Лабораторная работа

Практическая линейная алгебра

5

Андрей Зелепугин
03.12.23

Оглавление

Задание 1. Кластеризация социальной сети.

Был придуман связный граф из 15 вершин, всех подписчиков кинул в бан, рассматриваю только друзей. Собственно, вот он:

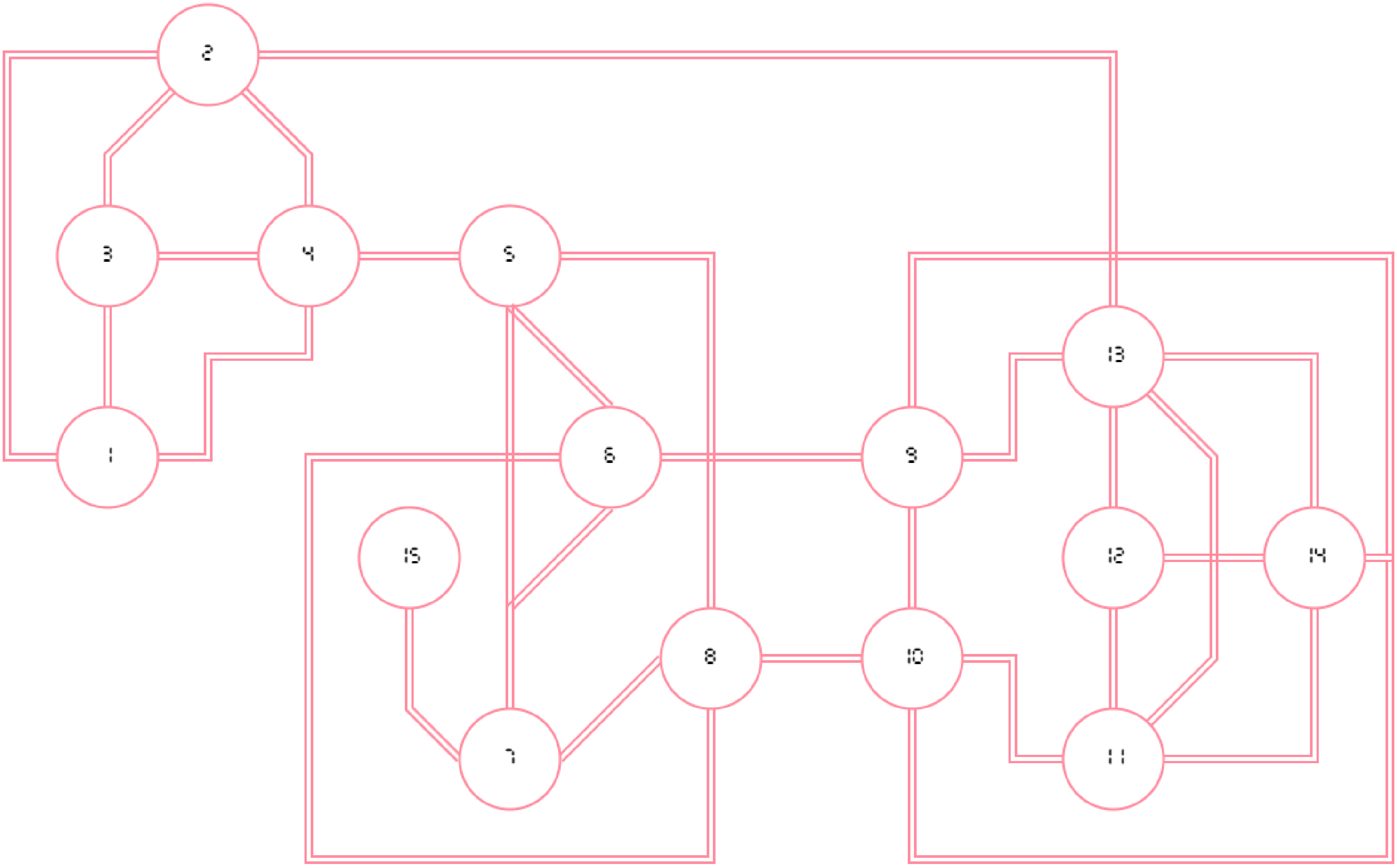


Рис. 1. Связный граф из 15 пронумерованных вершин с визуально различимыми кластерами.

Получил следующий лапласиан:

```
[[3 -1 -1 -1  0  0  0  0  0  0  0  0  0  0  0]
 [-1  4 -1 -1  0  0  0  0  0  0  0  0 -1  0  0]
 [-1 -1  3 -1  0  0  0  0  0  0  0  0  0  0  0]
 [-1 -1 -1  4 -1  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 -1  4 -1 -1 -1  0  0  0  0  0  0  0]
 [ 0  0  0  0 -1  4 -1 -1 -1  0  0  0  0  0  0]
 [ 0  0  0  0 -1 -1  4 -1  0  0  0  0  0  0 -1]
 [ 0  0  0  0 -1 -1 -1  4  0 -1  0  0  0  0  0]
 [ 0  0  0  0  0 -1  0  0  4 -1  0  0 -1 -1  0]
 [ 0  0  0  0  0  0  0 -1 -1  4 -1  0  0 -1  0]
 [ 0  0  0  0  0  0  0  0  0 -1  4 -1 -1 -1  0]
 [ 0  0  0  0  0  0  0  0  0  0 -1  3 -1 -1  0]
 [ 0 -1  0  0  0  0  0  0 -1  0 -1 -1  5 -1  0]
 [ 0  0  0  0  0  0  0  0 -1 -1 -1 -1 -1  5  0]
 [ 0  0  0  0  0  0 -1  0  0  0  0  0  0  0  1]]
```

Без помощи рук нашёл собственные числа и собственные векторы, тут все не привожу, они некрасивые, но их можно найти, если [запустить файл с заданием](#) с [GitHub.](#)

