarrow \forall u \in V_1 |u| \leq |C(u)|$

\Rightarrow Пусть \exists совершенное паросочетание $u \in V_1, v \in V_2 \Rightarrow \forall u \in V_1 : \exists C(u)$ входит $|u|$ вершин $u \in V_2$ (парных к вершинам $u \in V_1$ в совершенном паросочетании) + еще какие-то вершины.
 $\Rightarrow |u| \leq |C(u)|$.

\Leftarrow Добавим в граф G две вер-

шины x и y .

Пусть x соединена со всеми вершинами $u \in V_1$, $y - u \in V_2$.



Очевидно, что совершенное паросочетание $u \in V_1, v \in V_2 \exists \Leftrightarrow \exists (V_1)$ вершинно непересекающиеся простых (x, y) -цепей. Обозначим их число $|P(x, y)|$.

Имеем: $\max |P(x, y)| \leq |V_1|$, т.к. V_1 разделяет вершины x и y .

По (I) Минера $\max |P(x, y)| = \min |S(x, y)|$, где $S(x, y)$ - наименьшее мн-во, разделяющее x и y . Очевидно, что $|S(x, y)| \leq |V_1|$. Покажем, что $|S(x, y)| \geq |V_1|$.

Пусть $S(x, y) = S_1 \cup S_2$, где $S_1 \subseteq V_1, S_2 \subseteq V_2$. Мн-во смеж. $C(V_1 \setminus S_1) \subseteq S_2$ (иначе \exists -и бы обходной путь (x, v_1, v_2, y) и $S(x, y)$ не было бы разделяющим мн-ом для x и y)

В итоге получаем:

$$|V_1 \setminus S_1| \leq |C(V_1 \setminus S_1)| \leq |S_2| \Rightarrow$$

$|S| = |S_1| + |S_2| \geq |S_1| + |V_2 \setminus S_1| = |V_2| \Rightarrow$
 $|S| = |V_2| \Rightarrow V_2$ — минимальное
 разделение на ми-во в графе $G \Rightarrow$
 Идеальн. паросочет. из V_1 в V_2 .

Задача о свадьбах Пусть имеется конеч.
 ми-во юношей, каждый
 из которых знаком с некоторым
 подмножеством девушек. Можно
 ли составить свадьбы так,
 чтобы пожениться всех юношей
 на знакомых им девушках?

§5 Максимальное
паросочетание в двудольном
графе.

Но совершенное двл.
 максимальным
 Максимальное паросочетание
 не обязательно будет совершен-
 ным.

(I) Для решения этой задачи
 можно использовать метод
 Форда-Уокера.

Даны графы $G(V_1, V_2, E)$. Добав-
 лем источник s и сток t .
 s соедин. со всеми верши. из V_1 ,
 все верши. из V_2 соедин. с t .

Ориентир. все ребра от s к t .
 Пропускные способности всех
 руж приравняем к 1.

Построим сеть G' .

Лемма Число ребер в макс
 паросочетании в двудольном
 графе равно величине макс
 потока в сети G' .

(II) Метод чередующейся цепи.
 Пусть M — некоторое паросочет.
 в графе $G(V_1, V_2, E)$.

Оп: Вершину v назовем наход.,
 если \exists инцидентное ей ребро,
 входящее в M . в паросочетание

Оп: Чередующаяся цепь относ-
ительно M — это цепь, соединяющая

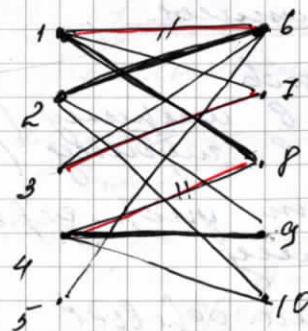
две непересекающиеся вершины, в которой чередуются ребра входящие и не входящие в M .

Имеем чередующуюся цепь P , с помощью которой увеличим паросочетание M , удалив из него те ребра, которые входят в P , и добавив ребра из P , которые не входили в M . Это новое паросочетание будем обозначать $M \Delta P$.

Теорема Паросочетание макс. \Leftrightarrow не \exists чередующейся цепи относительно этого паросочетания.

1. $M := \emptyset$
2. While \exists черед. цепь P отно-но M
 $\{ M := M \Delta P \}$

Пример.



$$M = \{(1,6), (3,7), (4,8)\}$$

$$P = 2 - 6 - 1 - 8 - 4 - 9$$

$$M \Delta P = \{(1,8), (2,6), (4,9), (3,7)\}$$

- Задача:
1. Представить в графовой постановке
 2. Найти трансверсаль или доказать что её \nexists (т. Холла)

$$S_1 = \{1, 2\}$$

$$S_2 = \{2, 5, 6\}$$

$$S_3 = \{2, 5\}$$

$$S_4 = \{1, 2, 5\}$$

$$S_5 = \{2, 3, 4, 5, 6\}$$

$$S_6 = \{2, 4, 5, 7\}$$

$$S_7 = \{2, 5, 6\}$$

$$\begin{aligned} & |S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_7| = \\ & = |\{1, 2, 5, 6\}| = 4 < 5 \\ & \text{по т. Холла} \\ & \text{Трансверсаль } \nexists \end{aligned}$$

$$S_1 = \{1, 2, 3\}$$

$$S_2 = \{1, 2\}$$

$$S_3 = \{1, 2\}$$

$$S_4 = \{1, 2, 3, 4\}$$

$$S_5 = \{1, 2, 4, 5\}$$

$$S_6 = \{1, 2, 3, 5, 6\}$$

Трансверсаль \exists