Тема 3

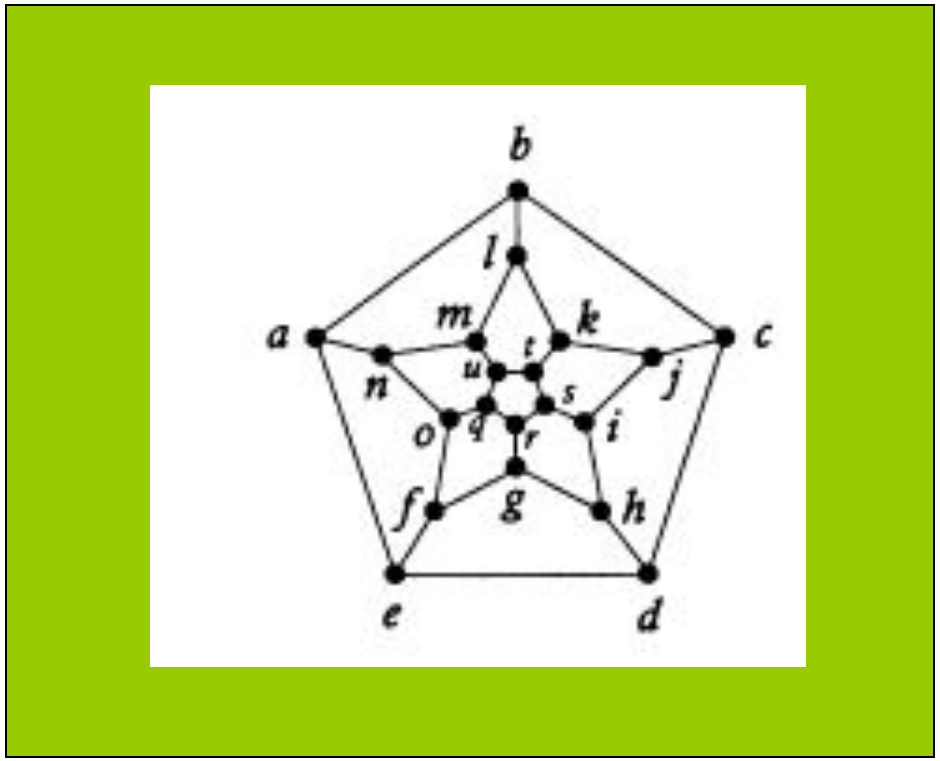
Гамильтоновы графы.
Пути и циклы Гамильтона.

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

Уильям Роуэн Гамильтон (William Rowan Hamilton, 1857).

Вершина – город. Игра – найти путь через все города, посетив каждый город один раз, и вернуться в исходный пункт.

Додекаэдр (12 граней, 20 углов) – пример для прохождения по всем вершинам-городам.



Цель игры – найти цикл графа, проходящего через каждую вершину.

Определение.

Любой цикл графа, обладающий таким свойством, называется ***гамильтоновым циклом***.

Гамильтонов цикл противоположен эйлерову циклу (проходит ребра один раз).

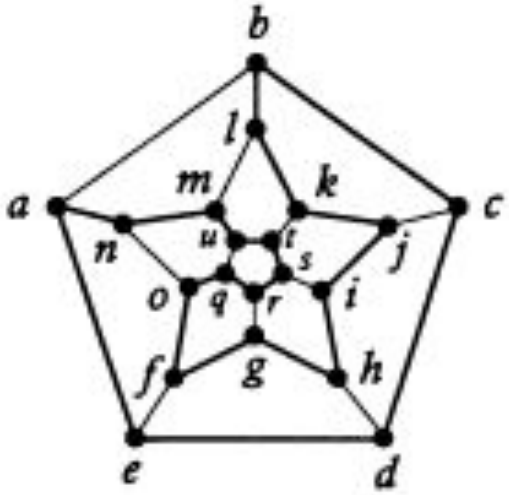
Формальное описание пути Гамильтона и цикла Гамильтона.

Определение.

Пусть *G* – граф. ***Гамильтонов путь*** – это простой путь, который проходит через каждую вершину графа *G*.

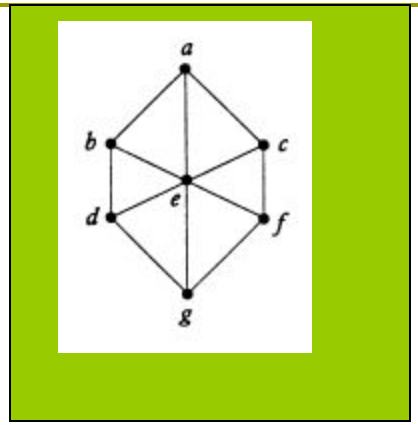
Гамильтонов цикл – это простой цикл, который проходит через каждую вершину графа *G (путь с возвращением)*.

Граф для игры Гамильтона имеет гамильтонов цикл.

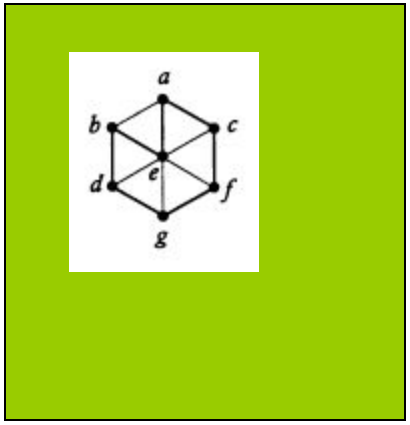


Пример.

Граф



имеет гамильтонов цикл

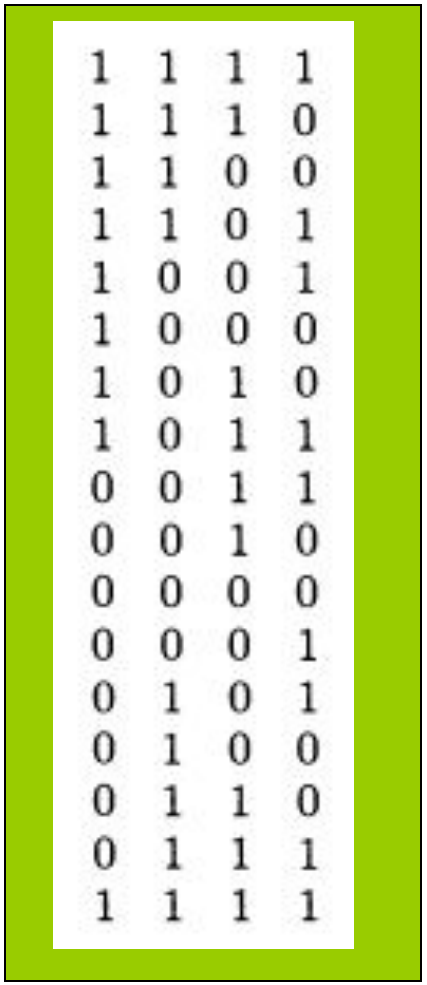


Пример.