Вижу три компоненты кластеризации, выбираю k=3. Взял 3 собственных вектора v1, . . . , vk матрицы Лапласа, соответствующих самым маленьким собственным числам. Составил из них матрицу V.

```
V=
[[ 1.   -0.77  0.51]
 [ 1.   -0.62  0.27]
 [ 1.   -0.77  0.51]
 [ 1.   -0.56  0.47]
 [ 1.    0.18  0.34]
 [ 1.    0.34  0.13]
 [ 1.    0.53  0.46]
 [ 1.    0.34  0.12]
 [ 1.    0.14 -0.47]
 [ 1.    0.16 -0.52]
 [ 1.    0.04 -0.74]
 [ 1.    0.01 -0.81]
 [ 1.   -0.08 -0.55]
 [ 1.    0.06 -0.69]
 [ 1.    1.    1.  ]]
```

Алгоритм кластеризации k-means вернул [0 0 0 0 2 2 2 2 1 1 1 1 1 1 2]. Соответственно раскрашу точки:

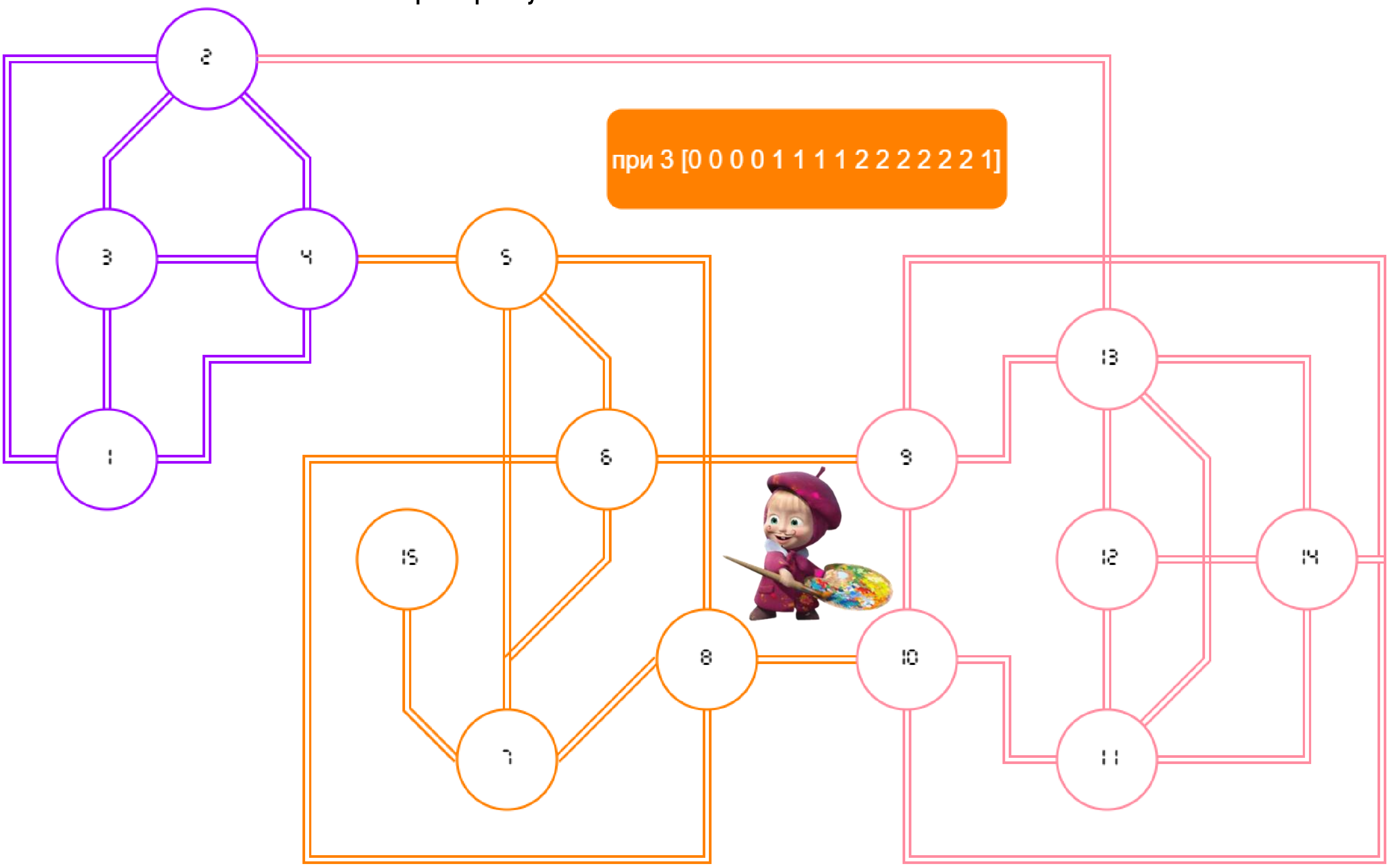


Рис. 2. Граф, разбитый k-means на 3 кластера.

Такой результат меня вполне устроил, я как раз ожидал, что кластерам окажутся 1-2-3-4, 5-6-7-8-15, 9-10-11-12-13-14. Теперь 4:

