

Асимптотическая сложности
алгоритмов, O нотация. Элементы
дискретной математики. Введение в
теорию графов.

Хайрулин Сергей Сергеевич
s.khayrulin@gmail.com

Overview

- O - нотация
- Асимптотическая сложность алгоритмов
- Основные понятия теории графов, связанные с графами, орграфами и мультиграфами.
- Ориентированный граф.
- Изоморфизм графов.
- Способы задания графов.
- Матрицы смежности и инцидентности, их свойства.

Литература и др. источники

- Дональд Эрвин Кнут. Искусство программирования (Том 1, 2, 3) // Вильямс 2015.
- Альфред В. Ахо, Джон Э. Хопкрофт, Джеффри Д. Ульман. Структуры данных и алгоритмы // Вильямс 2000.
- Емеличев В. А., Мельников О. И., Сарванов В. И., Тышкевич Р. И. Лекции по теории графов // М.: Наука, 1990.
- Харари Ф. Теория графов // М.: Мир, 1973.
- Косточка А. В. Дискретная математика. Часть 2 //Новосибирск: НГУ, 2001.
- Котов В. Е., Сабельфельд В. К. Теория схем программ // Наука 1991.
- <http://algotlist.manual.ru>
-

О - нотация

Пусть $f(x)$ и $g(x)$ — две функции, определенные в некоторой проколотой окрестности точки x_0 , причем в этой окрестности g не обращается в ноль. Говорят, что:

- f является « O » большим от g при $x \rightarrow x_0$, если существует такая константа $C > 0$, что для всех x из некоторой окрестности точки x_0 имеет место неравенство

$$|f(x)| \leq C|g(x)|;$$

- f является « o » малым от g при $x \rightarrow x_0$, если для любого $C > 0$ найдется такая проколотая окрестность U'_{x_0} точки x_0 , что для всех $x \in U'_{x_0}$ имеет место неравенство

$$|f(x)| < C|g(x)|.$$

Иначе говоря, в первом случае отношение $|f|/|g|$ в окрестности точки x_0 ограничено сверху, а во втором оно стремится к нулю при $x \rightarrow x_0$.

Асимптотическая сложность алгоритмов

Обозначение	Граница	Рост
(Тета) Θ	Нижняя и верхняя границы, точная оценка	Равно
(О - большое) O	Верхняя граница, точная оценка неизвестна	Меньше или равно
(о - малое) o	Верхняя граница, не точная оценка	Меньше
(Омега - большое) Ω	Нижняя граница, точная оценка неизвестна	Больше или равно
(Омега - малое) ω	Нижняя граница, не точная оценка	Больше

Асимптотическая сложность алгоритмов

Алгоритм	Эффективность
$o(n)$	$< n$
$O(n)$	$\leq n$
$\Theta(n)$	$= n$
$\Omega(n)$	$\geq n$
$\omega(n)$	$> n$

Свойства

Транзитивность

$$f(n) = \Theta(g(n)) \wedge g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$$

$$f(n) = O(g(n)) \wedge g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$$

$$f(n) = \Omega(g(n)) \wedge g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$$

$$f(n) = o(g(n)) \wedge g(n) = o(h(n)) \Rightarrow f(n) = o(h(n))$$

$$f(n) = \omega(g(n)) \wedge g(n) = \omega(h(n)) \Rightarrow f(n) = \omega(h(n))$$

