



@ININ1Y.UZ

PYTHONDA GRAPH

Presented by: Alimardon Boqijonov



Dars tartibi

- *Grafik nima?*
- *Graphga oid rasmlar*
- *Pythonda Graph*
- *Graphga oid masalalar*
- *Uy ishi*



Graph nima?

- Data structure graph – bu matematikaviy modeldir, bir nechta obyektlar (vertexlar) va ular orasidagi bog'lanishlarni (edge'lar) ifodalaydigan strukturadir. Graph, bir to'plamning tarkibidagi elementlarning o'zaro aloqalarini ifodalaydi.



Graphning asosiy xususiyatlari quyidagicha:

- 1. Vertexlar (yoki node'lar):** Graphning barcha elementlari. Har bir vertexning o'ziga xususiy nomi (key), qiymati yoki atributlari bo'lishi mumkin.
- 2. Edge'lar (yoki tartiblar):** Vertexlar orasidagi aloqalar, bog'lanishlar. Edge'lar vertexlarni birlashtiradi va ularga aloqalar yaratadi. Edge'lar bilan ifodalangan aloqalar boshqa vertexlarga yo'l olib kelishga imkon beradi.
- 3. Directed va undirected graph:** Directed (yo'nlilangan) graphda edge'lar biror bir yo'nalishni ko'rsatadi, ya'ni aloqaning yo'nalishi bor. Undirected (yo'nalmas) graphda esa edge'lar ikki vertex orasidagi aloqani yo'qotadi, ya'ni aloq bilan bir nechta yo'nalishlarni aks ettiradi.



GRAPHLAR BIR QATOR MASALALARNI YECHISHDA VA HAQIQIY DUNYODAGI BIR NECHTA TIZIMLARNI MODEL QILISHDA KENG QO'LLANILADI. ULAR ALGORITMLARNI QO'LLAB-QUVVATLASH, ALOQALAR TARMOG'INI ANALIZ QILISH VA TIZIMLARNI OPTIMALLASHTIRISHDA KENG QO'LLANILADI. MISOL UCHUN, SOSIAL TARMOQLAR, YO'L TUZISH TIZIMLARI, FILIALLARARO ALOQALAR TARMOQLARI VA BOSHQALAR GRAPHNING MUHIM MISOLLARIDIR



Graphlarni Python'da amalga oshirish uchun turli usullar mavjud. Quyidagi misolda, `networkx` kutubxonasini ishlatib, unda graph yaratish, vertexlarni qo'shish va edge'lar bilan ulardagi aloqalarni ifodalash uchun bir necha amallar ko'rsatilgan:

```
python Copy code  
  
import networkx as nx  
import matplotlib.pyplot as plt  
  
# Graph yaratish  
G = nx.Graph()  
  
# Vertexlarni qo'shish  
G.add_node(1)  
G.add_node(2)  
G.add_node(3)  
G.add_node(4)  
  
# Edge'lar bilan aloqalarni qo'shish  
G.add_edge(1, 2)  
G.add_edge(2, 3)  
G.add_edge(3, 4)  
G.add_edge(4, 1)  
  
# Graphni vizualizatsiya qilish  
nx.draw(G, with_labels=True)  
plt.show()
```



Ushbu kod `networkx` kutubxonasini o'rnatilgan holda graphni yaratadi, vertexlarni qo'shadi va edge'lar bilan ulardagi aloqalarni ifodalaydi. `nx.draw()` funksiyasi orqali graphni vizualizatsiya qiladi.

`networkx` kutubxonasini o'rnatish uchun `pip install networkx` buyrug'i bilan kerakli kutubxona o'rnatilishi kerak.




Python'da Graphlarni amalga oshirish uchun boshqa usullar ham mavjud. Bazi ko'p foydalaniladigan usullar quyidagilardir:

1. Adjacency Matrix (Qo'shilish matritsa): Vertexlar va ularning aloqalarini matritsa shaklida ifodalovchi usul. Matritsada vertexlar indekslar yordamida tartiblangan, va matritsaning elementlari esa aloqalarni ifodalaydi. Bunda, 1 elementi ikki vertexning aloqasini anglatadi, 0 esa aloqa yo'qligini ifodalaydi.
2. Adjacency List (Qo'shilish ro'yxati): Har bir vertexni ro'yxat bilan ifodalaydigan usul. Vertexlar ro'yxatida o'zining aloqalarini saqlab turadi. Bu usulda vertexlarga tegishli bo'lgan aloqalar bilan bog'liq ma'lumotlar ham saqlanishi mumkin.
3. Dictionary of Sets (Himoya ornatilgan lug'at): Pythonning o'zida himoya ornatilgan lug'at strukturasi ishlatilarak graph yaratish. Har bir vertexni kalit so'z (key) sifatida olib, unga aloqalar to'plamini (set) bog'lab boriladi.
4. Adjacency Matrix List (Qo'shilish matritsa ro'yxati): Bu usulda matritsa va ro'yxatlar birlashtirilgan bo'lib, vertexlarni indekslar orqali tartiblangan matritsa bilan aloqalar ro'yxatini saqlash mumkin.