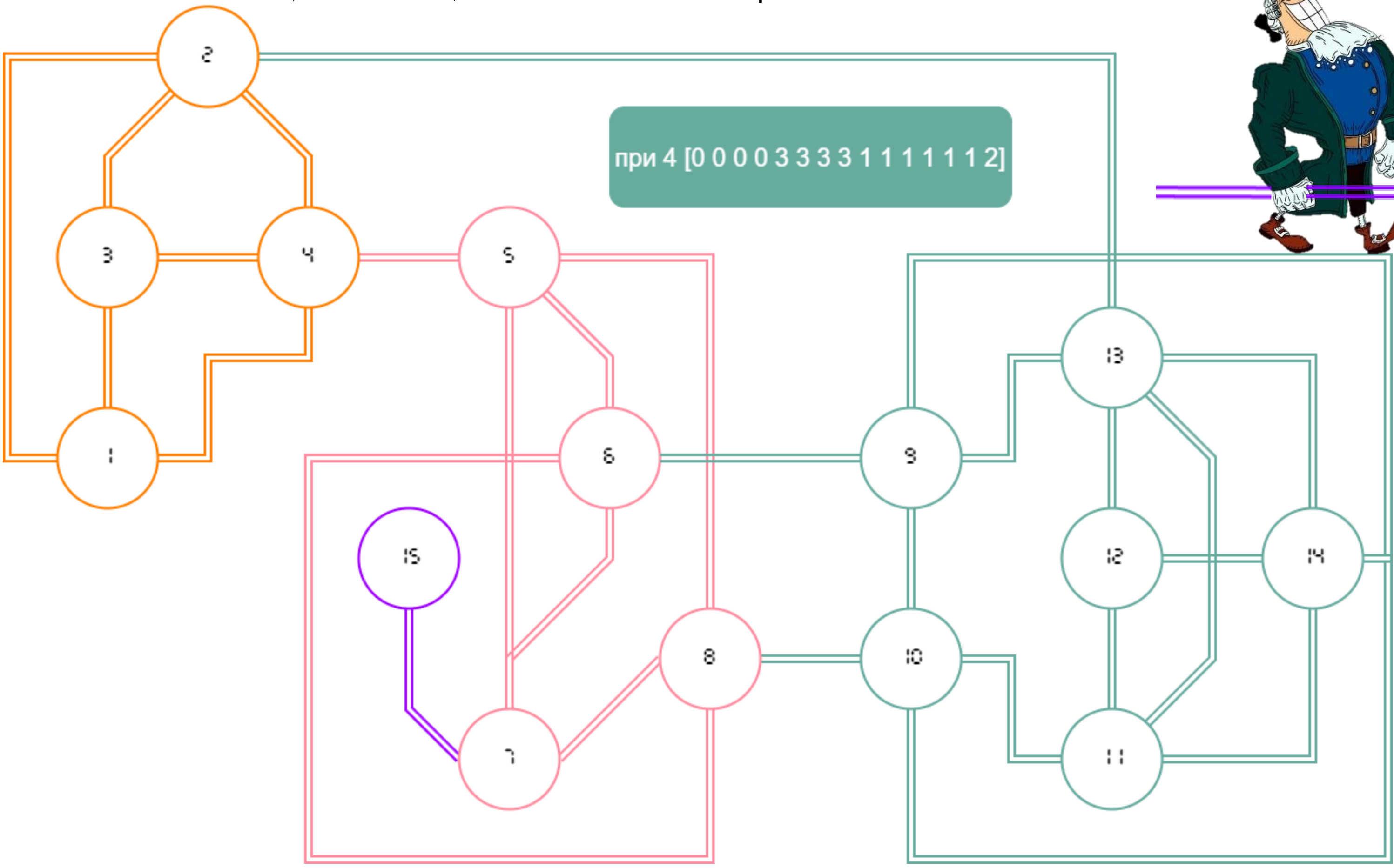


Рис. 3. Разбиение k-means на 4 кластера

Попробуем на 5, тогда

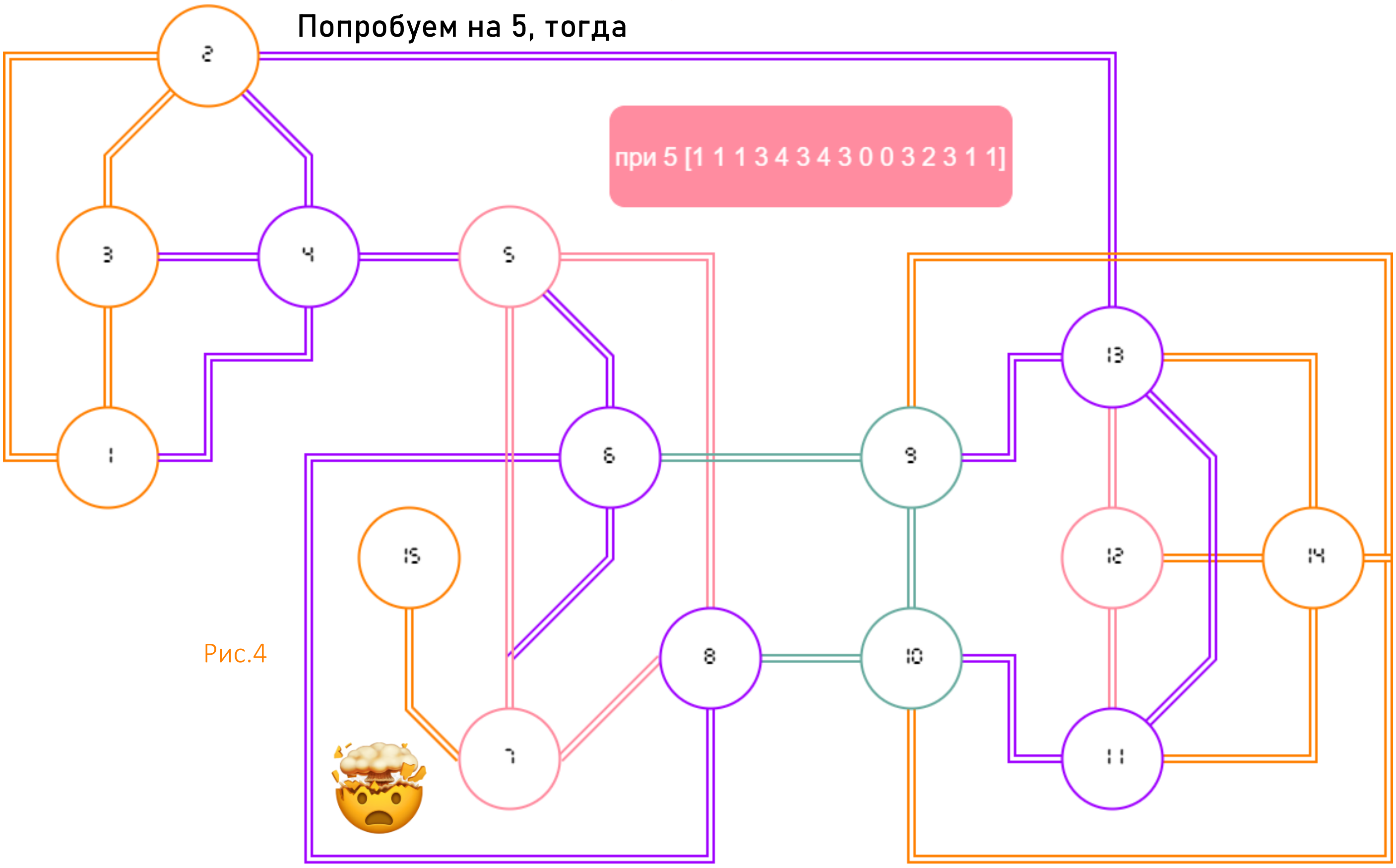


Рис.4

Вопрос: а почему 15 и 14 в одном кластере, а 12 изолирована от всех? Возможно, была ошибка где-то в моём коде, но после проверки собственных