Свойства

Рефлексивность

$$f(n) = \Theta(f(n))$$

$$f(n) = O(f(n))$$

$$f(n) = \Omega(f(n))$$

Симметричность

$$f(n) = \Theta(g(n)) \iff g(n) = \Theta(f(n))$$

Свойства

$$C \cdot o(f(n)) = o(f(n))$$

$$C \cdot O(f(n)) = O(f(n))$$

$$o(C \cdot f(n)) = o(f(n))$$

$$O(C \cdot f(n)) = O(f(n))$$

$$o(-f(n)) = o(f(n))$$

$$O(-f(n)) = O(f(n))$$

$$o(f(n)) + o(f(n)) = o(f(n))$$

$$o(f(n)) + O(f(n)) = O(f(n)) + O(f(n)) = O(f(n))$$

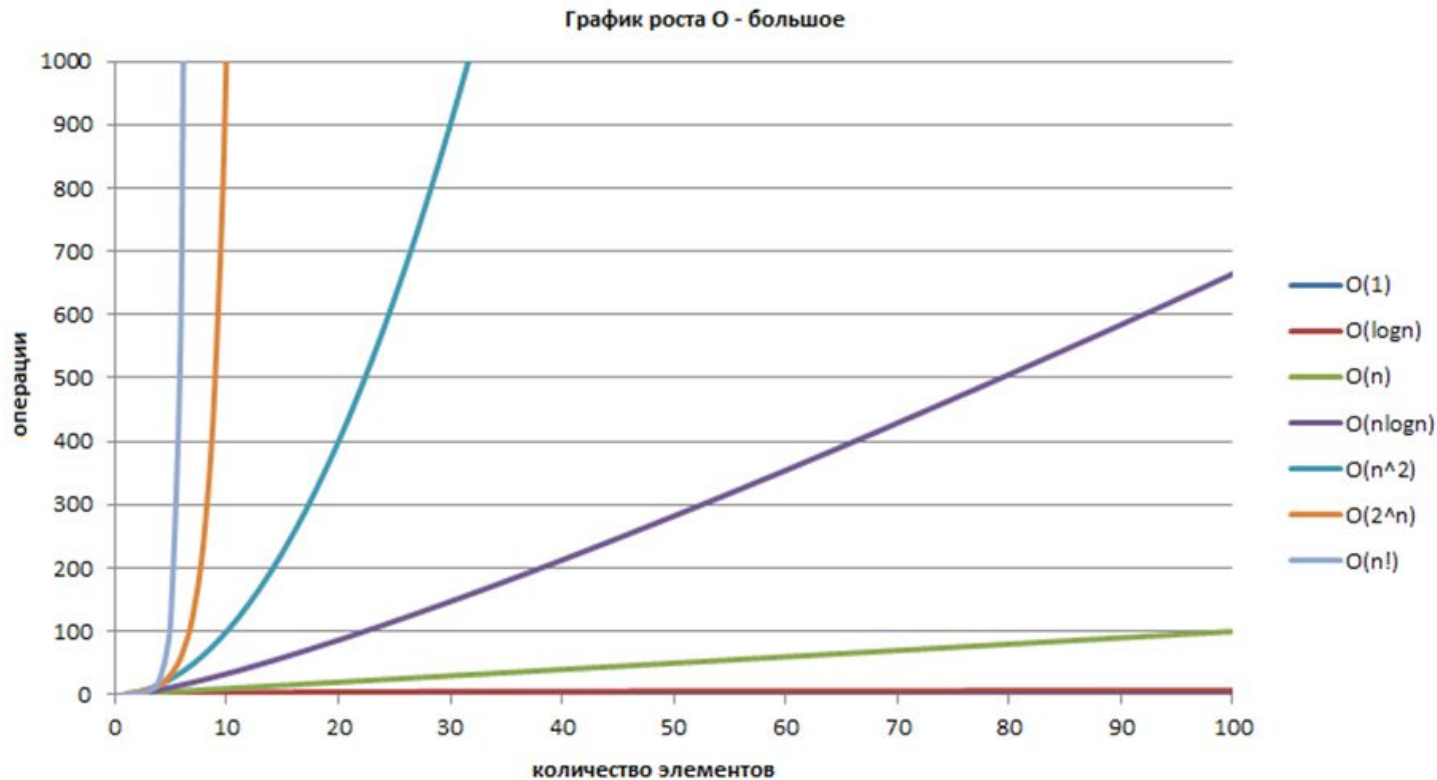
$$O(f(n)) \cdot O(g(n)) = O(f(n) \cdot g(n))$$

$$o(f(n)) \cdot O(g(n)) = o(f(n)) \cdot o(g(n)) = O(f(n) \cdot g(n))$$

$$O(O(f(n))) = O(f(n))$$

$$o(o(f(n))) = o(O(f(n))) = O(o(f(n))) = o(f(n))$$

Асимптотическая сложность алгоритмов



$O(1)$

- Порядок роста $O(1)$ означает, что вычислительная сложность алгоритма не зависит от размера входных данных.