Har bir usulning o'ziga xos xususiyatlari va foydalanish sohalariga ko'ra afzalliklari mavjud. Asosiy mavzularni o'rganishdan so'ng, graphlar bilan ishlash uchun kerakli usullarni tanlash mumkin.

Adjacency Matrix (qo'shilish matritsa) usuli bilan graphni ifodalash uchun Python tilida quyidagi kodni ishlatishingiz mumkin:

python

 Copy code

```
class Graph:
    def __init__(self, num_vertices):
        self.num_vertices = num_vertices
        self.matrix = [[0] * num_vertices for _ in range(num_vertices)]

    def add_edge(self, source, destination):
        if 0 <= source < self.num_vertices and 0 <= destination < self.num_v
            self.matrix[source][destination] = 1
            self.matrix[destination][source] = 1

    def print_graph(self):
        for row in self.matrix:
            print(row)

# Graph obyektini yaratish
graph = Graph(5)


# Aloqalarni qo'shish
graph.add_edge(0, 1)
graph.add_edge(0, 4)
graph.add_edge(1, 2)
graph.add_edge(1, 3)
graph.add_edge(1, 4)
graph.add_edge(2, 3)
graph.add_edge(3, 4)

# Graphni chop etish
graph.print_graph()
```

 Regenerate response

Adjacency List (qo'shilish ro'yxati) usulini Python tilida quyidagi kod orqali amalga oshirishingiz mumkin:

python

 Copy code

```
from collections import defaultdict

class Graph:
    def __init__(self):
        self.adj_list = defaultdict(list)


    def add_edge(self, source, destination):
        self.adj_list[source].append(destination)
        self.adj_list[destination].append(source)

    def print_graph(self):
        for vertex, neighbors in self.adj_list.items():
            print(f"Vertex {vertex}: {neighbors}")

# Graph obyektini yaratish
graph = Graph()


# Aloqalarni qo'shish
graph.add_edge(0, 1)
graph.add_edge(0, 4)
graph.add_edge(1, 2)
graph.add_edge(1, 3)
graph.add_edge(1, 4)
graph.add_edge(2, 3)
graph.add_edge(3, 4)

# Graphni chop etish
graph.print_graph()
```

 Regenerate response

Dictionary of Sets (himoya ornatilgan lug'at) usulini Python tilida quyidagi kod orqali amalga oshirishingiz mumkin:

python

 Copy code

```
class Graph:
    def __init__(self):
        self.adj_dict = {}

    def add_edge(self, source, destination):
        if source in self.adj_dict:
            self.adj_dict[source].add(destination)
        else:
            self.adj_dict[source] = {destination}

        if destination in self.adj_dict:
            self.adj_dict[destination].add(source)
        else:
            self.adj_dict[destination] = {source}

    def print_graph(self):
        for vertex, neighbors in self.adj_dict.items():
            print(f"Vertex {vertex}: {neighbors}")
```

Graph obyektini yaratish

graph = Graph()

Aloqalarni qo'shish

graph.add_edge(0, 1)

graph.add_edge(0, 4)

graph.add_edge(1, 2)

graph.add_edge(1, 3)

graph.add_edge(1, 4)

graph.add_edge(2, 3)

☐ Stop generating

Adjacency Matrix List (qo'shilish matritsa ro'yxati) usulini Python tilida quyidagi kod orqali amalga oshirishingiz mumkin:

python

 Copy code

```
class Graph:
    def __init__(self, num_vertices):
        self.num_vertices = num_vertices
        self.matrix = [[0] * num_vertices for _ in range(num_vertices)]

    def add_edge(self, source, destination):
        if 0 <= source < self.num_vertices and 0 <= destination < self.num_v
            self.matrix[source][destination] = 1
            self.matrix[destination][source] = 1

    def adjacency_list(self):
        adj_list = {}
        for i in range(self.num_vertices):
            neighbors = []
            for j in range(self.num_vertices):
                if self.matrix[i][j] == 1:
                    neighbors.append(j)
            adj_list[i] = neighbors
        return adj_list

    def print_graph(self):
        for vertex, neighbors in self.adjacency_list().items():
            print(f"Vertex {vertex}: {neighbors}")
```

Project Timeline