векторов, обнаружить её не удалось. Кстати, второе собственное число графа $\lambda = 0.47$.

Попробую сделать это иерархически:

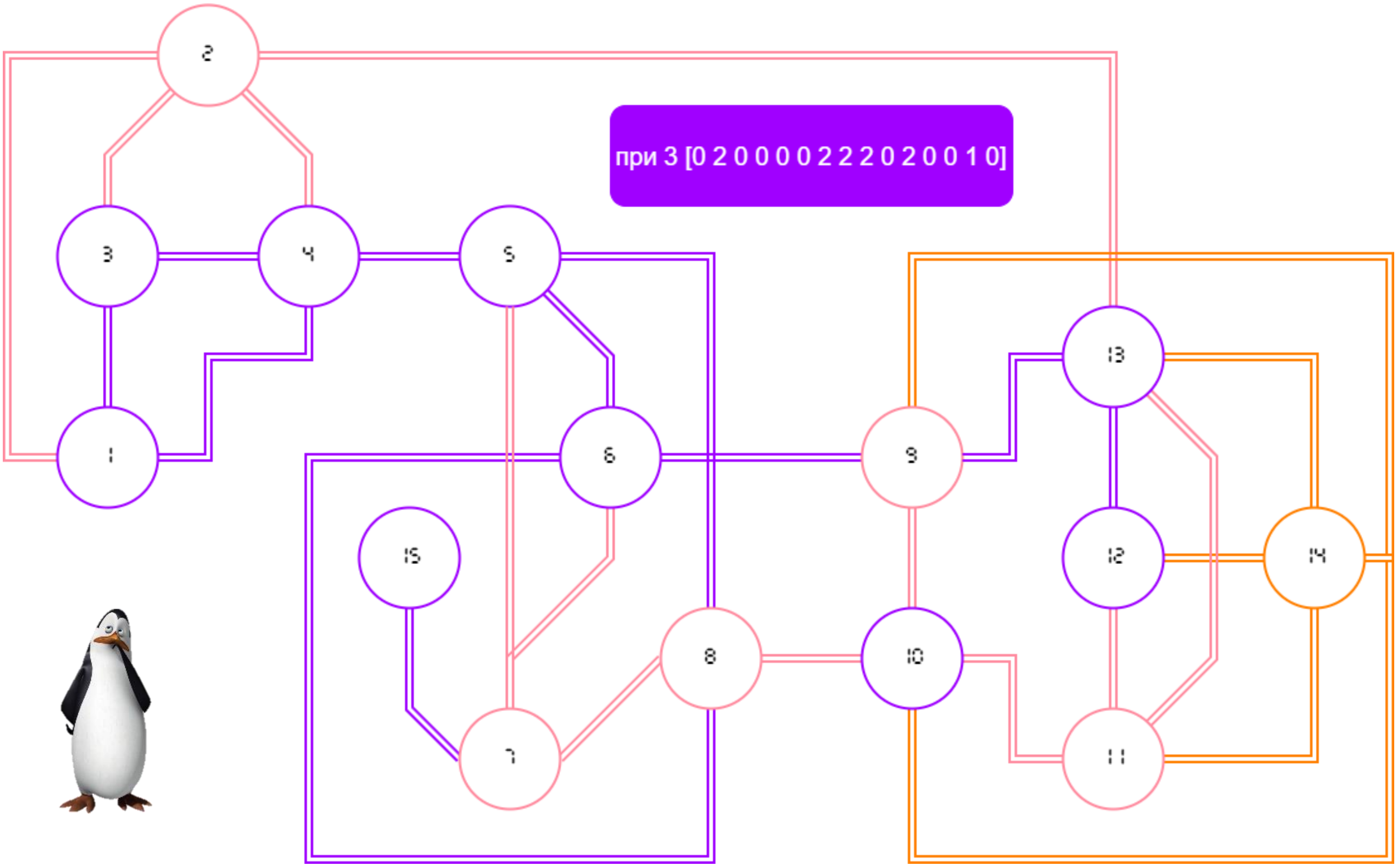


Рис. 5

Так можно по лапласиану из программы восстановить граф: убеждаюсь, что лапласиан всё это время был составлен верно.

