Графы, II семестр, краткое содержание

Проскуряков Иван КМБО-02-19

14 июня 2020 г.

1 Стандартные обозначения:

- Г - граф
- E - множество рёбер графа
- V - множество вершин графа
- Р - мощность множества рёбер
- В - мощность множества вершин
- F - множество граней графа
- $S : \overrightarrow{E} \to V$ - функция, возвращающая начало ребра
- $t : \overrightarrow{E} \to V$ - функция, возвращающая конец ребра

2 Определения:

1. **Граф**:
   - Определим граф без кратных рёбер и петель:
   $$E \subset V \times V$$
   (Иными словами, множество рёбер определяется как подмножество прямого произведения множества вершин графа с собой)
   - Определим граф с кратными рёбрами и петлями(обычно в литературе их называют "мультиграфами"и "псевдографами"соответственно):
   Пусть E - некоторый набор отрезков, δE - множество концов отрезков, $\mathfrak{R}$ - отношение эквивалентности на δE, тогда вершины являются некоторыми классами эквивалентности.
2. **Матрица смежности** - матрица размерности Р$\times$Р, где на позиции с индексом i, j (номер строки, номер столбца соответственно) стоит 1, если $i-$ая и $j-$ая вершины пересекаются (т.е. существует ребро, связывающее их), иначе стоит 0.
3. **Матрица инциденции** - матрица размерности В$\times$Р(пусть это матрица $H^{\text{В}\times\text{Р}}$), где строки соответствуют вершинам графа, столбцы - рёбрам графа. Соответственно, в ячейке h_{ij} будет указано отношение инциденции для i-ой вершины и j-ого ребра графа.

   Напомним, что ребро и вершина находятся в отношении **инциденции**, если *вершина лежит на заданном ребре.*

   Если граф ненаправленный, имеем:
   $$h_{ij} = \begin{cases} 1, & \text{если вершина } v_i \text{ инцидентна ребру } e_j \\ 0, & \text{в противном случае} \end{cases}$$

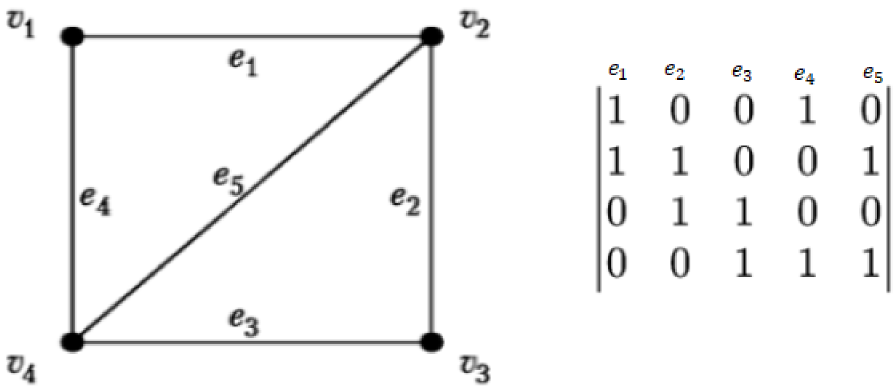


Рис. 1: Пример ненаправленного графа с матрицей инциденции

Если направленный:

$$h_{ij} = \begin{cases} 1, \text{ если вершина } v_i \text{ инцидентна ребру } e_j \text{ и является его концом} \\ 0, \text{ вершина } v_i \text{ не инцидентна ребру } e_j \\ -1, \text{ если вершина } v_i \text{ инцидентна ребру } e_j \text{ и является его началом} \end{cases}$$

