Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский Государственный Университет Телекоммуникаций и Информатики»

Кафедра вычислительных систем

Лабораторная работа
по дисциплине «Моделирование»

Выполнили: студенты 4 курса группы ИС-941
Патрушев Н.В.
Сизов М.А.
Проверил: старший преподаватель кафедры ВС
Петухова Я. В.

Новосибирск
2023 г.

Оглавление

Постановка задачи

Написать программу генерирующую случайный граф с заданной высотой и шириной.

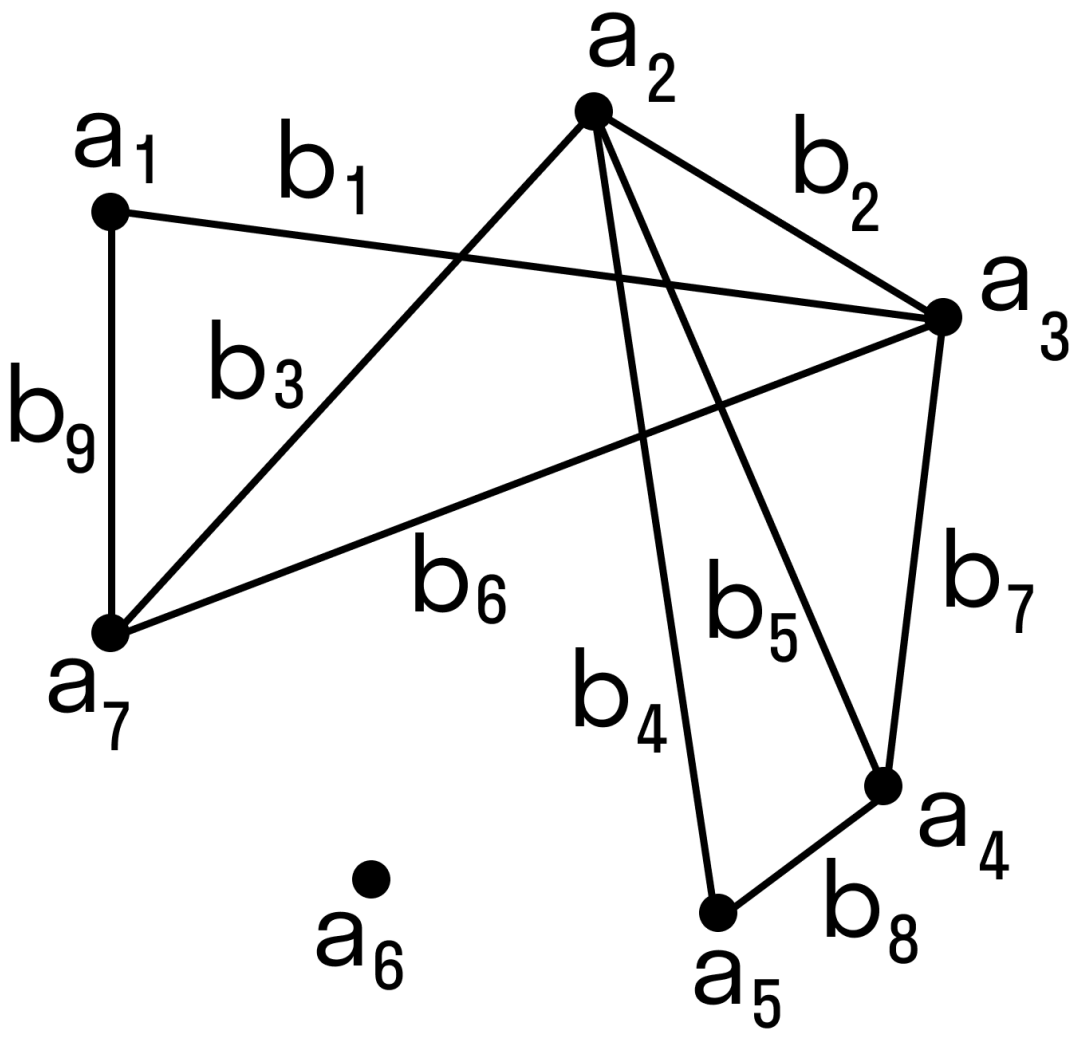
Теория

Граф

В переводе с греческого граф — «пишу», «описываю». В современном мире граф описывает отношения. И наоборот: любое отношение можно описать в виде графа.