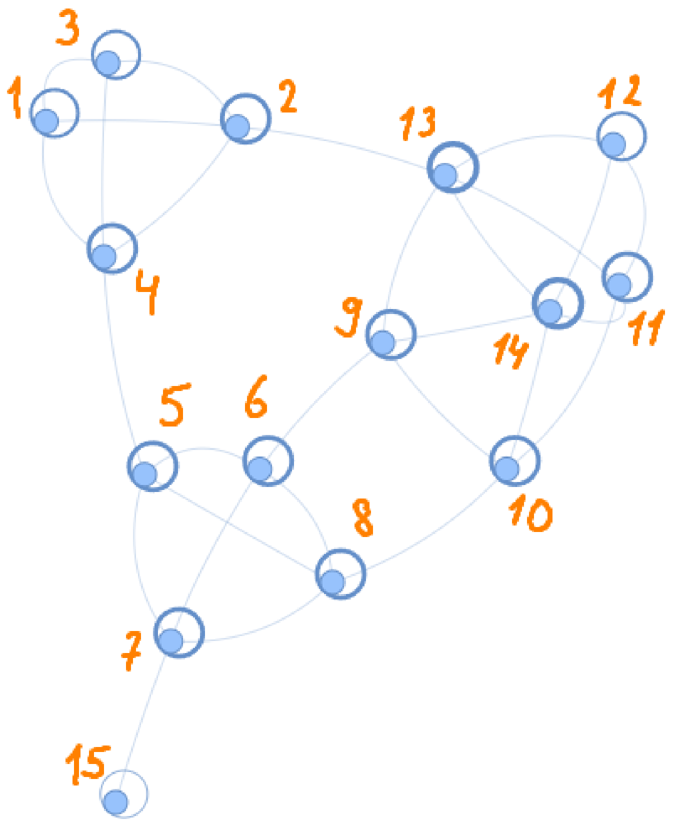
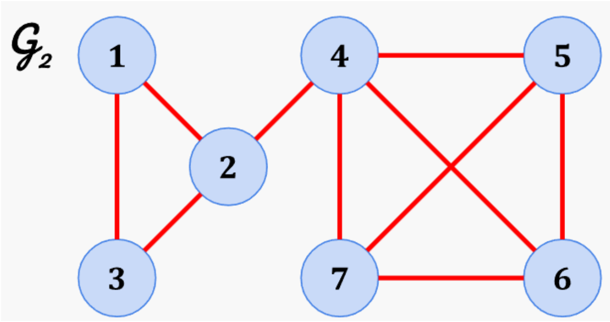
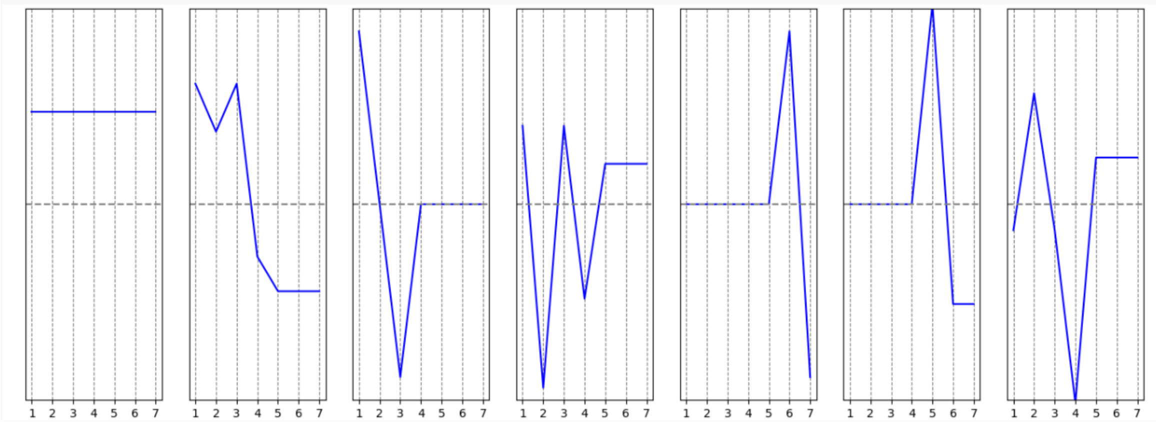


Рис.6 Граф, программно восстановленный по лапласиану с пронумерованными мной вершинами

Программно и математически такой подход к кластеризации может иметь смысл: собственное число 0 – весь граф по сути, раз у нас одна компонента связности, далее рассмотрим рисунок: ([который я взял по ссылке](#))



И отложим его собственные векторы в порядке возрастания соб. чисел по оси вершин:



Второй вектор разделяет вершины: положительные координаты этого вектора соответствуют вершинам 1,2,3, а отрицательные вершинам 4,5,6,7. Если продолжать анализ этого вектора, то можно отметить, что абсолютные значения координат для вершин 1 и 3 одинаковые и больше абсолютного значения координаты для вершины 2, аналогично вершина 4 выделяется относительно вершин 5,6,7. Далее надо придумать и доказать теоремку о достаточности n собственных векторов для разбиения на n кластеров, что я оставляю на совести читателя.

Собственно, в моём случае с векторами была похожая ситуация:

[[1.   -0.77 0.51 0.13 0.83]
 [1.   -0.62 0.27 0.16 0.21]
 [1.   -0.77 0.51 0.13 0.83]
 [1.   -0.56 0.47 -0.07 -0.99]
 [1.   0.18 0.34 -0.61 -2.91]
 [1.   0.34 0.13 -0.62 0.]
 [1.   0.53 0.46 -0.32 -1.95]
 [1.   0.34 0.12 -0.62 -0.14]
 [1.   0.14 -0.47 -0.12 5.]
 [1.   0.16 -0.52 -0.12 4.71]
 [1.   0.04 -0.74 0.26 -0.92]
 [1.   0.01 -0.81 0.4 -6.24]
 [1.   -0.08 -0.55 0.23 -0.45]
 [1.   0.06 -0.69 0.17 1.02]
 [1.   1.   1.   1.   1.]]

Получается, что мы берём группы точек со значениями, и k-means пытается расположить k секторов так, чтобы от точки до сектора было наименьшее расстояние, вероятно, это объясняет, почему встречалась такая ситуация, что две вершины вокруг одной вершины принадлежали одному графу, а сама вершина другому: см. 5 для фиолетовой группы на рис. 4 – просто она была ближе к другому графу через собственные векторы, а на рисунке выглядело иначе. ИТОГО, разбиение работает, потому что при взятии k собственных векторов для наименьших чисел получается каждый раз всё более мелкое дробление, и действительно, если нужно разбить на одну компоненту, собственный вектор имеет одинаковые координаты во всех «точках», если на два, то у половины будет +, у другой -, и в приложении 1 можно найти граф, разбитый на два кластера k-means, заметно соответствие второму столбцу, и далее, продолжая так действовать, мы разобьём граф на нужное число компонент. И так k means приближает группы, а раз они с одинаковым знаком, то и самые близкие.