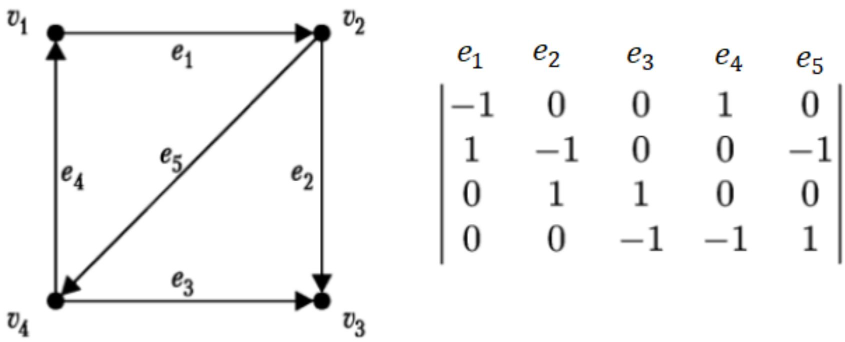


Рис. 2: Пример направленного графа с матрицей инциденции

4. **Путь** $\overrightarrow{l}$ в графе - набор рёбер $\overrightarrow{l} = (\overrightarrow{e}_1, \overrightarrow{e}_2, \cdots, \overrightarrow{e}_n)$ таких, что $t(\overrightarrow{e}_i) = S(\overrightarrow{e}_{i+1})\ \forall i \in \{1, 2, \cdots, n-1\}$.

5. **Простой путь** - путь, в котором каждое из рёбер графа пройдено не более одного раза.

6. **Ориентированное ребро** - ребро, имеющее направление.

7. **Число рёберной связности** λ - минимальное число рёбер, при удалении которых граф перестанет быть связным.

8. **Число вершинной связности** $\ae$ - наименьшее число вершин, при удалении которых граф потеряет связность или станет одновершинным.

9. **Валентность (степень) вершины** $\nu(a)$ - определяет, сколько концов рёбер входит в вершину a.

10. **Паспорт графа** - матрица $1 \times$ В, где на i-ой позиции задаётся валентность i-ой вершины.

11. **Изоморфизм графов** - Пусть Γ и Γ' - два графа, а отображение $\phi : V(\Gamma) \to V(\Gamma')$ таково, что $xy \in E(\Gamma) \Leftrightarrow \phi(x)\phi(y) \in E(\Gamma')$. Тогда ϕ - изоморфизм графов Γ и Γ', а сами графы изоморфны.

    Пояснение: x, y - в данном случае некоторые вершины графа Γ, соответственно xy - некоторое ребро, связывающее эти две вершины. Отображение ϕ переводит вершины x, y графа Γ в некоторые вершины графа Γ', для простоты назовём их x', y'. Но тогда $x' = \phi(x), y' = \phi(y)$; теперь очевидно, что $\phi(x)\phi(y)$ - это некоторое ребро, связывающее вершины x' и y', и если это ребро является ребром графа (Γ'), то отображение ϕ будет изоморфизмом.

12. **Дерево** - связный граф, не имеющий циклов.

    Пояснение 1: **связным** называется граф такой, что из любой вершины графа существует путь в любую другую.

    Пояснение 2: **циклом** называется путь $\vec{l} = (\vec{e_1}, \vec{e_2}, \cdots, \vec{e_n}) : S(\vec{e_1}) = t(\vec{e_n})$ (то есть вершина отправления является вершиной окончания пути).

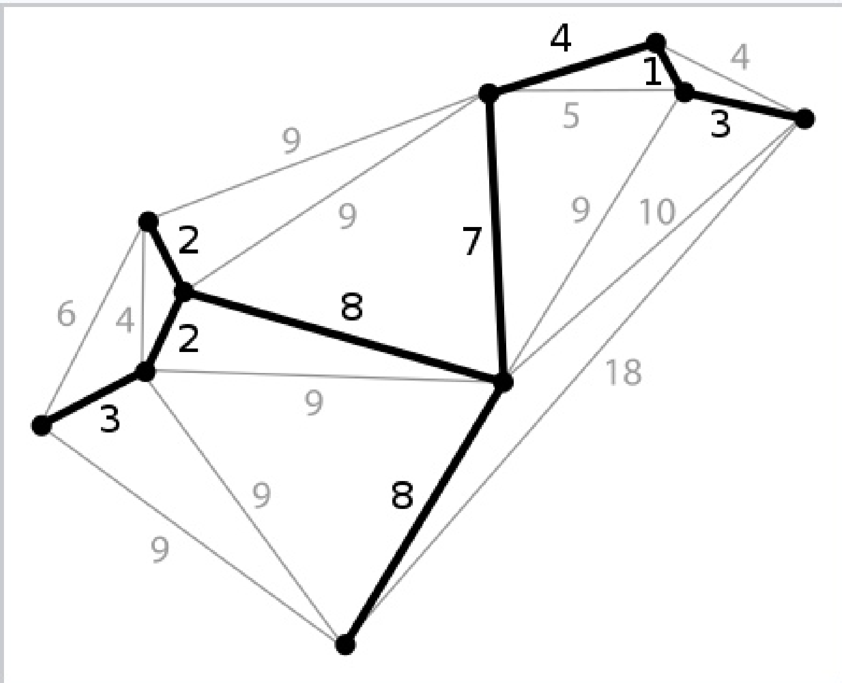
    Пояснение 3: Это не строгое определение дерева, $\exists$ 4 строгих эквивалентных определения. Все они будут приведены в следующем разделе.