$O(n)$

- Порядок роста $O(n)$ означает, что сложность алгоритма линейно растёт с увеличением входного массива.

$O(n^2)$

Время работы алгоритма с порядком роста $O(n^2)$ зависит от квадрата размера входного массива.

$O(\log(n))$

Порядок роста $O(\log n)$ означает, что время выполнения алгоритма растёт логарифмически с увеличением размера входного массива.

$$O(e^n)$$

Временная сложность алгоритма экспоненциально зависит от данных.

$O(n!)$

Задачи связанные с полным перебором обычно решаются за факториальное время. Например задача коммивояжёра решаемая методом полного перебора.

Поиск

Алгоритм	Структура данных	Временная сложность		Сложность по памяти
		В среднем	В худшем	В худшем
Поиск в глубину (DFS)	Граф с $ V $ вершинами и $ E $ ребрами	-	$O(E + V)$	$O(V)$
Поиск в ширину (BFS)	Граф с $ V $ вершинами и $ E $ ребрами	-	$O(E + V)$	$O(V)$
Бинарный поиск	Отсортированный массив из n элементов	$O(\log(n))$	$O(\log(n))$	$O(1)$
Линейный поиск	Массив	$O(n)$	$O(n)$	$O(1)$
Кратчайшее расстояние по алгоритму Дейкстры используя двоичную кучу как очередь с приоритетом	Граф с $ V $ вершинами и $ E $ ребрами	$O((V + E) \log V)$	$O((V + E) \log V)$	$O(V)$
Кратчайшее расстояние по алгоритму Дейкстры используя массив как очередь с приоритетом	Граф с $ V $ вершинами и $ E $ ребрами	$O(V ^2)$	$O(V ^2)$	$O(V)$
Кратчайшее расстояние используя алгоритм Беллмана-Форда	Граф с $ V $ вершинами и $ E $ ребрами	$O(V E)$	$O(V E)$	$O(V)$

<https://habrahabr.ru/post/188010/>

Сортировка

Алгоритм	Структура данных	Временная сложность			Вспомогательные данные
		Лучшее	В среднем	В худшем	В худшем
Быстрая сортировка	Массив	$O(n \log(n))$	$O(n \log(n))$	$O(n^2)$	$O(n)$
Сортировка слиянием	Массив	$O(n \log(n))$	$O(n \log(n))$	$O(n \log(n))$	$O(n)$
Пирамидальная сортировка	Массив	$O(n \log(n))$	$O(n \log(n))$	$O(n \log(n))$	$O(1)$
Пузырьковая сортировка	Массив	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$
Сортировка вставками	Массив	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$
Сортировка выбором	Массив	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$
Блочная сортировка	Массив	$O(n+k)$	$O(n+k)$	$O(n^2)$	$O(nk)$
Поразрядная сортировка	Массив	$O(nk)$	$O(nk)$	$O(nk)$	$O(n+k)$

<https://habrahabr.ru/post/188010/>

Структуры данных

Структура данных	Временная сложность								Сложность по памяти
	В среднем				В худшем				В худшем
	Индексация	Поиск	Вставка	Удаление	Индексация	Поиск	Вставка	Удаление	
Обычный массив	$O(1)$	$O(n)$	-	-	$O(1)$	$O(n)$	-	-	$O(n)$
Динамический массив	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Односвязный список	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$
Двусвязный список	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$
Список с пропусками	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n \log(n))$
Хеш таблица	-	$O(1)$	$O(1)$	$O(1)$	-	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Бинарное дерево поиска	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Декартово дерево	-	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	-	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Б-дерево	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$
Красно-черное дерево	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$
Расширяющееся дерево	-	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	-	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$
АВЛ-дерево	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$

<https://habrahabr.ru/post/188010/>

Принцип Дирихле

При любом распределении $nk + 1$ или более предметов по n ящикам в каком-нибудь ящике окажется не менее чем $k + 1$ предмет.