Гиперкуб порядка n > 3 имеет гамильтонов цикл.

Его код Грея:

Гамильтонов цикл для гиперкуба порядка 4



Пример.

Полный граф K_n при $n \geq 3$ имеет гамильтонов цикл.

Пусть $v_1, v_2, v_3, \ldots, v_n$ - вершины графа K_n.

Между любыми вершинами имеется ребро, то всегда существует ребро из v_i в v_{i+1} и существует ребро из последней вершины v_n обратно в v_1.

Теорема.

Для любой вершины из цикла Гамильтона существует ровно два ребра из этого цикла, инцидентные данной вершине.

Доказательство:

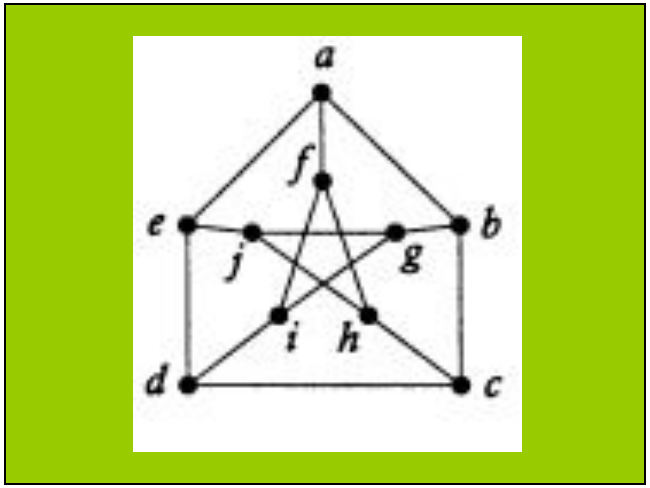
По ходу цикла для каждой вершины V имеется ребро к циклу и ребро из цикла. Если бы существовало еще одно ребро цикла, инцидентное вершине V, то цикл вернулся бы в вершину V, и она опять появилась бы в цикле, что противоречит определению гамильтонова цикла. Существует ровно два ребра, которые инцидентны вершине V из цикла Гамильтона.

Следствие.

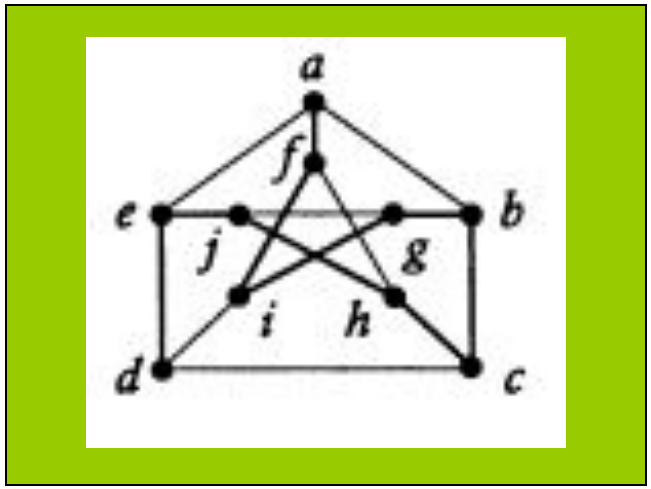
Любой граф, содержащий вершину степени 1, не может быть гамильтоновым.

Пример.

Граф



имеет гамильтонов путь, но не имеет гамильтонова цикла



Чтобы граф имел гамильтонов цикл, степень каждой вершины должна быть не меньше 2-х. Граф должен быть связным.

Теорема.

Если граф *G* имеет разрезающее ребро, то он не может иметь гамильтонов цикл. Если компоненты графа, полученные путем удаления разрезающего ребра, имеют гамильтонов цикл, то граф *G* имеет гамильтонов путь.

Теорема.

Если $G(V, E)$ - связный граф с n вершинами, где $n \geq 3$, и для каждой пары различных несмежных вершин $u, v \in V$,

$$\deg(u) + \deg(v) \geq n,$$

тогда граф G имеет гамильтонов цикл.

Обозначение.

$\deg(v)$ – степень вершины v.

Следствие.

Если $G(V, E)$ - связный граф с n вершинами, где $n \geq 3$, и если для каждой вершины $v \in V$ выполняется $\deg(v) \geq n/2$, то граф G имеет гамильтонов цикл.

Теорема.

Пусть $G(V, E)$ - связный граф с $n \geq 3$ вершинами и пусть u и v - несмежные вершины графа G такие, что

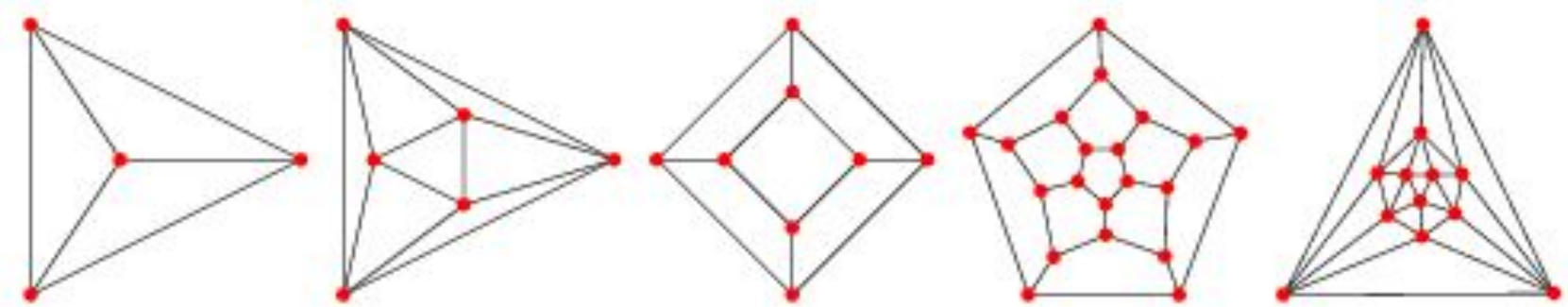
$$\deg(u) + \deg(v) \geq n,$$

Отсюда граф G^e, состоящий из графа G с присоединенным ребром $e = \{u, v\}$, имеет гамильтонов цикл тогда и только тогда, когда граф G имеет гамильтонов цикл.

Пример.

- *Платоновы графы* – графы, образованные вершинами и рёбрами платоновых тел – правильных многогранников.