13. **Мост, компонента связности:**

    Пояснение: понятие моста вводилось в семестре в довольно наивной формулировке, поэтому здесь будет приведена более строгая, хотя и с использованием терминов, не вводившихся лектором.

    - **Компонентой связности** называется связный подграф. Говоря строже, это такой набор вершин графа, между любой парой которых можно проложить путь.
    - **Мостом** называется ребро, удаление которого увеличивает количество компонент связности.

14. **Остовное дерево** Δ - дерево, содержащее все вершины некоторого графа.



Пример минимального остовного дерева в графе. Числа на ребрах обозначают вес ребер.

Рис. 3:

15. **Полный граф** - граф, в котором любые две вершины являются смежными.

16. **Двудольный граф** - граф, множество вершин которого может быть разбито на два непересекающихся множества $(V_1, V_2 : V_1 \cup V_2 = V, V_1 \cap V_2 = \emptyset)$, причём всякое ребро из E инцидентно вершине из V_1 и вершине из V_2.

17. **Полный двудольный граф** - двудольный граф, содержащий все рёбра, соединяющие множества V_1 и V_2.

    Пусть $\text{В}_1 = m, \text{В}_2 = n$, тогда полный двудольный граф обозначается $K_{m,n}$.

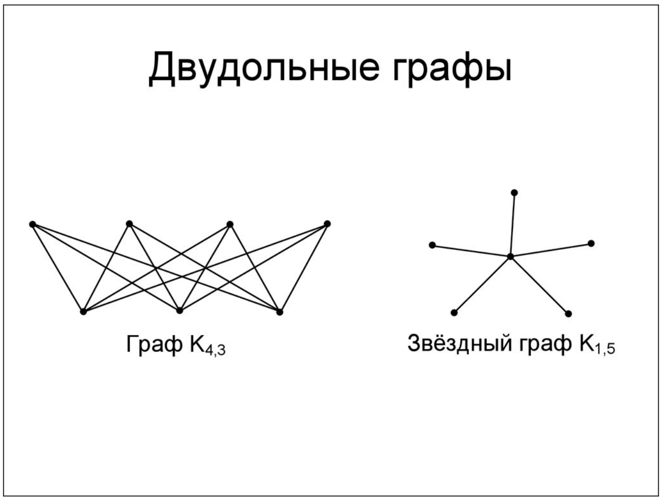


Рис. 4: Примеры двудольных графов (полных)

18. **Планарный граф** - граф, который можно изобразить на плоскости так, чтобы его рёбра не пересекались во внутренних точках.

    Иными словами, это такой граф, который можно реализовать как симплициальный комплекс с нулевыми вторыми гомологиями из симплексов размерности два.

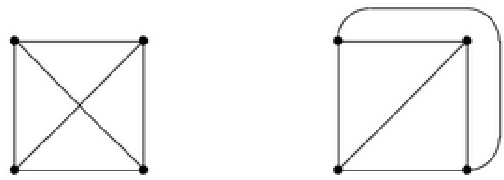


Рис. 5: Планарный краф K_4 и его укладка на плоскость

19. **Грани графа** - части, на которые *плоский граф* (т.е. граф в изображении на плоскость, неважно, пересекаются его рёбра во внутренних точках или нет) делит плоскость.