Теорема. При любом выборе пяти точек внутри единичного квадрата, найдётся пара точек, удалённых одна от другой менее чем на $\frac{\sqrt{2}}{2}$.

Размещения

Теорема. Общее количество различных наборов при выборе k элементов из n без возвращения и с учётом порядка равняется

$$A_n^k = n(n - 1) \cdot \dots \cdot (n - k + 1) = \frac{n!}{(n - k)!}$$

и называется числом размещений из n элементов по k элементов.

Перестановки

Следствие. Если в множестве n элементов, то существует ровно $n!$ перестановок этих элементов.

Доказательство. Перестановка — результат выбора без возвращения и с учётом порядка n элементов из n . Поэтому общее число перестановок равно $A_n^n = n!$

Сочетания

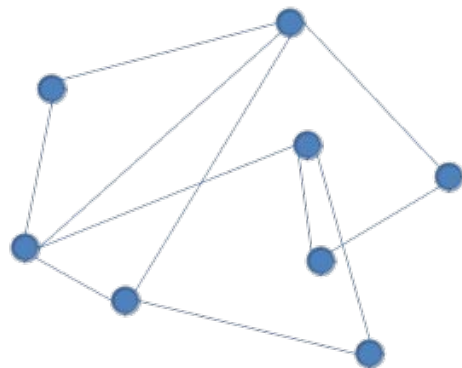
Теорема. Общее количество различных наборов при выборе k элементов из n без возвращения и без учёта порядка равняется

$$C_n^k = \frac{A_n^k}{k!} = \frac{n!}{k!(n - k)!}$$

и называется числом сочетаний из n элементов по k элементов.

Основные понятия теории графов

Граф состоит из конечного множества вершин и некоторого множества неупорядоченных пар вершин называемые ребрами.



Основные понятия теории графов

Маршрут в графе — это такая последовательность вершин и ребер $v_1, e_1, v_2, e_2, v_3, \dots, e_{n-1}, v_n$, что ребро e_i соединяет вершины v_i и v_{i+1} при всех $i = 1, 2, \dots, n-1$. Если $v_1 = v_n$, то говорят, что *соединяет* вершины v_1 и v_n ; если к тому же все вершины v_i различны, то он называется *путем*.

Основные понятия теории графов

Граф называется *связным*, если каждая пара его вершин соединена путем (или, что равносильно, маршрутом).

Маршрут, в котором, называется *замкнутым*, а если при этом он не имеет других повторяющихся вершин, то называется *циклом*. Легко видеть, что любой минимальный по включению замкнутый маршрут есть цикл.

Основные понятия теории графов

- Ребро (u, v) и вершина v *инцидентны*, а степенью $\deg(v)$ вершины v в графе называют число ребер, инцидентных v . Вершина степени 0 называется *изолированной*, а вершина степени 1 — *висячей*.

Для графа G через $V(G)$ и $E(G)$ обозначим множества вершин и ребер G , соответственно.

При работе с графами часто полезны следующих два факта.

Основные понятия теории графов

Полный граф — граф, в котором для каждой пары вершин, существует ребро, инцидентное и инцидентное (каждая вершина соединена ребром с любой другой вершиной).

Клика — подмножество вершин графа, полностью соединённых друг с другом, то есть подграф, являющийся *полным графом*.

Основные понятия теории графов

Лемма о рукопожатиях. Сумма степеней вершин любого графа равна удвоенному числу его ребер.

Лемма о цикле. Если в графе есть хотя бы одно ребро, но нет изолированных и висячих вершин, то в нем есть цикл.

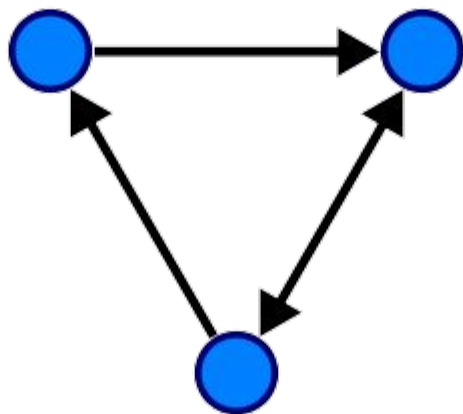
Орграф

- **Ориентированный**

граф (сокращённо **орграф**) G — это граф $G := (V, E)$, где V множество вершин графа, а E множество упорядоченных пар вершин $u, v \in V$.

