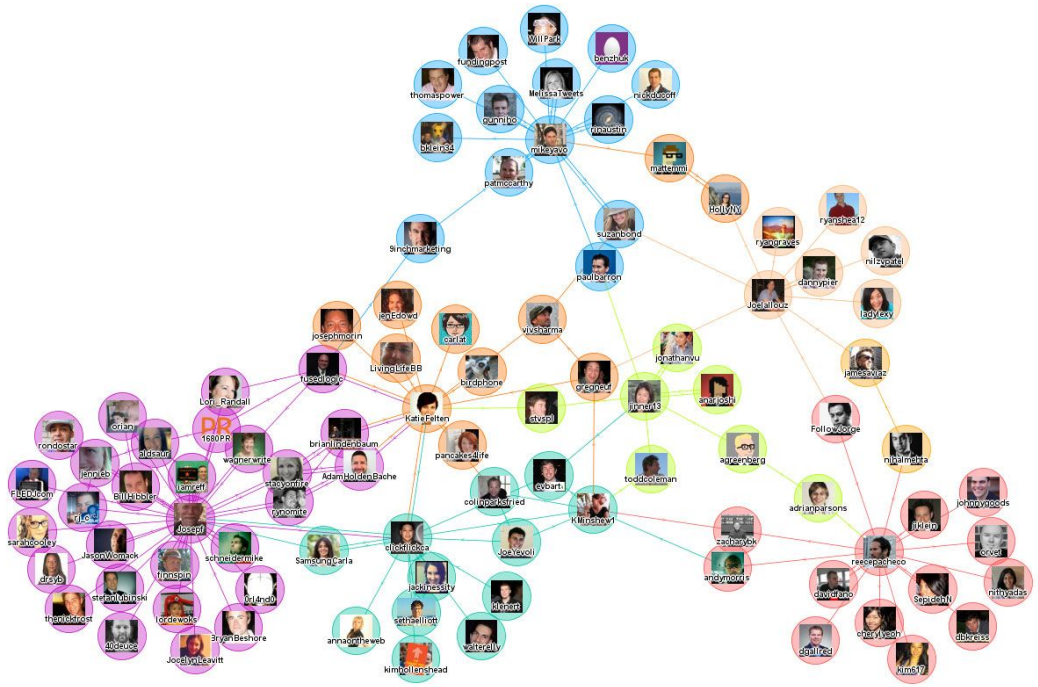
# Связность графов

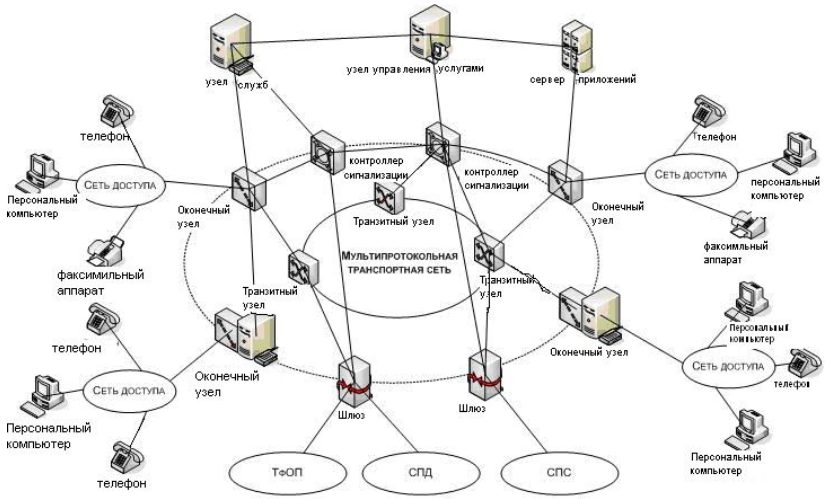
Многие практические задачи программирования требуют определения наличия пути или связи между элементами графа.