Для чего строят графы: чтобы отобразить отношения на множествах. По сути, графы помогают визуально представить всяческие сложные взаимодействия: аэропорты и рейсы между ними, разные отделы в компании, молекулы в веществе.

Пример графа:



Графом называется геометрическая фигура, состоящая из точек и соединяющих их линий. Точки называются вершинами графа, а линии — ребрами**.**

Дерево

Иерархия — это расположение элементов системы в порядке подчиненности (от высшего к низшему). Системы, элементы которых находятся в отношениях «является разновидностью», «входит в состав» и других отношениях подчиненности, называются иерархическими системами (системами с иерархической структурой).

Граф иерархической системы называется деревом**.** Между любыми двумя вершинами этого графа существует единственный путь. Дерево не содержит циклов и петель.

Главный (основной) элемент иерархической системы называется корнем дерева. Каждая вершина дерева (кроме корня) имеет только одного предка — обозначенный ею объект входит в один класс верхнего уровня. Любая вершина дерева может порождать несколько потомков — вершин, соответствующих классам нижнего уровня. Такой принцип связи называется «один ко многим». Вершины, не имеющие порожденных вершин, называются листьями.

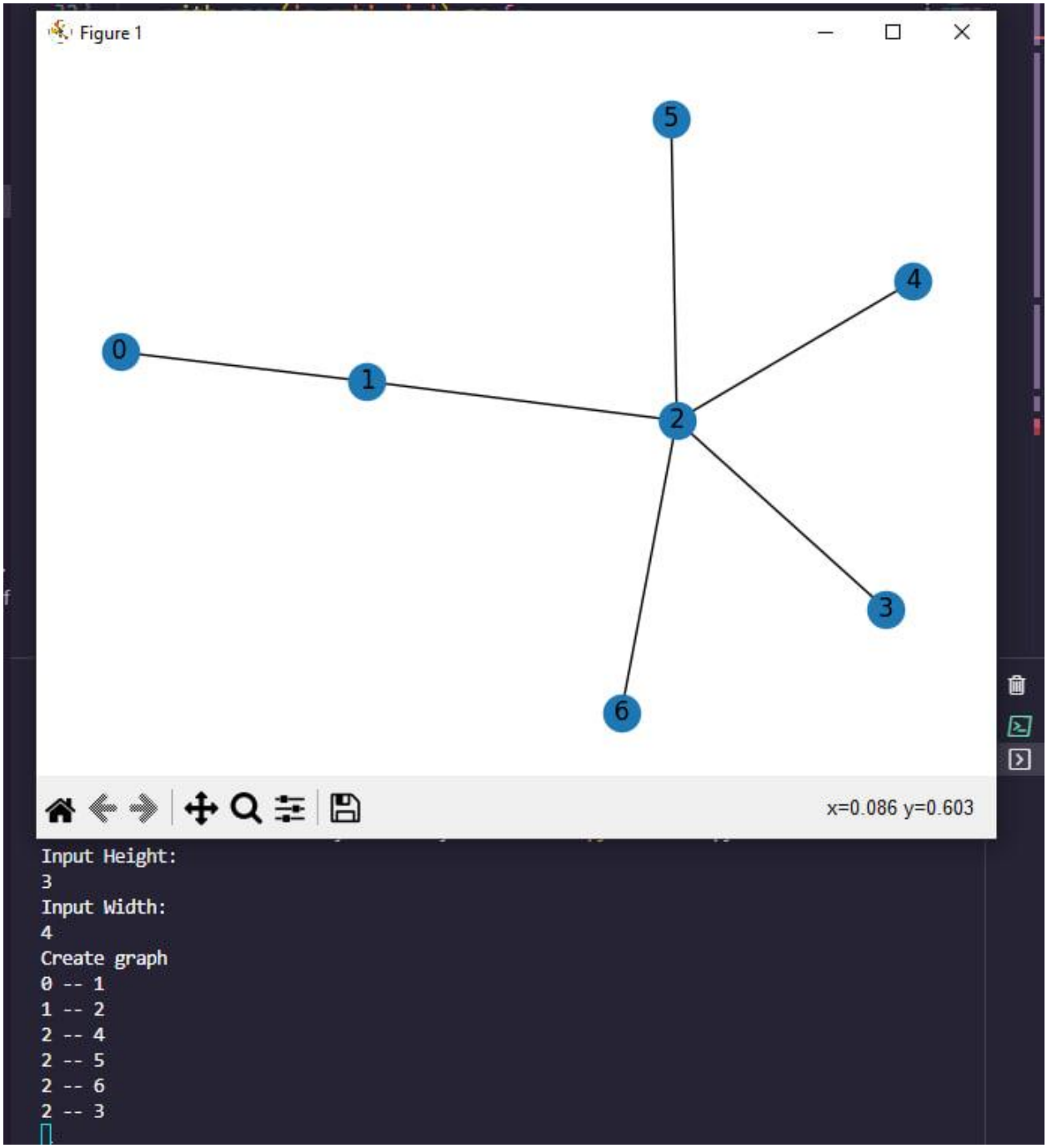
Вихрь Мерсе́нна (англ. *Mersenne twister*, MT) — генератор псевдослучайных чисел (ГПСЧ), алгоритм, разработанный в 1997 году японскими учёными Макото Мацумото (яп. 松本 眞) и Такудзи Нисимура (яп. 西村 拓士).