Какие из платоновых графов являются 1) эйлеровыми, 2) гамильтоновыми? Указать в каждом случае эйлеров или гамильтонов цикл.



Определение.

Пусть имеется граф G с n вершинами. Добавим ребра к несмежным вершинам u и v графа G, для которых

$$\deg(u) + \deg(v) \geq n$$

до тех пор, пока это возможно. По завершении процесса полученный граф называют ***замыканием графа*** G.

Определение.

Пусть G – граф с n вершинами. **Замыканием графа** G, обозначаемым $\mathrm{cl}(G)$, называется граф, полученный из графа G рекурсивным добавлением ребер к несмежным вершинам u и v, для которых

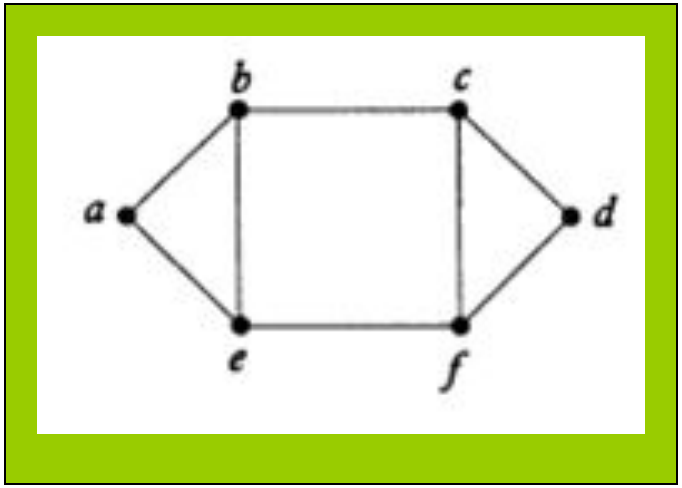
$$\deg(u) + \deg(v) \geq n$$

до тех пор, пока это возможно.

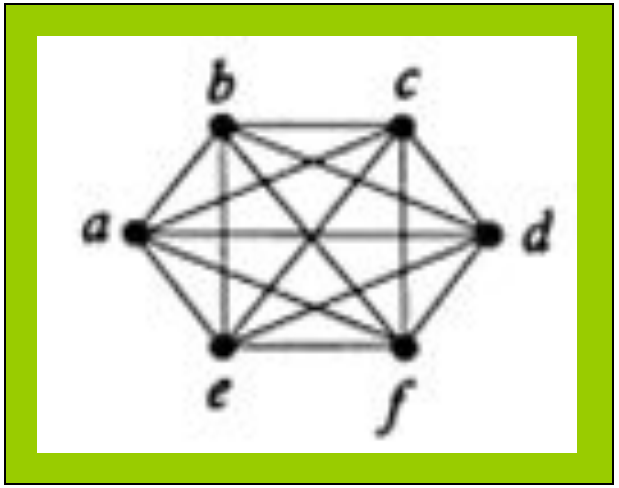
Для произвольного графа
G cl(cl(G)) = cl(G)

Пример.

Если G – граф

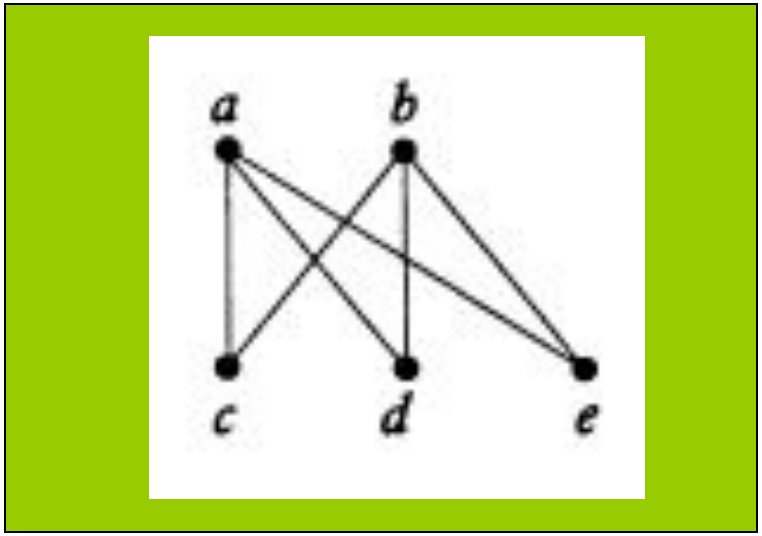


то cl(G) - граф
(замыкание графа G)

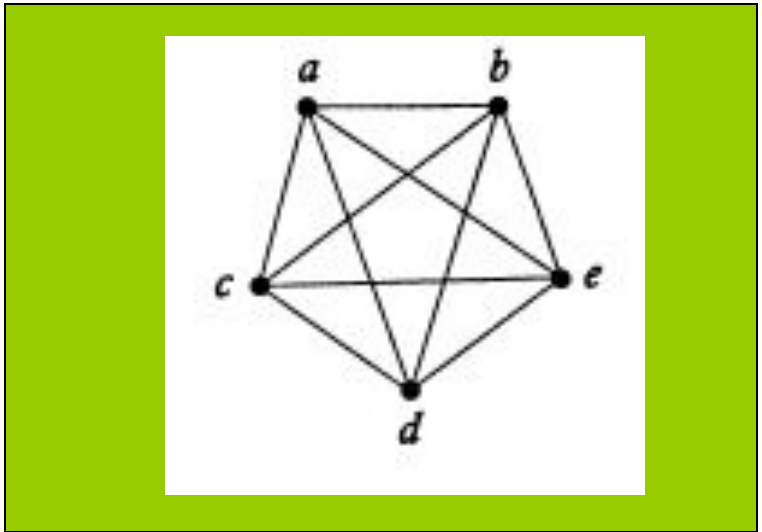


Пример.

Если G – граф



то $cl(G)$ - граф



Пример.

Если G - полный двудольный граф $K_{m,m}$ при $m \geq 1$, то $cl(G)$ – полный граф K_{m+m} .

***Теорема*.**

Граф G имеет гамильтонов цикл тогда и только тогда, когда граф $cl(G)$ имеет гамильтонов цикл.

Следствие.

При $m \geq 1$ полный двудольный граф $K_{m,m}$ имеет гамильтонов цикл.

Взвешенные графы
и алгоритмы поиска кратчайшего пути

Определение.

Пусть $A = (a_1, a_2, a_3, \dots, a_n)$ и $B = (b_1, b_2, b_3, \dots, b_n)$ – матрицы-строки, каждое из a_i и b_i - неотрицательные целые числа или ∞. Тогда

$$A \wedge B = (\min(a_1, b_1), \min(a_2, b_2), \min(a_3, b_3), \dots, \min(a_n, b_n))$$

Определение.