Примерами таких задач являются:

1 Анализ социальных графов



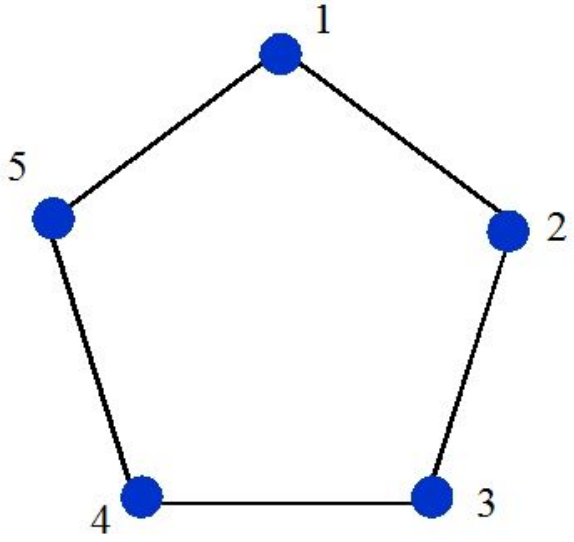
2 Анализ транспортных путей и оптимизация маршрутов

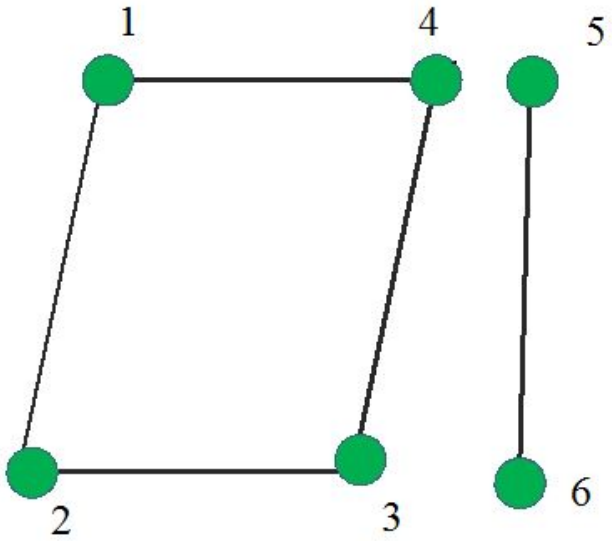




3 Анализ каналов связи и передачи информации

Маршрут в неориентированном графе – последовательность вершин и ребер $v_{i0}$, $e_{i1}$, $v_{i1}$, $e_{i2}$, … , $v_{ik}$, $e_{ik}$, где любые два соседних элемента инцидентны. Т.е. каждая пара вершин $v_i$, $v_{i+1}$ образует ребро (вершины являются смежными).

Граф называют *связным* если любую пару его вершин соединяет какой-нибудь маршрут. Любой общий граф можно разбить на подграфы, каждый из которых окажется связным.




*Компонента связности* - множество вершин графа такое, что из любой вершины этого множества есть путь в любую другую вершину этого множества, но ни из какой вершины этого множества нельзя попасть в некоторую вершину вне этого множества.

Поиск компонент связности реализуется через алгоритмы обхода графа. Для этого совершается серия обходов графа. Каждый обход из некоторой вершины находит одну компоненту связности.

**Алгоритм поиска компонент связности**

**Вход**: G=(V, A) - неориентированный граф, представленный списками смежностей: для каждой вершины v список **Lv** содержит перечень всех смежных с v вершин.

**Выход**: Cw – список с номерами вершин в компонентах связности.

**Алгоритм** ПОИСК(v):

```
1. Пометить v как посещённую NUM[v] = true;
2. Поместить v в список Cv += v;
3. ДЛЯ КАЖДОЙ w принадлежащей Lv ВЫПОЛНЯТЬ
4.     ЕСЛИ NUM[w] = false;
5.    ТО
6.     {
7.        ПОИСК(w);
8.     }
```

**Алгоритм ПКС**

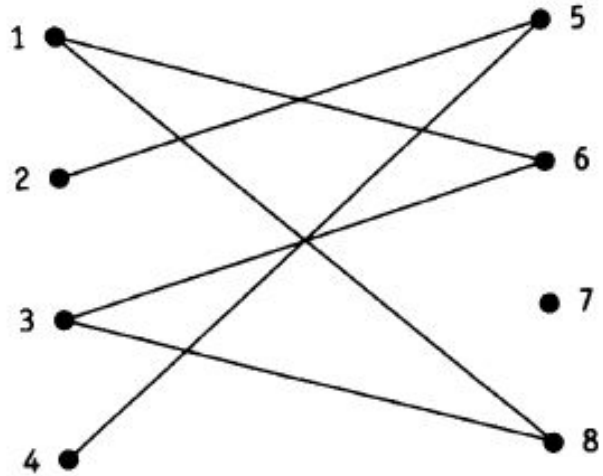
```
1.    Для всех v положим NUM[v] = false - пометим как не посещённую;
2.    Для всех v
3.           ЕСЛИ NUM[v] = false;
4.                  Cv = {};
5.                   ПОИСК(v);
```