20. **Правильно вложенный в плоскость граф** - граф, грани которого на плоскости есть многоугольники.

21. **Цикломатическое число графа** $g(\Gamma) = \text{Р} - \text{В} + 1$.

    Замечание: Единица в данном равенстве есть количество компонент связности в данном графе. В данном курсе это понятие не рассматривается, но осознавать его стоит. Выше дано его определение.

22. **Гомеоморфность графа** - граф называется гомеоморфным, если он получен путём стягивания рёбер или долбавлением вершин на существующие рёбра.

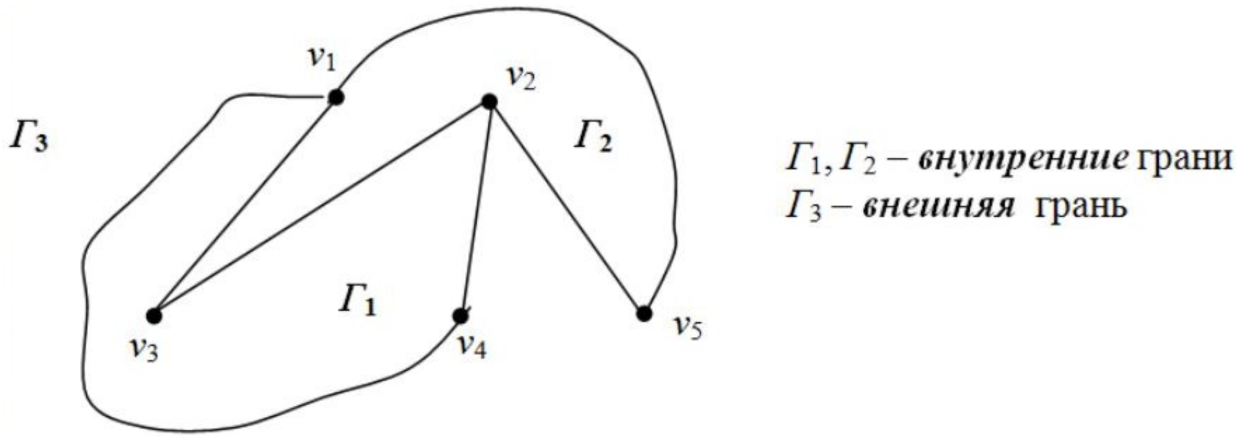


Рис. 6: Иллюстрация к понятию граней графа

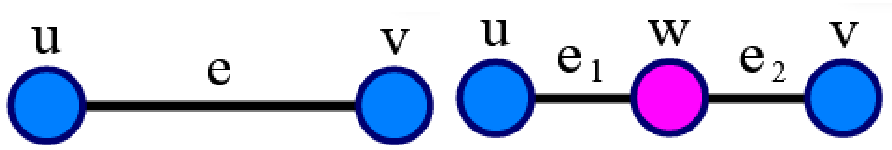


Рис. 7: Пример гомеоморфных графов

3 Теоремы и алгоритмы:

1. **Алгоритмы нахождения кратчайшего пути в графе** - Алгоритм Дейкстры(для взвешенного графа) и алгоритм поиска в ширину(для не взвешенного графа):

   Пояснение 1: взвешенным графом называется граф, рёбра которого имеют определённую цену (цена - она же вес ребра).

   (a) **Алгоритм Дейкстры:**

   - Словесное описание:

     i. Найти узел с наименьшей стоимостью (то есть узел, за который можно добраться за минимальную цену).

        Замечание 1: перед тем, как обходить граф, в котором ещё не известна стоимость ни одной вершины, необходимо узел отправления пометить ценой ноль, остальные получат стоимость ∞.

     ii. Обновить стоимость соседей этого узла.