Задание 2. Google RagePunk

Придумал граф 14 вершин, каждая вершина – альбом, который я слушаю, каждая стрелка – альбом, который могу слушать после этого. Пришлось, кончено, несколько подсократить и оставить только самые прослушиваемые из всех. Вершины пронумеровал слева направо сверху вниз, для ориентира, Черный обелиск оказался под номером 5.

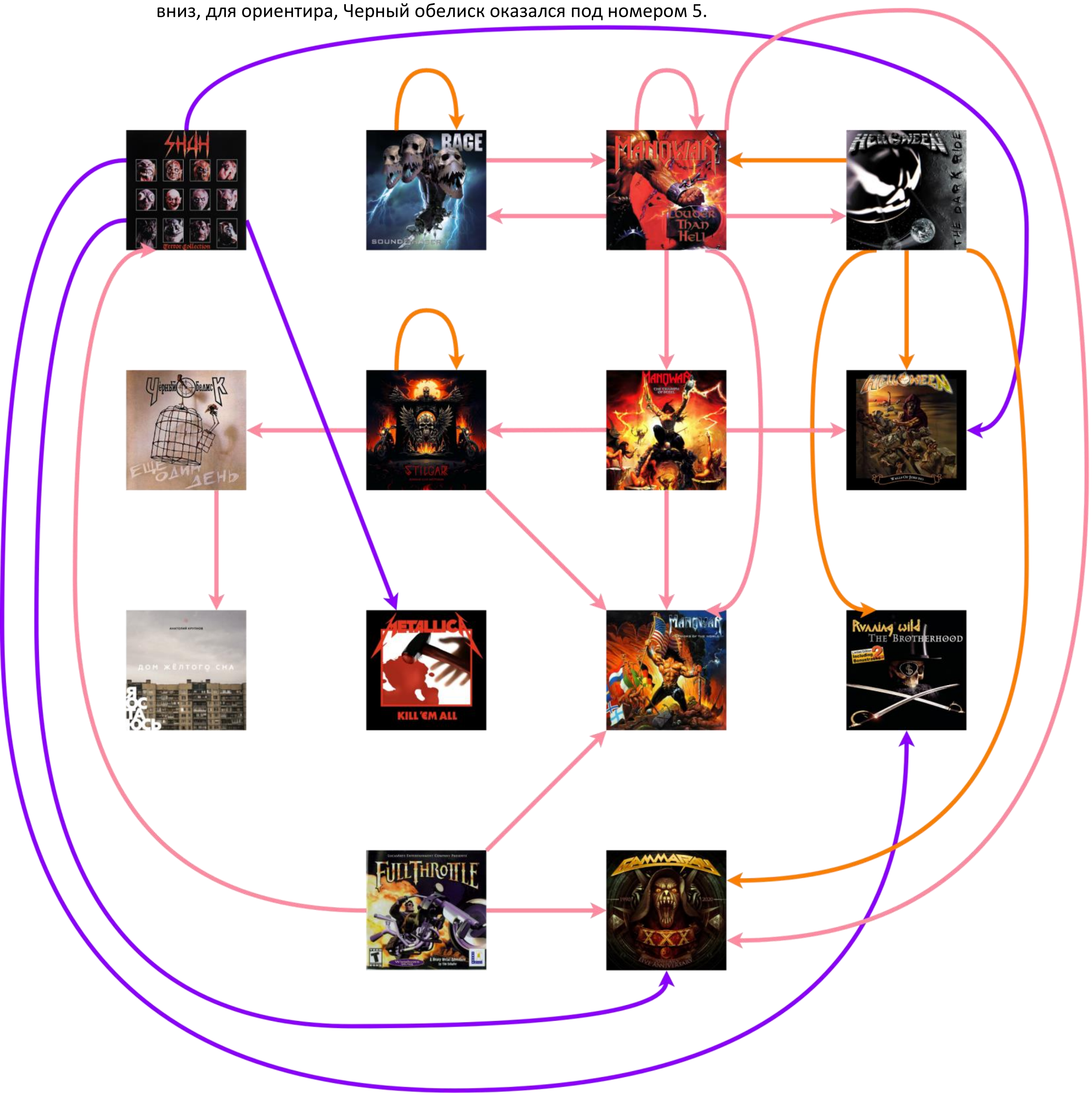
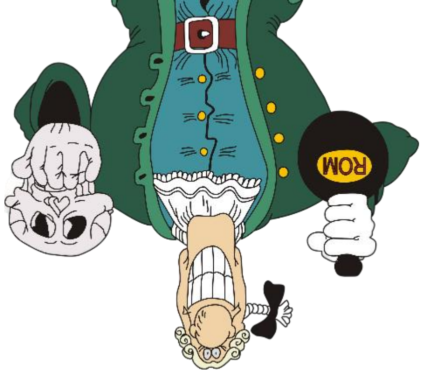


Рис. 7 Граф для задания 2

Составил матрицу M из $m_{ij} = \frac{\text{стрелки из } j \text{ в } i}{\text{всего из } j}$.