Минимальное число связных компонент называется *числом связности* графа и обозначается через *c(G).*

**Алгоритм вычисления числа связности**

**Вход**: G=(V, A) - неориентированный граф.

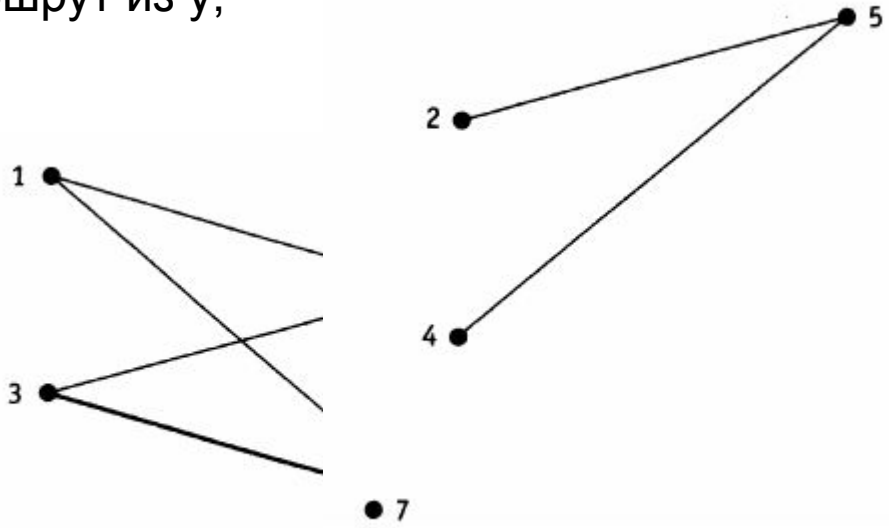
**Выход**: c – число связности.




**Алгоритм ВЧС**

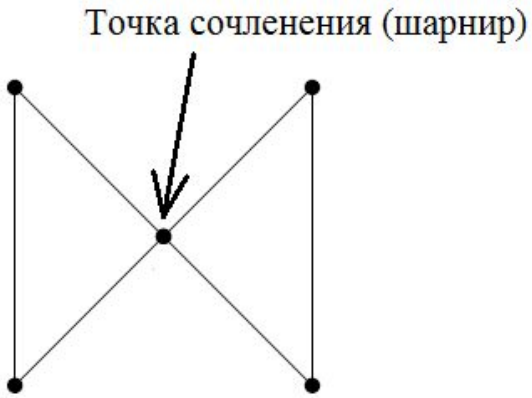
1. V` = V; c=0;
2. **ПОКА** W != ∅ ;
3. Выбрать y ∈ V`;
4. Найти все v ∈ V для которых есть маршрут из y;
5. Удалить y и все найденные v из V`;
6. c += 1.

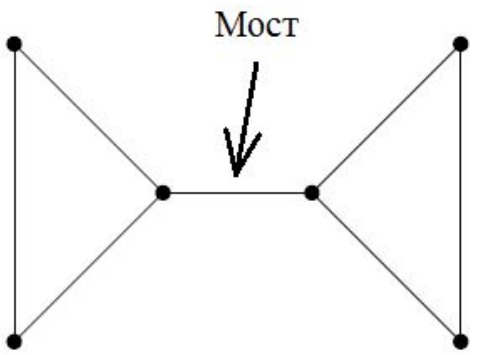
| | $V'$ | $c$ |
|---|---|---|
| Исходные значения | $\{1, 2, 3, 4, 5, 6, 7, 8\}$ | 0 |
| Выбор $y = 1$ | $\{2, 4, 5, 7\}$ | 1 |
| Выбор $y = 2$ | $\{7\}$ | 2 |
| Выбор $y = 7$ | $\varnothing$ | 3 |



Вершина графа называется *точкой сочленения*, если её удаление делает граф несвязным.

*Мостом* называется ребро, удаление которого увеличивает число компонент связности.




Вершинной связностью графа *G*, называется наименьшее число вершин, удаление которых приводит к несвязному или тривиальному (состоящему из одной вершины) графу.

Рёберной связностью графа *G,* называется наименьшее число рёбер, удаление которых приводит к несвязному или тривиальному графу.

Граф называется *k – связным*, если удаление любых *k*-1 вершин не приводит к расчленению графа.

## Связность ориентированных графов

Для ориентированного графа различают сильную, одностороннюю и слабую связность.