Вихрь Мерсенна генерирует псевдослучайные последовательности чисел с периодом равным одному из простых чисел Мерсенна, отсюда этот алгоритм и получил своё название и обеспечивает быструю генерацию высококачественных по критерию случайности псевдослучайных чисел.

Ход работы

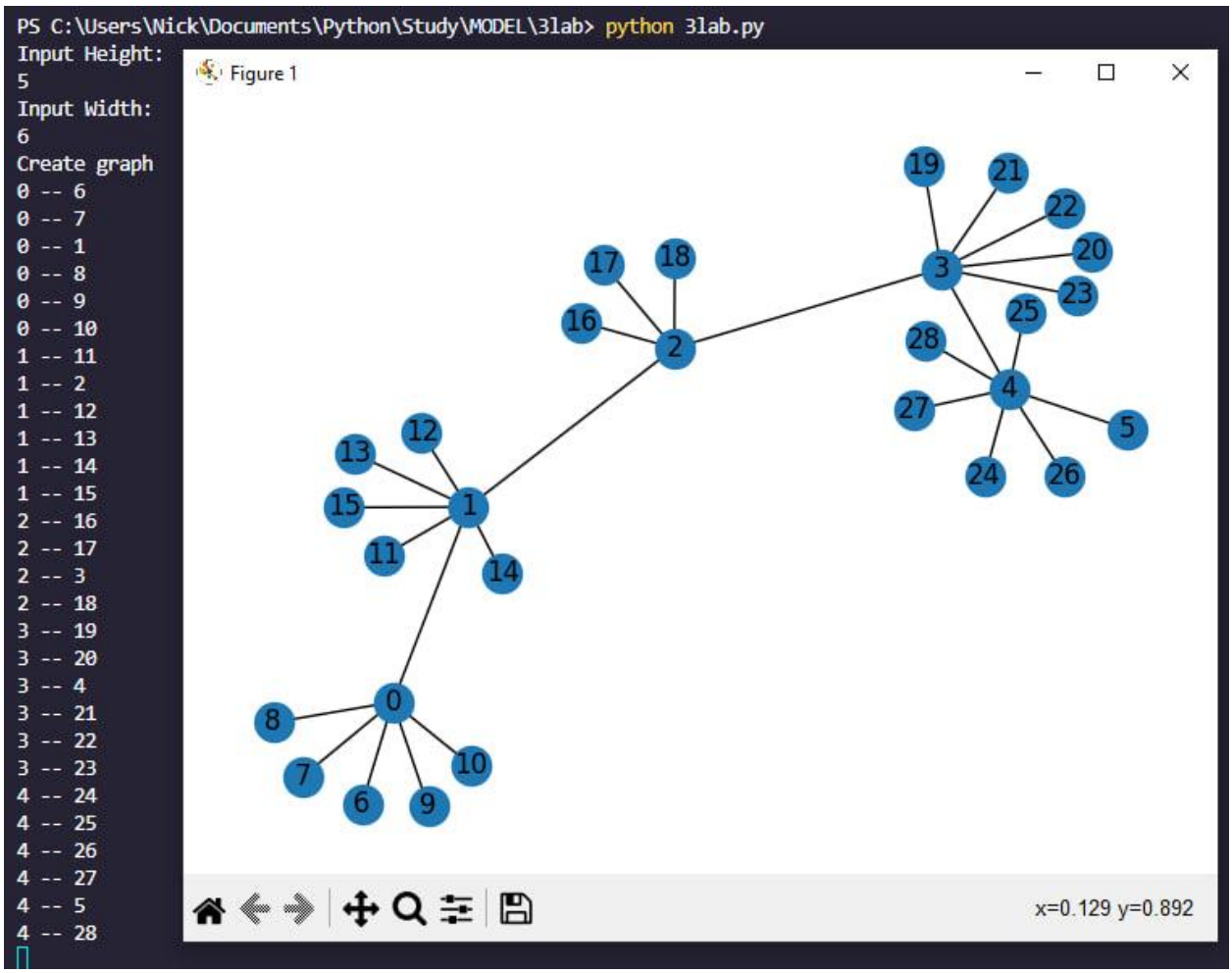
В ходе нашей работы был написан код генерирующий граф с определенной высотой и шириной.