Пусть c – число, $A = (a_1, a_2, a_3, \dots, a_n)$ - матрица-строка. Тогда

$$c + A = c + (a_1, a_2, a_3, \dots, a_n) = (c + a_1, c + a_2, c + a_3, \dots, c + a_n).$$

Теорема.

Пусть $a = v_1$ и $b = v_n$.

Если $v_1, v_2, \ldots, v_i, v_{i+1}, \ldots, v_j, v_{j+1}, \ldots, v_n$ есть кратчайший путь между a и b, то $v_i, v_{i+1}, \ldots, v_j$, часть этого пути между вершинами v_i и v_j, является кратчайшим путем между v_i и v_j .

Отыскивается кратчайшее расстояние между v_1 и vn

Алгоритм Дейкстры(1)

Для данного взвешенного графа алгоритм дает кратчайшее расстояние от вершины v_1 к вершине v_n. Каждой вершине поставим в соответствие упорядоченную пару $(\infty, 0)$. Первая координата вершины $v_i(m, v_r)$ будет означать присвоенное расстояние от вершины v_1 к вершине v_i, а вторая координата — предыдущую вершину пути от v_1 к v_i.

(1) Начать в вершине $v_1(\infty, 0)$, заменить ее на $v_1(0, 0)$ и сделать постоянной. Остальные вершины на этот момент оставить временными.

(2) Когда вершина $v_k(m, v_r)$ станет постоянной, для каждой вершины v_j, смежной к v_k, прибавить величину m к расстоянию от вершины v_k к вершине v_j. Если это значение меньше, чем текущее расстояние, присвоенное вершине v_j, заменить текущее расстояние этой суммой и заменить вторую координату на v_k.

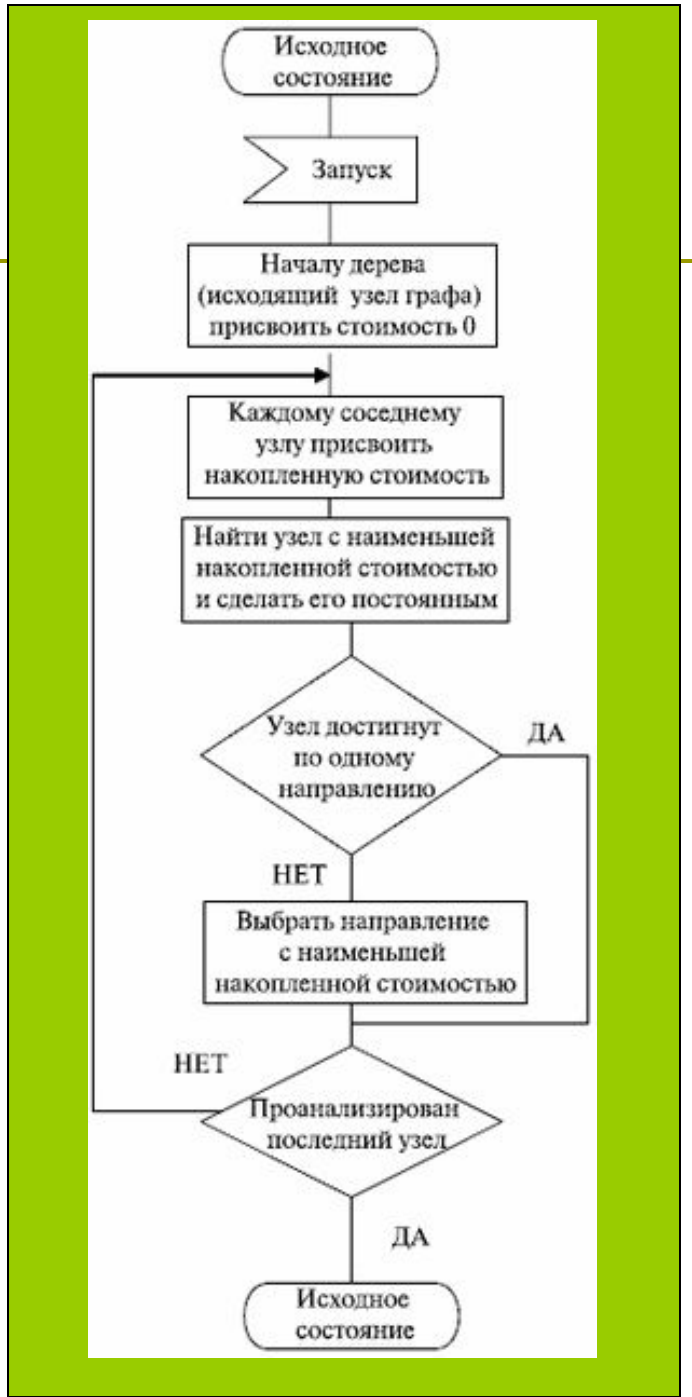
(3) Найти минимум из расстояний, приписанных временным вершинам. Первую из вершин с таким расстоянием делаем постоянной.

(4) Если v_n — не постоянная вершина, то возвращаемся к пункту (2).

(5) Если v_n — постоянная вершина, то расстояние, присвоенное вершине v_n, является кратчайшим расстоянием от v_1 к v_n.

(6) Для нахождения пути начать в вершине v_n, найти предшествующую ей вершину пути (вторая координата). Для каждой вершины пути v_j находить предшествующую ей вершину пути, пока не будет достигнута вершина v_1. Перестановка вершин в обратном порядке даст кратчайший путь.

Алгоритм Дейкстры (Dijkstra's algorithm) — алгоритм на графах, изобретённый нидерландским ученым Э. Дейкстрой в 1959 году. Находит кратчайшее расстояние от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса.



Примеры

- *Пример 1.* Дана сеть автомобильных дорог, соединяющих города Московской области. Некоторые дороги односторонние. Найти кратчайшие пути от города Москва до каждого города области (если двигаться можно только по дорогам).
- *Пример 2.* Имеется некоторое количество авиарейсов между городами мира, для каждого известна стоимость. Стоимость перелёта из A в B может быть не равна стоимости перелёта из B в A. Найти маршрут минимальной стоимости (возможно, с пересадками) от Челябинска до Ставангера (Норвегия).

Пример

- Каждой вершине из *V* сопоставим метку — минимальное известное расстояние от этой вершины до *a*. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.
- **Инициализация**. Метка самой вершины *a* полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от *a* до других вершин пока неизвестны. Все вершины графа помечаются как непосещённые.