M=

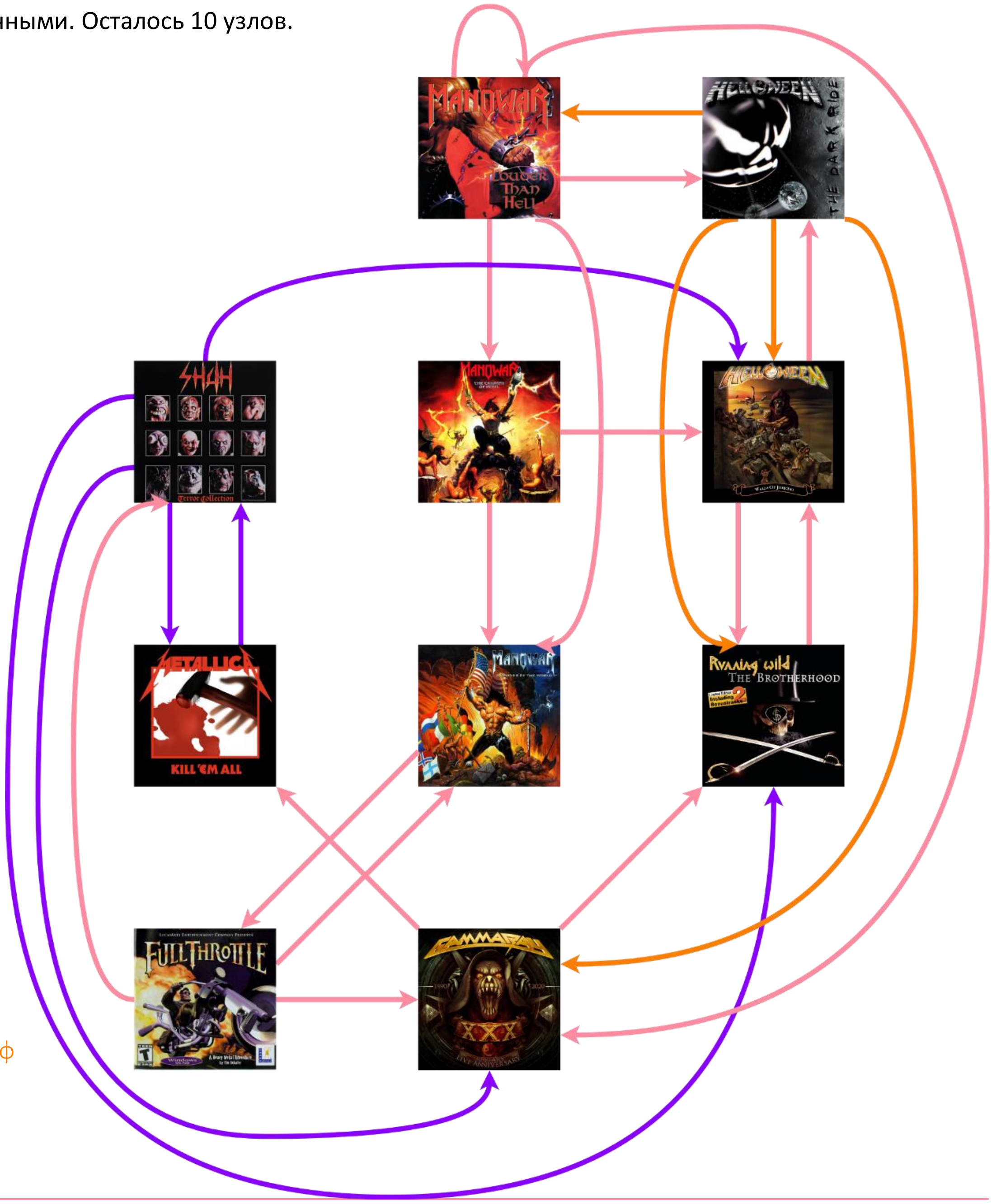
```
[[0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.33 0.  ]
 [0.   0.5  0.17 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.5  0.17 0.25 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.17 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.33 0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.33 0.33 0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.17 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.25 0.   0.   0.25 0.   0.   0.33 0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   1.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.25 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.17 0.   0.   0.   0.33 0.33 0.   0.   0.   0.   0.33 0.  ]
 [0.25 0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.25 0.   0.17 0.25 0.   0.   0.   0.   0.   0.   0.   0.   0.33 0.  ]]
```

Нашел, что наибольшее собственное число 0.68, а ему соответствует

$$v = \begin{pmatrix} 0 \\ 2{,}695 \\ 2{,}962 \\ 0{,}733 \\ 0{,}326 \\ 0{,}678 \\ 0{,}733 \\ 0{,}619 \\ 0{,}474 \\ 0 \\ 1{,}383 \\ 0{,}267 \\ 0 \\ 1 \end{pmatrix}$$

Далее при помощи алгоритма было найдено, что за 5 итераций всё превращается в [0.   0.01 0.01 0.   0.   0.   0.   0.   0.   0.   0.01 0.   0.   0.], что связано с тем, что у меня есть вершины, в которые не входят стрелки, и те, в которые входят сразу много, не выходят. Должно работать не так.

Ловко переоценив ценности, я добавил пару рёбер, вышел за ограничение, убрал те альбомы, что считаю самодостаточными. Осталось 10 узлов.



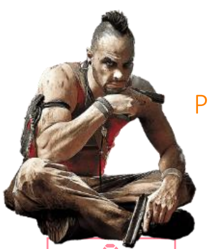


Рис. 8 Настоящий граф

Пересчитал M, пересчитал векторы, появились комплексные собственные числа и векторы, а наибольшее $\lambda = 1$. Выглядит доверительно. Нормализую новый собственный вектор: теперь он [0.05 0.16 0.06 0.01 0.3 0.05 0.02 0.24 0.02 0.07]

$$v = \begin{pmatrix} 0{,}696 \\ 2{,}226 \\ 0{,}804 \\ 0{,}139 \\ 4{,}173 \\ 0{,}701 \\ 0{,}312 \\ 3{,}345 \\ 0{,}312 \\ 1 \end{pmatrix}$$

Запускаю алгоритм, получаю [0.05 0.16 0.06 0.01 0.3 0.05 0.02 0.24 0.02 0.07].

Исходя из этого:










Собственно, секрет прост: алгоритм переходит по всем ссылкам, пока не кончится ограничение по итерациям, а раз я установил 1000 итераций, то сохранится как раз только влияние собственного вектора, соответствующего числу 1. Матрицу M мы так построили, будто с равной вероятностью после просушивания одного альбома переходим к одному из других, то есть это суть матрица Марковского процесса, в этом её «физический» смысл. На всякий случай обращу внимание на то, что по столбцам у неё сумма 1. Получается не что иное, как описание вероятностей. Дальше проще, чем в лекции, трудно объяснить.