        Пример: пусть V_1 - вершина со стоимостью 4. V_3 - вершина со стоимостью 15 (возможно, эта цена была получена, когда в эту вершину мы попали по другому пути, а, возможно, такая цена была установлена изначально). Эти две вершины связаны друг с другом ребром стоимостью 6. Совершенно логично, что $4 + 6 < 15, \Rightarrow$ новая цена V_3 будет $4 + 6 = 10$.

     iii. Перейти снова к пункту i и повторять до тех пор, пока все узлы графа не будут оценены.

     iv. Вычислить итоговый путь.

   - Псевдокод:

```
'''
graph — ассоциативный массив, где вершина — ключ, которому
 соответствует список смежных вершин и стоимости рёбер.

costs — ассоциативный массив, где ключ — вершина, значение — её стоимость.

parents — ассоциативный массив, где ключ — вершина,
```

```
a значение — вершина, из которой в неё пришли.

neighbors — ассоциативный массив с элементами,соседними
вершине node; ключ — смежная вершина, значение —
стоимость соединяющего ребра.
'''


def Dijkstra():
    #Найти узел с наименьшей стоимостью среди необработанных
    node = find_lower_cost_node(costs)
    while node is not None:#Если обработаны все узлы, цикл завершается
        cost = costs[node]#Получить цену текущей вершины
        #Получить коллекцию соседей вершины в формате ключ" — значение"
        neighbors = graph[node]
        for n in neighbors.keys():#Перебрать всех соседей текущего узла
            new_cost = cost + neighbors[n]
            #Если к соседу можно быстрее добраться через текущий узел...
            if costs[n] > new_cost:
                costs[n] = new_cost
                parents[n] = node
            #обновить стоимость этого узла, текущий узел становится его родителем
        processed.append(node)#узел помечается как обработанный
        node = find_lower_cost_node(costs)

def find_lower_cost_node(costs):
    lowest_cost = float("inf")
    lowest_cost_node = None
    for node in costs:#Перебрать все узлы
        cost = costs[node]
        #Если этот узел с наименьшей стоимость из уже виденных
        #И он ещё не обработан...
        if cost < lowest_cost and node not in processed:
            lowest_cost = cost
            lowest_cost_node = node
        #Он назначается новым узлом с наименьшей стоимостью
    return lowest_cost_node
```

(b) **Алгоритм поиска в ширину** (словесное описание):

   i. Расставляем метки вершинам графа: V_1 – метка 1, V_2 - вершина, смежная с предыдущей, получает метку 2, $\cdots$

   ii. На k-ом шаге метку k+1 получат вершины, которые либо ещё не имеют метки, либо являются смежными с вершиной k.

   iii. Если на некотором шаге не нашлось вершин из предыдущего пункта, и последняя вершина, до которой удалось дойти, не является пунктом назначения, то граф несвязный. Иначе последняя вершина - $t(e_n)$.

2. **Жадный алгоритм:**

   Пояснение 2: используется для нахождения минимального и максимального остовного дерева.

   Пояснение 3: **минимальным остовным деревом** будем называть дерево, сумма рёбер которого минимальна среди всех остовных деревьев данного графа.

   (a) Найти самое дешёвое (самое дорогое) ребро в графе. Если таких несколько - берём любое.

(b) Ищем самоё дешёвое (самое дорогое) реброе среди рёбер, смежным с ребром из предыдущего пункта. Берём его.

(c) Проверяем, не вошли ли мы в цикл. Если вошли - выбираем другое смежное ребро.

(d) Повторяем алгоритм до тех пор, пока не обойдём все вершины.

3. **Теорема 1 (Лемма о рукопожатиях)** *Данный набор натуральных чисел является паспортом **какого-нибудь графа** $\Leftrightarrow$ сумма всех валентностей - всех чисел данного набора - является чётной.*

   Следствие 1 *Количество вершин с нечётным количеством валентностей всегда чётно.*

4. **Теорема 2 (Необходимое и достаточное условие для связности графа)** *Граф с данным паспортом является связным*

$$\Leftrightarrow \sum_{i=1}^{n} (\nu_i - 2) + 2 \geq 0$$

   Где ν_i - валентность i-ой вершины.