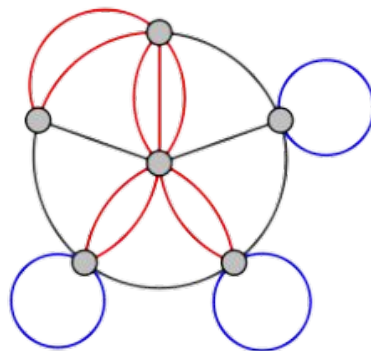
Дуга (u, v) **инцидентна** вершинам u и v . При этом говорят, что u — **начальная вершина** дуги, а v — **конечная вершина**.

Орграф



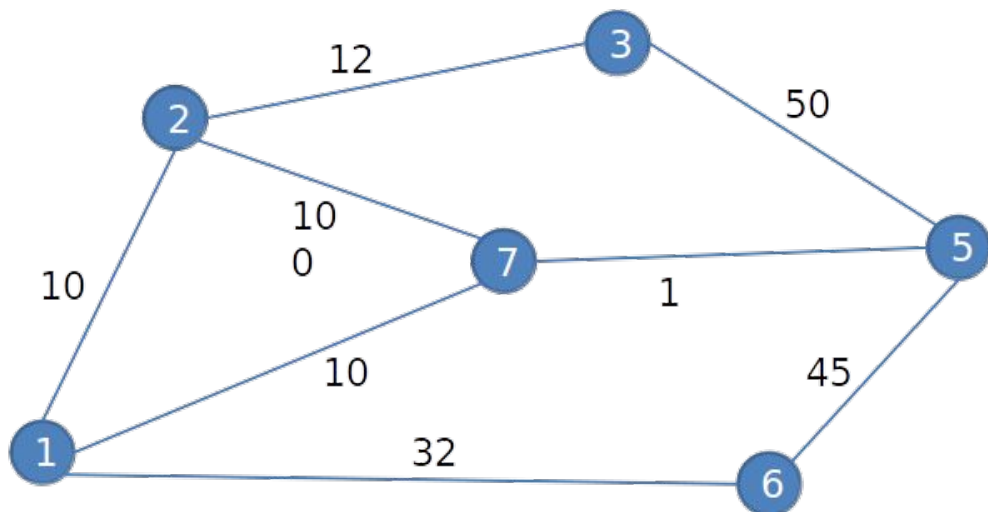
Мультиграф

Мультиграф — граф, в котором может быть пара вершин, которая соединена более чем одним ребром (ненаправленным), либо более чем двумя дугами противоположных направлений.



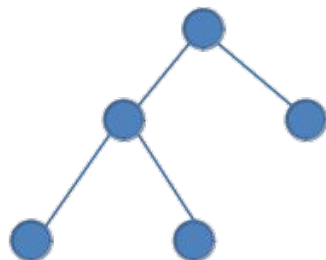
Взвешенный граф

Взвешенный граф — граф, каждому *ребру* которого поставлено в соответствие некое значение (*вес ребра*).



Деревья

Связный граф без циклов называется *деревом*.



Деревья

- Вернемся теперь к рассмотрению деревьев.
Назовем граф *1-конструируемым*, если его можно построить из одновершинного графа последовательным добавлением висячих вершин.

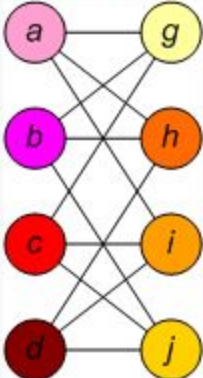
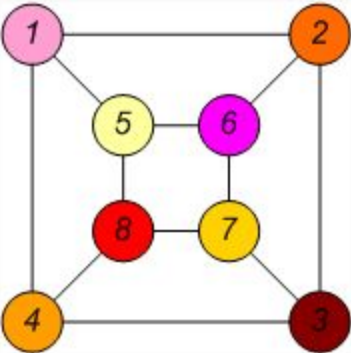
Теорема. Следующие условия для графа $G = (V, E)$ равносильны:

- (1) G — дерево;
- (2) любые две вершины в G соединены ровно одним путем;
- (3) G связен и $|E(G)| \leq |V(G)| - 1$;
- (4) $|E(G)| \geq |V(G)| - 1$ и G не имеет циклов;
- (5) G является 1-конструируемым.