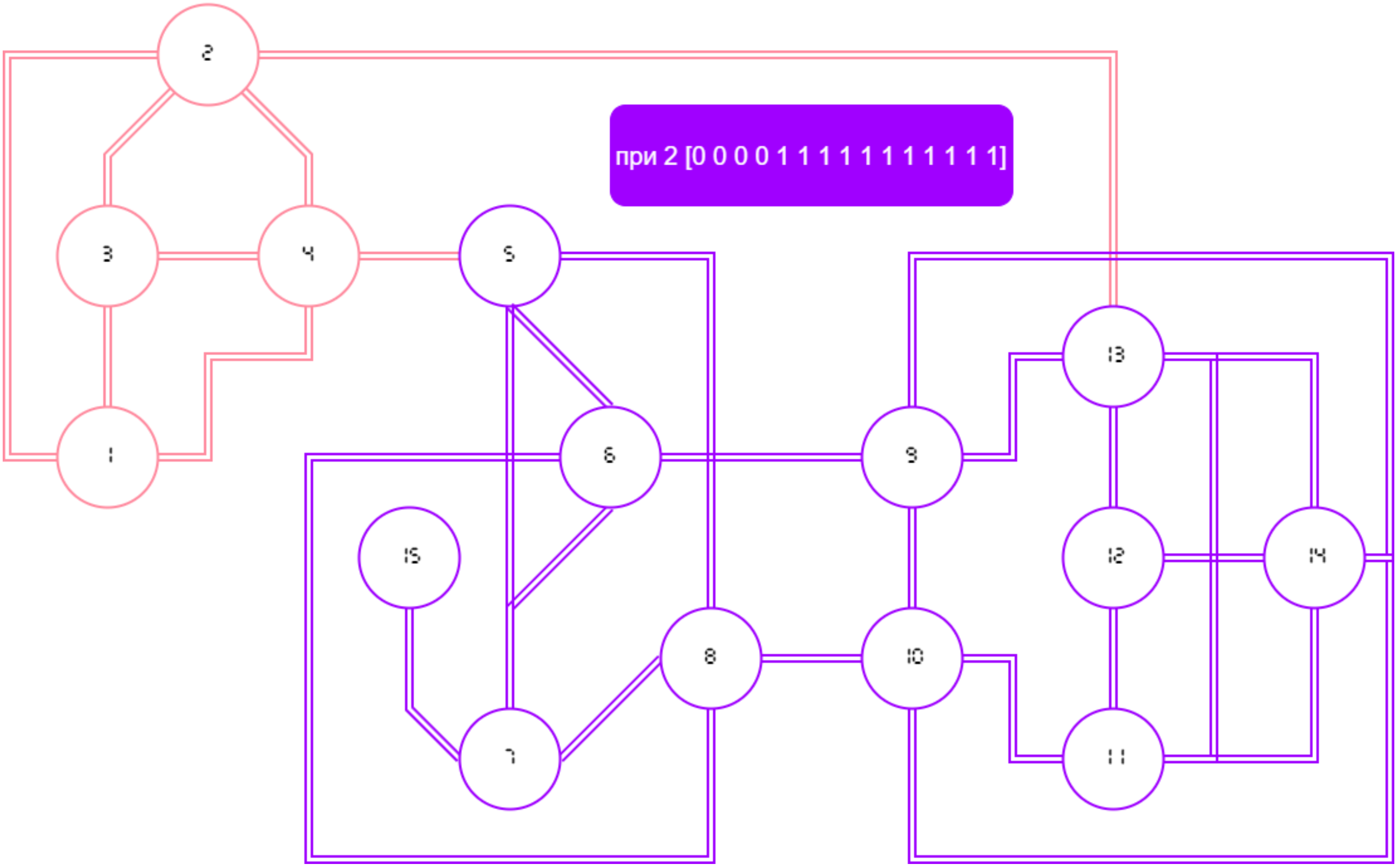
На Helloween и Running Wild действительно было больше всего ссылок, так совпало, что именно их я предпочитаю слушать последними, поэтому в алгоритме они стоят первыми, а с Manowar я предпочитаю начинать.

Другие вектора соответствуют собственным числам меньше 1, поэтому при большом временном промежутке их влияние сойдет на нет, ибо число, меньше 1, возведённое в большую степень – это примерно 0.

Зачем нужен d и почему я перестраивал граф: коэффициент демпфирования d отвечает за то, что я не буду слушать альбомы вечно, а рано или поздно устану, точно также и человек, переходящий по ссылкам рано или поздно перестанет на них кликать. Учитывая, что в формуле идет 1-d, получается, что при d=1 я слушаю альбомы бесперебойно, а пользователь кликает на ссылки вечно. На самом деле такая модель тоже имеет смысл, если брать большой промежуток времени: если мои вкусы не изменятся, то у альбомов, которым присвоен высший PageRank, будет больше прослушиваний, точно также, пользователь, который сегодня пошёл спать, может продолжить интернет-сёрфинг завтра, причём с той же страницы, на которой он остановился.

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

Приложение 1



Приложение 2

M=

```
[[0.2  0.25 0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.2  0.   0.   0.   0.5  0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   1.   0.   0.   0.33 0.  ]
 [0.2  0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.25 0.25 0.5  0.   0.   0.   1.   0.   0.  ]
 [0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.5 ]
 [0.2  0.   0.   0.5  0.   0.   0.   0.   0.33 0.  ]
 [0.   0.25 0.25 0.   0.5  0.   0.   0.   0.   0.5 ]
 [0.   0.   0.   0.   0.   0.   1.   0.   0.   0.  ]
 [0.2  0.25 0.25 0.   0.   0.   0.   0.   0.33 0.  ]]
```


Спасибо за внимание