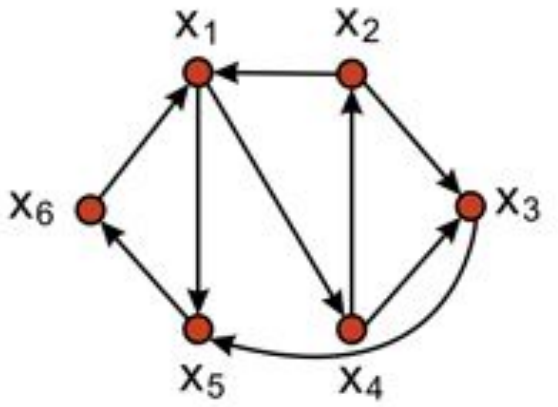
Две вершины *V*1 и *V*2 *сильно связаны, е*сли существует путь (ориентированная цепь) из *V*1 в *V*2 и из *V2* в *V*1*.*

Две вершины *V1 И V*2 *слабо связаны, е*сли они связаны в неориентированном дубликате орграфа.

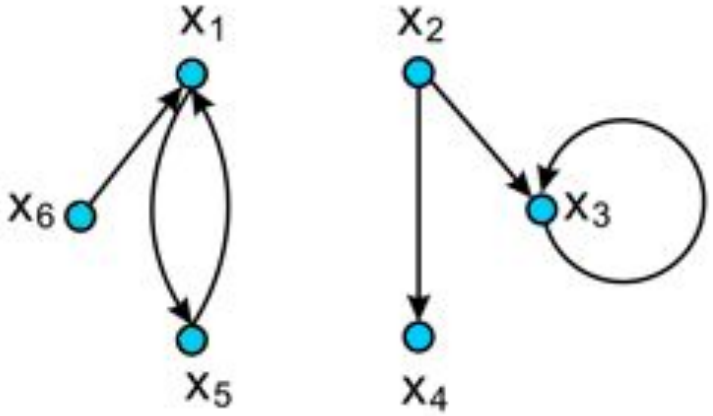
Если все вершины в орграфе сильно связаны, то орграф называется *сильно связным*. Сильная связность влечет слабую связность, но не наоборот.

Орграф является *слабо связным*, если связным графом является его неориентированный дубликат.

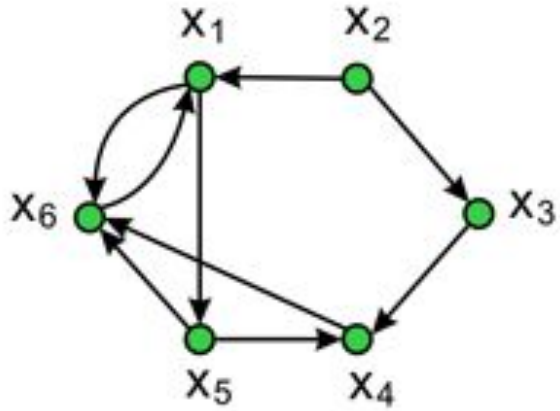
Орграф называется *односторонне связным*, если для любой пары его вершин, по меньшей мере, одна из них достижима из другой.



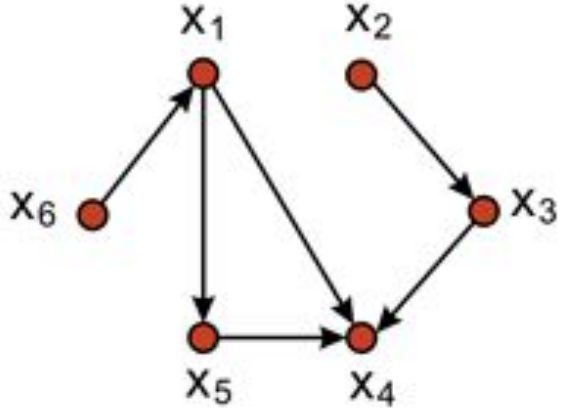

*Сильно связный*.