Изоморфизм графов

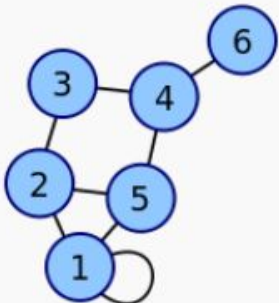
Изоморфизм. Два графа называются **изоморфными**, если существует перестановка вершин, при которой они совпадают. Иначе говоря, два графа называются **изоморфными**, если существует взаимно-однозначное соответствие между их вершинами и рёбрами, которое сохраняет *смежность* и *инцидентность* (графы отличаются только названиями своих вершин).

Изоморфизм графов

Граф G	Граф H	Изоморфизм между графами G и H	Подстановка изоморфизма f
		$f(a) = 1$ $f(b) = 6$ $f(c) = 8$ $f(d) = 3$ $f(g) = 5$ $f(h) = 2$ $f(i) = 4$ $f(j) = 7$	$\begin{pmatrix} a & b & c & d & g & h & i & j \\ 1 & 6 & 8 & 3 & 5 & 2 & 4 & 7 \end{pmatrix}$

Способы задания графов

- **Матрица смежности** G с конечным числом вершин n (пронумерованных числами от 1 до n) — это квадратная матрица A размера n , в которой значение элемента a_{ij} равно числу рёбер из i -й вершины графа в j -ю вершину.

Граф	Матрица смежности
	$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$

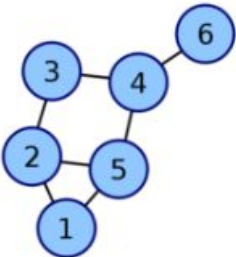
Свойства матрицы смежности

- Матрица смежности неориентированного графа симметрична, а значит обладает действительными собственными значениями и ортогональным базисом из собственных векторов.
- Два графа G_1 и G_2 с матрицами смежности A_1 и A_2 являются изоморфными тогда и только тогда, когда существует перестановочная матрица P , такая что $PA_1P^{-1} = A_2$.
- Из этого следует, что матрицы A_1 и A_2 подобны, а значит имеют равные наборы собственных значений, определители и характеристические многочлены.

Способы задания графов

● **Матрица инцидентности** графа G — одна из форм представления графа, в которой указываются связи между инцидентными элементами графа (ребро(дуга) и вершина). Столбцы матрицы соответствуют ребрам, строки — вершинам. Ненулевое значение в ячейке матрицы указывает связь между вершиной и ребром (их инцидентность).

- В случае ориентированного графа каждой дуге $\langle x, y \rangle$ ставится в соответствующем столбце: « -1 » в строке вершины x и « 1 » в строке вершины y ; если связи между вершиной и ребром нет, то в соответствующую ячейку ставится « 0 ».

Граф	Матрица инцидентности ^[1]
	$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$

Указания для задач

Для замера работы функции нужно использовать метод `now()` класса `datetime` модуля `datetime`

Указания для задач

```
array = [0] * N    import datetime
array.insert(N,0)
```

```
def main():
    t1 = datetime.datetime.now()
    #You'r code here
    ...
    print(datetime.datetime.now() - t1)
```

```
if __name__ == '__main__':
    main()
```

Указания для задач

```
import numpy as np
```

```
...
```

```
# Generate numpy Array with N random numbers
```