   Замечание 1 $g(\Gamma) = 0 \Leftrightarrow$ *граф является деревом.*

5. **Цикломатическое число** вычисляется по следующим эквивалентным формулам:

$$g(\Gamma) = \sum_{i=1}^{n} (\nu_i - 2) + 2 = \text{Р} - \text{В} + 1 \geq 0$$

6. **Теорема 3 (Об эквивалентности определений дерева)** *Следущие 4 определения дерева эквивалентны:*

   (a) $\text{В} = \text{Р} + 1$

   (b) При удалении $\forall$ ребра граф теряет связность ($\Leftrightarrow \forall$ ребро является мостом)

   (c) $\forall a, b \in V\ \exists!\ \vec{l}$, причём $\vec{l}$ - простой путь.

   (d) В графе нет простых замкнутых путей. Пояснение: замкнутым называется путь, точка отправления которого является и точкой прибытия.

7. **Теорема 4 (Связь чисел вершнинной связности æ и рёберной связности λ)**

$$\lambda \geq \text{æ}$$

8. **Теорема 5 (Критерий эйлеровости графа)** *Граф является эйлеровым, если либо все степени вершин чётны, либо количество вершин с нечётной степенью ≤ 2.*

   Замечание 2 *Эйлеров граф - граф, который можно обойти, пройдя каждое ребро не более одного раза.*

9. **Теорема 6 (Критерий двудольности графа)** *Все циклы в графе имеют чётную длину.*

10. **Теорема 7 (Теорема Эйлера о планарных графах)** *Для $\forall$ планарного графа выполняется:*

$$\text{В} - \text{Р} + \Gamma = 2$$

   Где Γ - количество граней графа.

11. **Теорема 8 (О непланарности V_5, $V_{3,3}$)** *Графы V_5, $V_{3,3}$ непланарны.*

   Замечание 3 *V_5 - полный граф на 5 вершинах. $V_{3,3}$ - полный двудольный граф: ситуация 'Три соседа - три колодца'.*

12. **Теорема 9 (Теорема Понтрягина-Куратовского)** *Граф не планарен $\Leftrightarrow$ в нем $\exists$ подграф, гомеоморфнаый либо V_5, либо $V_{3,3}$.*

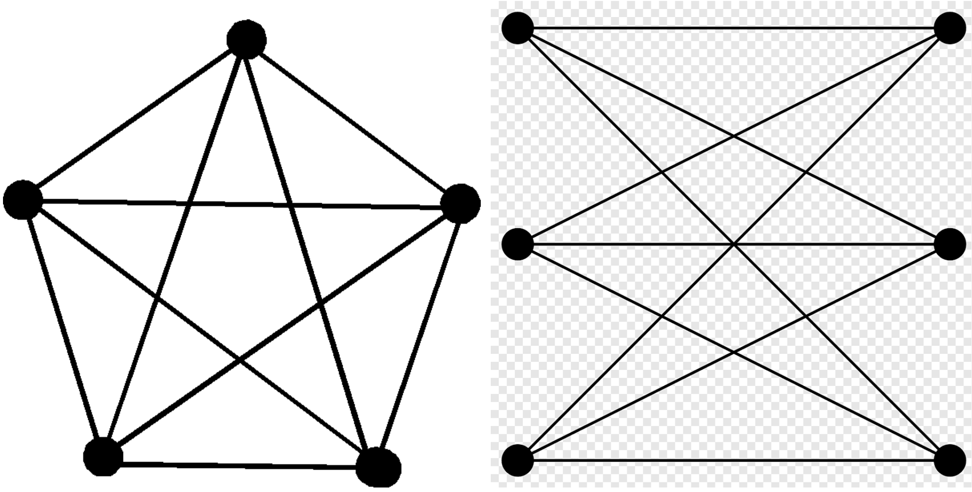


Рис. 8: $V_5, V_{3,3}$

Список рекомеднуемой литератруры:

- Ф. А. Новиков - 'Дискретная математика для программистов'
- Д.В. Карпов - 'Теория графов'
- Адитья Бхаргава - 'Грокаем алгоритмы'