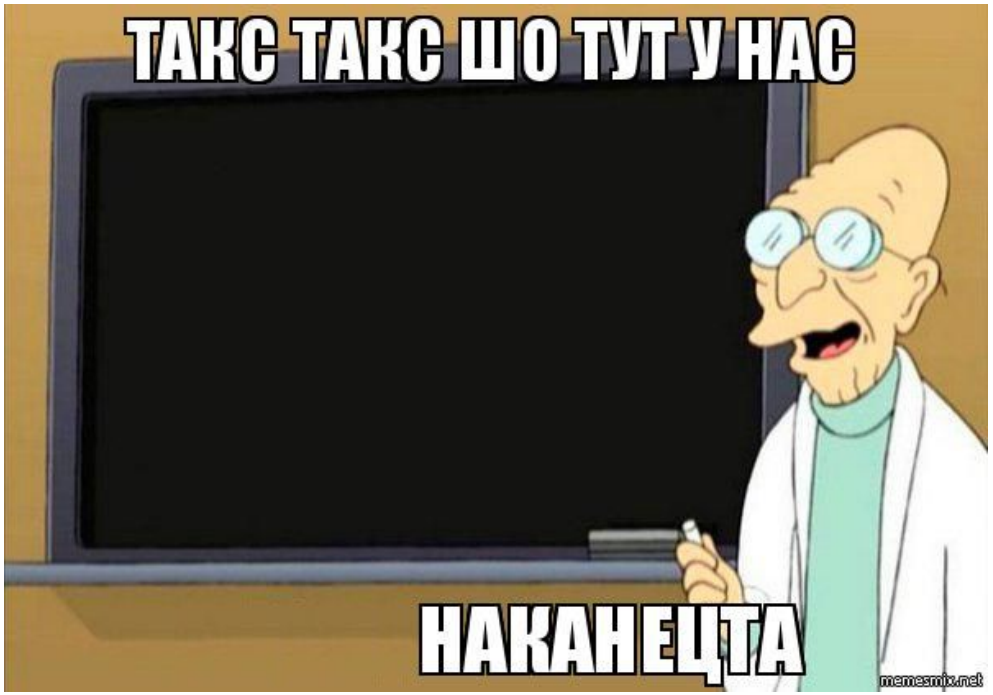

*Не связный*.




*Односторонне связный*,
т.к. нет пути из x1 в x3




*Слабо связный*,
т.к. не существует путей от x2 к x5 и от x5 к x2, но если убрать ориентацию, то эти пути есть

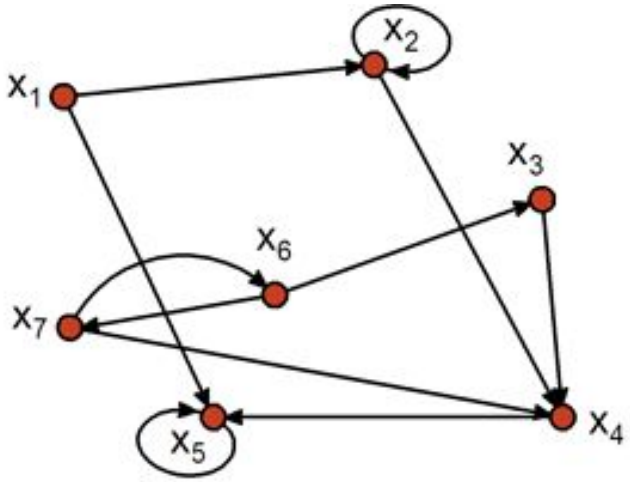



https://forms.gle/hE3aMmNEKbR7Mazz5

# Достижимость в ориентированных графах

Если существует путь из $x_j$ в $x_i$, то говорят, что вершина $x_j$ достижима из вершины $x_i$.

Достижимость в графе описывается матрицей достижимости **R**=[$r_{ij}$], где i и j=1, 2, ... n, n – число вершин графа, а $r_{ij}$=1, если вершина $x_j$ достижима из $x_i$, $r_{ij}$=0, в противном случае.




| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| $x_2$ | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| $x_3$ | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| $x_4$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| $x_5$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| $x_6$ | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| $x_7$ | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

R=

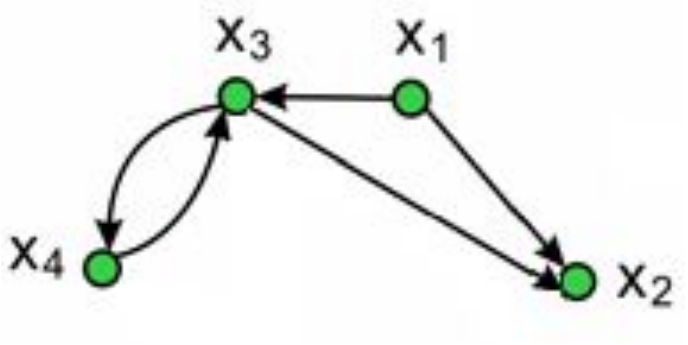
| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|---|---|---|---|
| $x_1$ | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| $x_2$ | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| $x_3$ | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| $x_4$ | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| $x_5$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| $x_6$ | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| $x_7$ | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

## Матричный метод нахождения путей в графе

Матрица смежности графа даёт информацию о всех *путях* длины 1.