На рисунке 1 изображена матрица связей графа и результат визуализации, при значениях высоты равной три и значения ширины равной четыре.



(Рис 1)

Изменим значения параметров. Используем высоту равной пяти, ширину равной шести (рисунок 2).


PS C:\Users\Nick\Documents\Python\Study\MODEL\3lab> python 3lab.py
Input Height:
5
Input Width:
6
Create graph
0 -- 6
0 -- 7
0 -- 1
0 -- 8
0 -- 9
0 -- 10
1 -- 11
1 -- 2
1 -- 12
1 -- 13
1 -- 14
1 -- 15
2 -- 16
2 -- 17
2 -- 3
2 -- 18
3 -- 19
3 -- 20
3 -- 4
3 -- 21
3 -- 22
3 -- 23
4 -- 24
4 -- 25
4 -- 26
4 -- 27
4 -- 5
4 -- 28
Figure 1
x=0.129 y=0.892

(рисунок 2)

Вывод

Мы использовали равномерный генератор псевдослучайных чисел для генерации графа. В модуле random реализованы различные генераторы псевдослучайных чисел. Здесь присутствуют методы, с помощью которых можно получить равномерное, Гауссовское, бета и гамма распределения и другие функции. Практически все они зависят от метода random(). В Python, в качестве основного, используется генератор псевдослучайных чисел Mersenne Twister, который выдает 53-х битные вещественные числа. Так как генератор, используемый нами, равномерен, мы можем утверждать, что вероятность выпадения данного количества вершин графа равновозможна.

Листинг

3lab.py

```
import networkx as nx
import matplotlib.pyplot as plt
import sys
import random
from io import TextIOWrapper
import sys


def read_graph(f: TextIOWrapper) -> nx.Graph:
    g = nx.Graph()

    s = f.readline()
    l = s.split(' ')
    n = int(l[0])
    g.add_nodes_from(range(n))

    for s in f.readlines():
        l = s.split(' ')
        g.add_edge(int(l[0]), int(l[1]))
        g.add_edge(int(l[1]), int(l[0]))

    return g

def display_graph():
    g = nx.Graph()
    with open('a.gph', 'r') as f:
        g = read_graph(f)
    nx.draw_spring(g, with_labels=True)
    plt.show()

def create_graph(height: int, width: int):
    with open('a.gph', 'w') as f:
        print("Create graph")
        f.write(f'{1} {1}')
        m = 1
        if(width > 1):
            r = random.randint(1, height-m)
            rp = random.randint(1, width-m)

        cnt = height + 1
        for i in range(height):
            if (i == r):
                if width == 1:
                    print(i,'--',i + 1)
                    f.write(f'{i} {i + 1}')
                else:
                    for j in range(width):
                        if j == rp:
                            print(i,'--',i + 1)
                            f.write(f'\n{i} {i + 1}')
                        else:
                            print(i,'--',cnt)
                            f.write(f'\n{i} {cnt}')
                            cnt += 1
```

```
            else:
                if width > 1:
                    rr = random.randint(1, width-m+1)
                if rr == 1:
                    print(i,'--',i + 1)
                    f.write(f'\n{i} {i + 1}')
                else:
                    if rr > 1:
                        rrp = random.randint(1, rr-m)
                    for k in range(rr):
                        if (k == rrp):
                            print(i,'--',i + 1)
                            f.write(f'\n{i} {i + 1}')
                        else:
                            print(i,'--',cnt)
                            f.write(f'\n{i} {cnt}')
                            cnt += 1

if __name__ == '__main__':
    print('Input Height: ')
    height = int(input())
    if height < 2:
        print("Height < 2")
        sys.exit()
    print('Input Width: ')
    width = int(input())
    if width < 1:
        print("Width < 1")
        sys.exit()

    create_graph(height, width)
    display_graph()


```