Пример (продолжение)

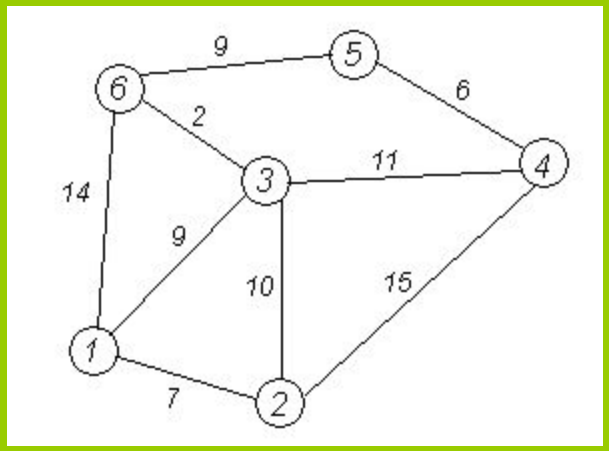
- **Шаг алгоритма**. Если все вершины посещены, алгоритм завершается. В противном случае, из ещё не посещённых вершин выбирается вершина *u*, имеющая минимальную метку. Мы рассматриваем всевозможные маршруты, в которых *u* является предпоследним пунктом.

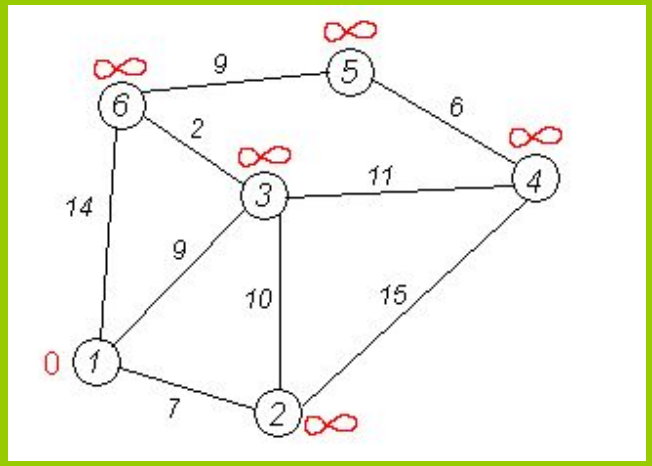
  Вершины, в которые ведут рёбра из *u*, назовем *соседями* этой вершины. Для каждого соседа вершины *u*, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки *u* и длины ребра, соединяющего *u* с этим соседом. Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину *u* как посещенную и повторим шаг алгоритма.

Пример

Пусть требуется найти кратчайшие расстояния от 1-й вершины до всех остальных.

Кружками обозначены вершины, линиями — пути между ними (ребра графа). В кружках обозначены номера вершин, над ребрами обозначена их «цена» — длина пути. Рядом с каждой вершиной красным обозначена метка — длина кратчайшего пути в эту вершину из вершины 1.

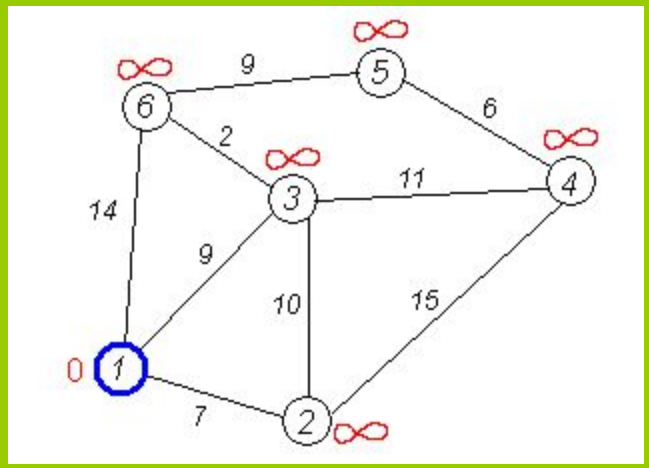


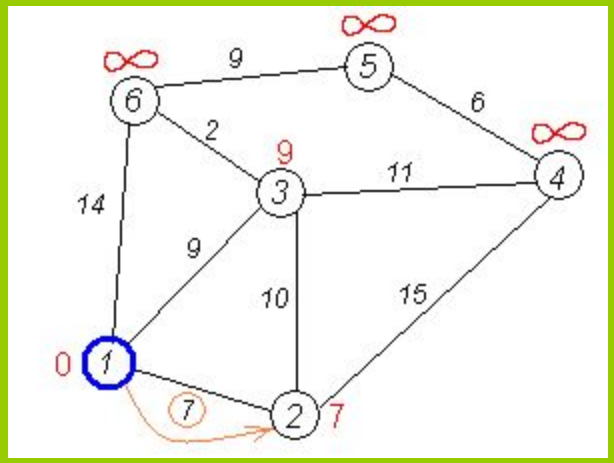


Пример (продолжение)

Минимальную метку имеет вершина 1. Её соседями являются вершины 2, 3 и 6.

Первый по очереди сосед вершины 1 — вершина 2, потому что длина пути до неё минимальна. Длина пути в неё через вершину 1 равна сумме кратчайшего расстояния до вершины 1, значению её метки, и длины ребра, идущего из 1-й в 2-ю, то есть 0 + 7 = 7. Это меньше текущей метки вершины 2, бесконечности, поэтому новая метка 2-й вершины равна 7.

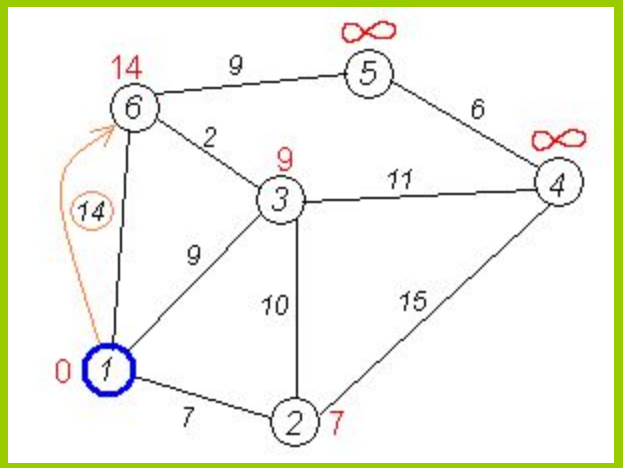


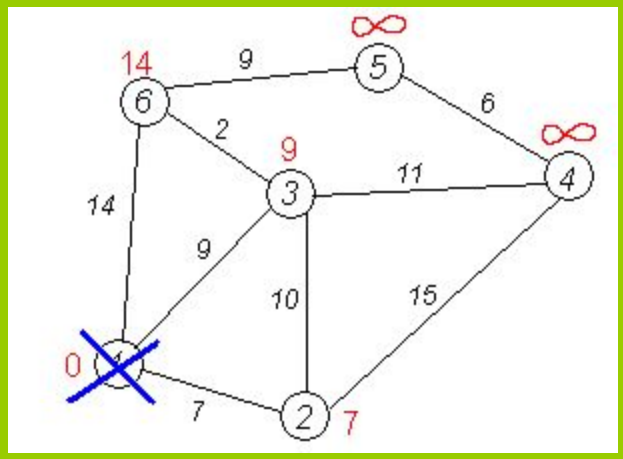


Пример (продолжение)

Аналогичную операцию проделываем с двумя другими соседями 1-й вершины — 3-й и 6-й.