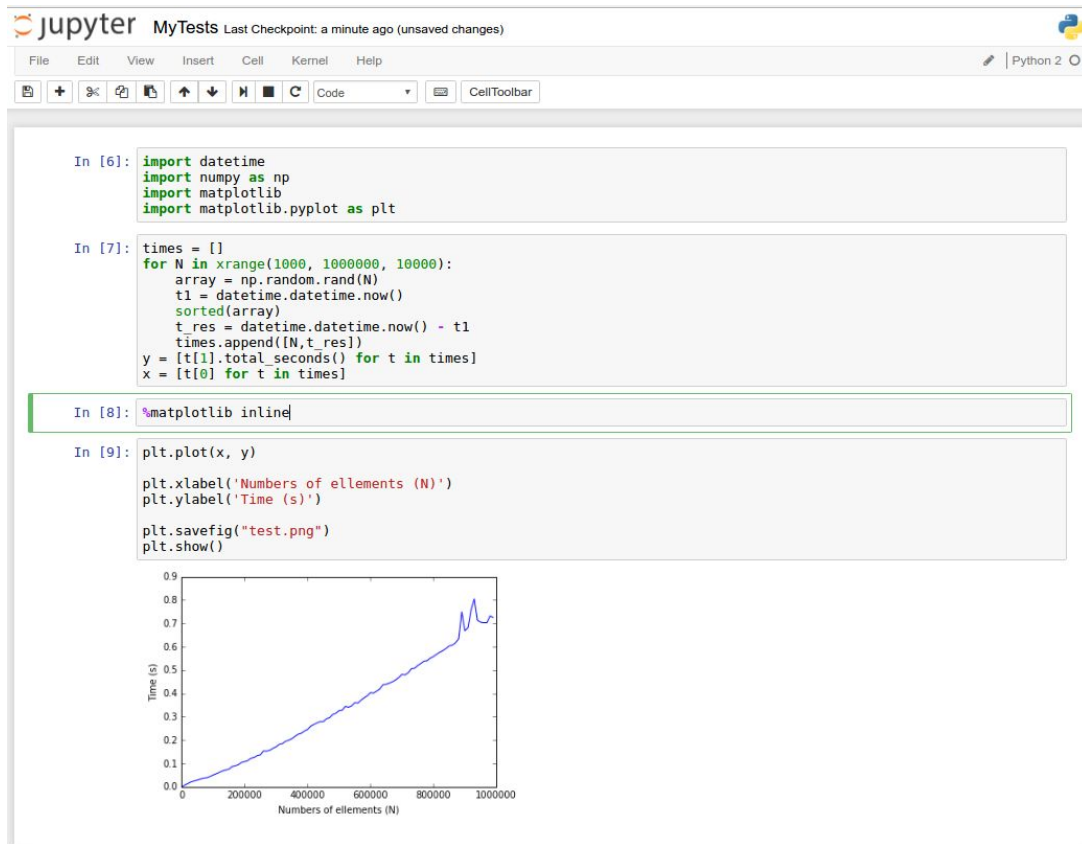
```
array = np.random.rand(N)
```

```
#Sort Array by quick sort
```

```
sorted(array)
```

```
...
```

Указания для задач



Задачи

- Реализовать алгоритм перемножения квадратных матриц. Матрицы могут задаваться как список списков. Считывать можно из файла потока ввода, или задавать случайным образом (используя функцию `pr.random.rand(N)`). Оценить временную и асимптотическую сложность алгоритма, построить график.
- Найти все пифагоровы тройки ($c^2 = a^2 + b^2$) для заданного интервала. Интервал задается парой чисел через пробел считанных из входного потока (например: 10 100) помните, что верхняя грань отрезка должна быть больше нижней. Если задано одно число, то считаем, что ограничение снизу равно по умолчанию 1. Оценить временную и асимптотическую сложность алгоритма, построить график.
- Реализовать алгоритм факторизации числа (разложение числа как произведение двух других чисел). Оценить временную и асимптотическую сложность алгоритма, построить график.
- Реализовать алгоритм рассчитывающий сочетания и размещения.
- Факториал довольно емкостная функция, при расчете которого для больших значений может случиться переполнение (т.е. полученное число будет больше чем максимально возможное число в вашей системе). Подумайте как преодолеть эту проблему.

Задачи

Написать оболочку для работы с графами:

- создавать графы
- Выводить граф (в виде таблицы смежности)
- Удалять ребра
- Ищет путь в графе для заданных вершин

Спасибо за внимание!