Для поиска путей длины 2 достаточно перемножить матрицу смежности саму на себя по правилам перемножения матриц, заменив алгебраические операции логическими.

$$\mathbf{E} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$




$$\mathbf{E^2} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

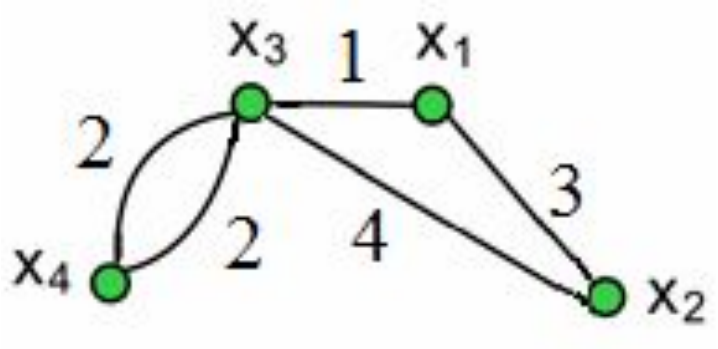
$$e^2{}_{ij} = \left(\sum e_{ik} e_{kj}\right) = \left((e_{i1} \wedge e_{1j}) \vee (e_{i2} \wedge e_{2j}) \vee \ldots \vee (e_{in} \wedge e_{nj})\right)$$

Для поиска путей длины 3 достаточно перемножить матрицу смежности саму на себя 3 раза и т.д.

$$\mathbf{E^3} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad \mathbf{E^4} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

# Кратчайшие пути в графе

Граф называется *взвешенным* (*нагруженным*), если каждому его ребру поставлено в соответствие некое значение – вес или длина ребра.




$$\mathbf{E} = \begin{pmatrix} 0 & 3 & 1 & \infty \\ 3 & 0 & 4 & \infty \\ 1 & 4 & 0 & 2 \\ \infty & \infty & 2 & 0 \end{pmatrix}$$

Длина пути во взвешенном связном графе - это сумма длин (весов) тех ребер, из которых состоит путь.

Тогда можно поставить задачу нахождения кратчайшего пути в графе. Такая задача возникает при оптимизации транспортных путей.

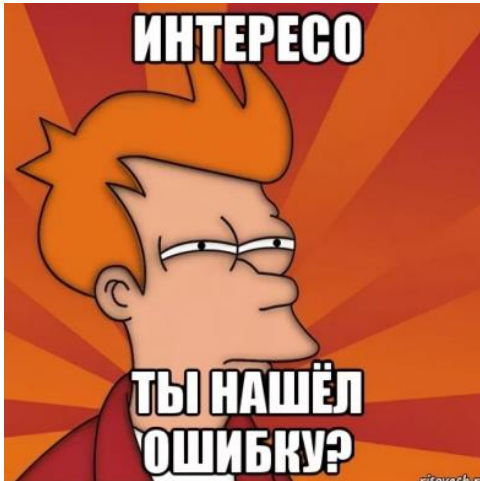
# Алгоритм нахождения кратчайшего пути (алгоритм Дейкстра)

**Вход**: G=(V, A) - неориентированный граф, представленный матрицей смежностей W, s – номер исходной вершины.

**Выход**: Cw – список с номерами вершин кратчайшего пути до вершины w, d[w] – расстояние до вершины w.

**Алгоритм** ДКП:

```
1.  ДЛЯ ВСЕХ v dist[v] = ∞;
2.  dist[s] = 0;
3.  Создать пустой список посещённых вершин S={};
4.  Создать список непосещённых вершин U=V;
5.  ПОКА U != ∅ ;
6.          Выбрать из U элемент u с минимальным расстоянием до s;
7.          Поместить u в список S += u;
8.          ДЛЯ ВСЕХ w являющихся соседями u;
9.                  ЕСЛИ dist[w] > dist[u]+ W(u,w);
10.                         ТО
11.                                 {
12.                                  dist[w]=dist[u]+ W(u,w);
13.                                  Cw +=u;
14.                                  }
```



# Алгоритм Дейкстра. Пример работы.