Все соседи вершины 1 проверены. Текущее минимальное расстояние до вершины 1 считается окончательным и пересмотру не подлежит (то, что это действительно так, впервые доказал Э. Дейкстра). Вычеркнем её из графа, чтобы отметить, что эта вершина посещена.

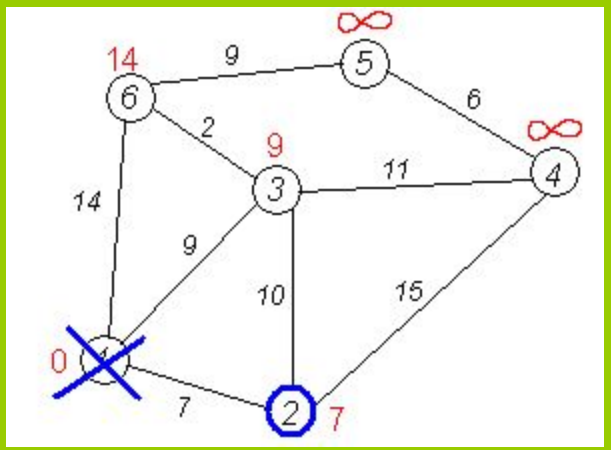


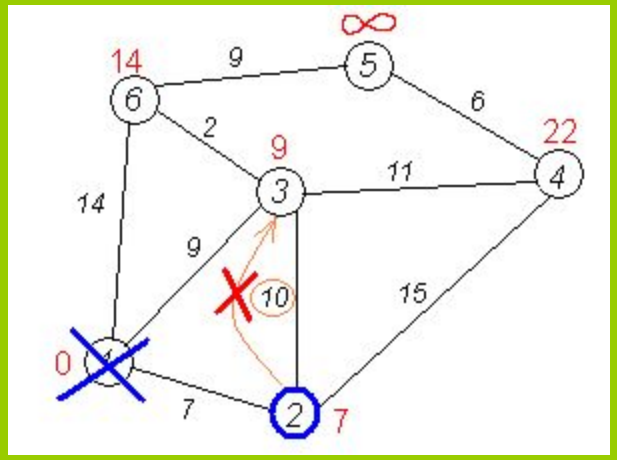


Пример (продолжение)

Второй шаг. Шаг алгоритма повторяется. Снова находим «ближайшую» из непосещенных вершин. Это вершина 2 с меткой 7.

- Снова пытаемся уменьшить метки соседей выбранной вершины, пытаясь пройти в них через 2-ю вершину. Соседями вершины 2 являются вершины 1, 3 и 4.
- Первый (по порядку) сосед вершины 2 — вершина 1. Но она уже посещена, поэтому с 1-й вершиной ничего не делаем.
- Следующий сосед вершины 2 — вершина 3, так как имеет минимальную метку из вершин, отмеченных как не посещённые. Если идти в неё через 2, то длина такого пути будет равна 17 (7 + 10 = 17). Но текущая метка третьей вершины равна 9, а это меньше 17, поэтому метка не меняется.

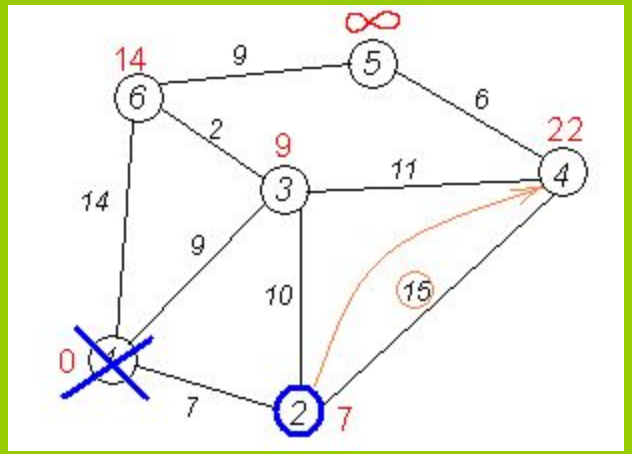


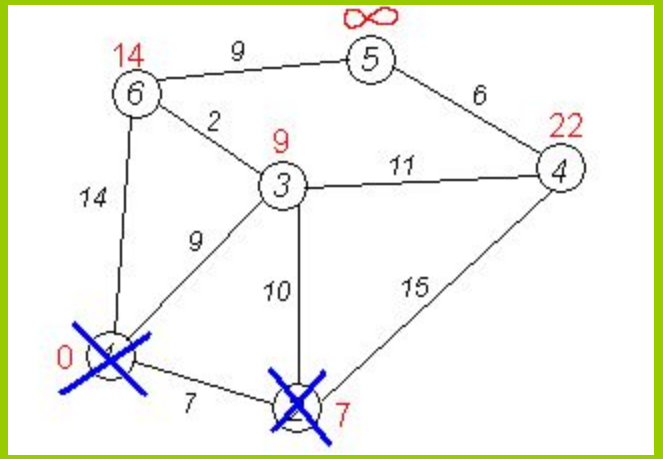


Пример (продолжение)

Ещё один сосед вершины 2 — вершина 4. Если идти в неё через 2-ю, то длина такого пути будет равна сумме кратчайшего расстояния до 2-й вершины и расстояния между вершинами 2 и 4, то есть 22 (7 + 15 = 22). Поскольку 22<, устанавливаем метку вершины 4 равной 22.

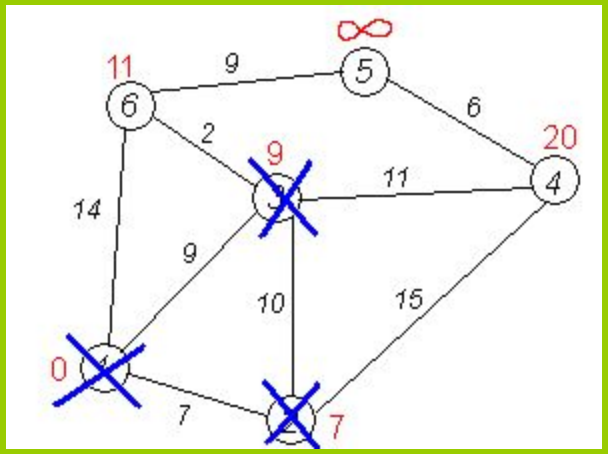
Все соседи вершины 2 просмотрены, замораживаем расстояние до неё и помечаем её как посещенную.





Пример (продолжение)

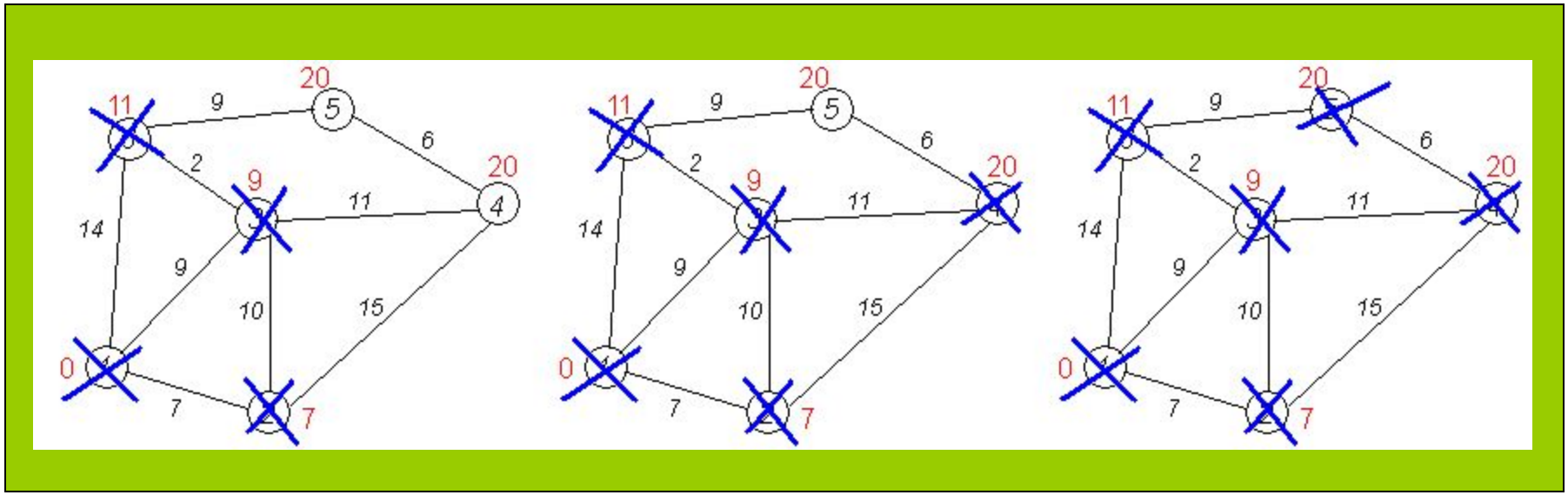
Третий шаг. Повторяем шаг алгоритма, выбрав вершину 3. После её «обработки» получим такие результаты:



Пример (продолжение)

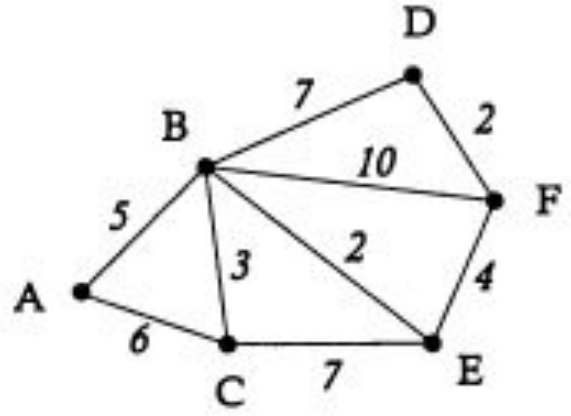
Дальнейшие шаги. Повторяем шаг алгоритма для оставшихся вершин. Это будут вершины 6, 4 и 5, соответственно порядку.

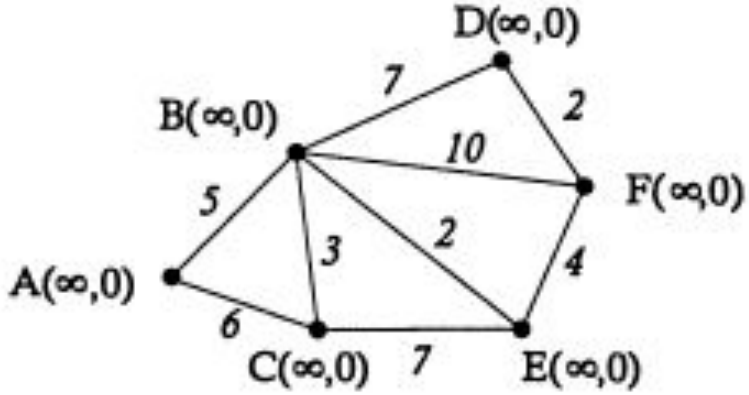
Завершение выполнения алгоритма. Алгоритм заканчивает работу, когда нельзя больше обработать ни одной вершины. В данном примере все вершины зачеркнуты, однако ошибочно полагать, что так будет в любом примере - некоторые вершины могут остаться не зачеркнутыми, если до них нельзя добраться. Результат работы алгоритма виден на последнем рисунке: кратчайший путь от вершины 1 до 2-й составляет 7, до 3-й — 9, до 4-й — 20, до 5-й — 20, до 6-й — 11.



Пример.

Взвешенный граф, в котором отыскивается кратчайшее расстояние от вершины A к вершине F. Каждой вершине поставим в соответствие упорядоченную пару (∞, 0)





В алгоритме для кратчайшего расстояния между двумя вершинами использованы матрицы

Алгоритм Дейкстры(2)

(1) Установить $P(1) = 0$, $T(1) = \infty$ и $W_{j1} = \infty$ при $1 \leq j \leq 5$. Положить $V = 1$.

(2) Добавить $P(V)$ к каждому элементу $W(V)$. Более точно, пусть $Sum = P(V) + W(V)$. Для $Sum(j) < T(j)$ положить $pred(j) = V$. Далее, положить $T = T \wedge Sum$, и для наименьшего i такого, что $T(i) = \min\{T(j) : 1 \leq j \leq n\}$ положить $P(i) = T(i)$, $T(i) = \infty$ и $W_{ji} = \infty$ для $1 \leq j \leq n$. Положить $V = i$.

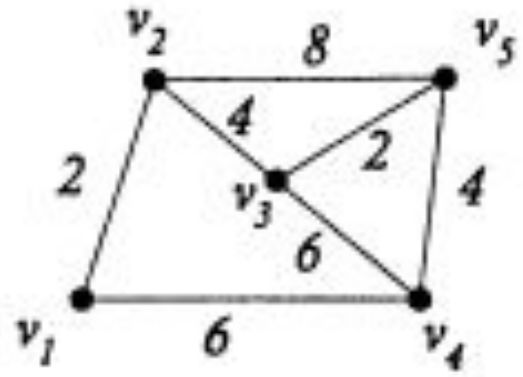
(3) Если $i \neq n$, вернуться к шагу 2.

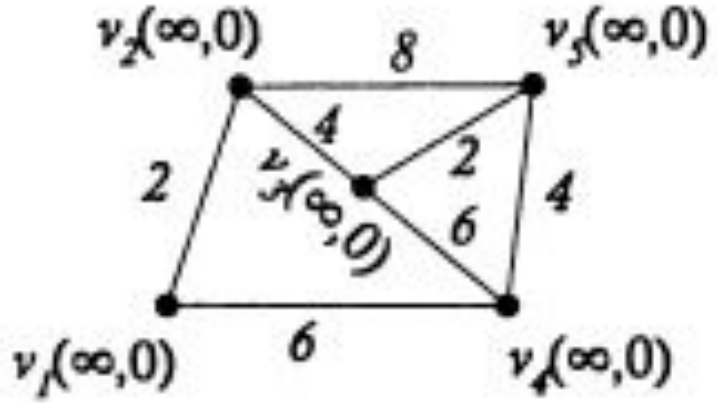
(4) Если $i = n$, то $P(i)$ — кратчайшее расстояние от v_1 к v_n.

(5) Для нахождения пути использовать $pred(n)$ и найти вершину, которая предшествует n. Для каждой вершины пути v_j использовать $pred(j)$ и находить предшествующую ей вершину, пока не будет достигнута вершина v_1. Перестановка вершин в обратном порядке даст кратчайший путь.

Пример.

Определить кратчайший путь от вершины v_1 к вершине v_5 для графа





Алгоритм более удобен для вычисления вручную

I-й алгоритм Флойда-Уоршолла

(1) Посмотрим на столбец 1 матрицы A. Если в j-ой строке столбца имеется положительное целое число, добавить его к строке 1, чтобы сформировать A_j^1. Положить A_j равной j-ой строке матрицы A и заменить j-ую строку матрицы A на $A_j^1 \wedge A_j$. Назвать эту матрицу $A^{(1)}$.

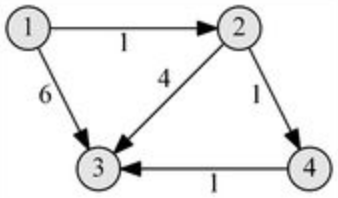
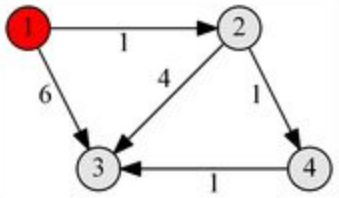
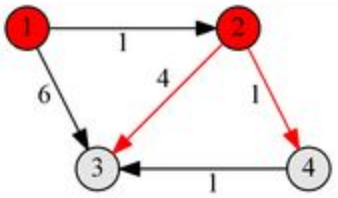
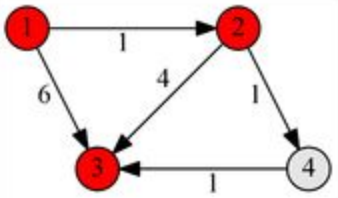
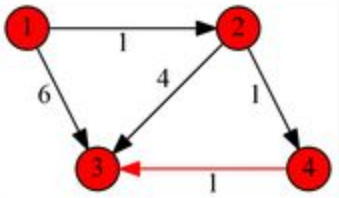
(2) Рассмотреть столбец 2 матрицы $A^{(1)}$, построенной на шаге 1. Если в j-ой строке столбца имеется положительное целое число, добавить его к строке 2, чтобы сформировать A_j^2. Положить A_j равной j-ой строке матрицы A и заменить j-ую строку матрицы $A^{(1)}$ на $A_j^2 \wedge A_j$. Назвать эту матрицу $A^{(2)}$.

(3) Рассмотреть столбец 3 матрицы $A^{(2)}$, построенной на шаге 2. Если в j-ой строке столбца имеется положительное целое число, добавить его к строке 3, чтобы сформировать A_j^3. Положить A_j равной j-ой строке матрицы A и заменить j-ую строку матрицы $A^{(2)}$ на $A_j^3 \wedge A_j$. Назвать эту матрицу $A^{(3)}$.

(4) Продолжать процесс, рассматривая следующий, например, i-ый столбец матрицы, построенной на предыдущем шаге. Если в j-ой строке этого столбца имеется положительное целое число, добавить его к соответствующей строке, чтобы сформировать A_j^i. Положить A_j равной j-ой строке матрицы A и заменить j-ю строку матрицы $A^{(i-1)}$ на $A_j^i \wedge A_j$. Назвать эту матрицу $A^{(i)}$.

(5) Продолжать, пока все столбцы не будут проверены.

Пример

$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
				
$\begin{pmatrix} \times & 1 & 6 & \infty \\ \infty & \times & 4 & 1 \\ \infty & \infty & \times & \infty \\ \infty & \infty & 1 & \times \end{pmatrix}$	$\begin{pmatrix} \times & 1 & 6 & \infty \\ \infty & \times & 4 & 1 \\ \infty & \infty & \times & \infty \\ \infty & \infty & 1 & \times \end{pmatrix}$	$\begin{pmatrix} \times & 1 & \mathbf{5} & \mathbf{2} \\ \infty & \times & 4 & 1 \\ \infty & \infty & \times & \infty \\ \infty & \infty & 1 & \times \end{pmatrix}$	$\begin{pmatrix} \times & 1 & 5 & 2 \\ \infty & \times & 4 & 1 \\ \infty & \infty & \times & \infty \\ \infty & \infty & 1 & \times \end{pmatrix}$	$\begin{pmatrix} \times & 1 & \mathbf{3} & 2 \\ \infty & \times & \mathbf{2} & 1 \\ \infty & \infty & \times & \infty \\ \infty & \infty & 1 & \times \end{pmatrix}$

Алгоритм более удобен для вычисления на компьютере

II-й алгоритм Флойда-Уоршолла

Цикл по k от 1 до n:

    Цикл по i от 1 до n:

        Цикл по j от 1 до n:

            $A_{ij} = A_{ij} \wedge (A_{ik} + A_{kj})$.

Последний слайд лекции

!