Grupo 1: Caminhos Mínimos e Árvores Geradoras

Exercício 1.1 – Algoritmo de Dijkstra (Caminhos Mínimos)

Implemente o algoritmo de Dijkstra para encontrar os caminhos mínimos a partir de um vértice de origem. O grafo é representado por um dicionário onde cada chave é um vértice e seu valor é uma lista de tuplas contendo (vértice vizinho, peso da aresta).

Exemplo:
Dado o grafo:

- A: [(B, 1), (C, 4)]
- B: [(A, 1), (C, 2), (D, 5)]
- C: [(A, 4), (B, 2), (D, 1)]
- D: [(B, 5), (C, 1)]
  E vértice de origem "A", a saída será algo como:

  Distância até A: 0
  Distância até B: 1
  Distância até C: 3
  Distância até D: 4

Exercício 1.2 – Algoritmo de Prim (Árvore Geradora Mínima)

Utilize o algoritmo de Prim para calcular a árvore geradora mínima de um grafo não direcionado e ponderado. O grafo é representado por um dicionário onde cada chave é um vértice e seu valor é uma lista de tuplas (vizinho, peso da aresta).

Exemplo:
Dado o grafo:

- A: [(B, 2), (C, 3)]
- B: [(A, 2), (C, 1), (D, 4)]
- C: [(A, 3), (B, 1), (D, 5)]
- D: [(B, 4), (C, 5)]

  E vértice inicial "A", uma possível árvore geradora mínima (MST) pode ser:
  A - B com peso 2
  B - C com peso 1
  B - D com peso 4

Grupo 2 – Problemas NP-completos com Heurísticas

Exercício 2.1 – Problema da Mochila com Heurística Gulosa

Utilize uma abordagem heurística (gulosa) para resolver o problema da mochila. Ordene os itens pela razão valor/peso e adicione-os à mochila até atingir a capacidade máxima.

Exemplo:

Itens disponíveis:

- item1: peso 2, valor 40
- item2: peso 3, valor 50
- item3: peso 5, valor 100
- item4: peso 4, valor 90
  Capacidade da mochila: 8
  Uma solução possível é selecionar os itens "item1" e "item2" (peso total 5, valor total 90) ou outra combinação que respeite a capacidade. O algoritmo guloso, pela razão valor/peso, selecionará primeiro o item com melhor relação.

Exercício 2.2 – Problema do Caixeiro Viajante com Heurística do Vizinho Mais Próximo

Implemente uma solução aproximada para o TSP usando a heurística do vizinho mais próximo. A partir de uma cidade inicial, escolha repetidamente a cidade mais próxima que ainda não foi visitada.

Exemplo:
Considere as cidades com coordenadas:

- A: (0, 0)
- B: (1, 5)
- C: (5, 2)
- D: (6, 6)
- E: (8, 3)
  Uma rota possível obtida pelo algoritmo pode ser:

A -> B -> D -> E -> C

Grupo 3 – Comunicação Segura com TLS e Manipulação de Pacotes

Exercício 3.1 – Servidor e Cliente TLS (Eco)

Implemente um servidor TLS (eco) e um cliente TLS utilizando o módulo `ssl` do Python. O servidor utiliza um certificado autoassinado para estabelecer uma conexão segura e ecoa os dados enviados pelo cliente.

Exemplo:

Após gerar o certificado autoassinado e iniciar o servidor, execute o cliente que enviará a mensagem "Olá, servidor TLS!".

A saída esperada será:

- No servidor:
  ```
  Conexão estabelecida com ('127.0.0.1', <porta>)
  Recebido: Olá, servidor TLS!
  ```

- No cliente:
  ```
  Cliente: conexão estabelecida
  Cliente: recebido: Olá, servidor TLS!
  ```

Exercício 3.2 – Cliente TLS com Logging de Pacotes

Implemente um cliente TLS que, além de estabelecer uma conexão segura, intercepta (por meio de monkey patch) os dados enviados e recebidos para exibir o conteúdo dos pacotes antes da camada TLS processá-los.

Exemplo:
Ao executar o cliente, a mensagem "Mensagem segura com logging de pacotes" é enviada. Durante a comunicação, serão exibidos logs interceptando os dados enviados e recebidos, como:

```
Interceptado (envio): b'Mensagem segura com logging de pacotes'
Interceptado (recebido): b'Mensagem segura com logging de pacotes'
```

Grupo 4 – Network Scanning e Packet Sniffing

Exercício 4.1 – Varredura de Rede via ARP (Scapy)

Utilize a biblioteca Scapy para realizar uma varredura ARP na rede local. O script envia requisições ARP para um intervalo de IPs e exibe os hosts que responderam, indicando que estão ativos.

Exemplo:
Utilizando o intervalo "192.168.1.0/24", se houver um host ativo com IP 192.168.1.10 e MAC "aa:bb:cc:dd:ee:ff", a saída incluirá:

```
IP: 192.168.1.10, MAC: aa:bb:cc:dd:ee:ff
```

Observação:
Execute o script com privilégios de administrador.

Exercício 4.2 – Detecção de ARP Spoofing

Monitore os pacotes ARP na rede para detectar mudanças suspeitas no mapeamento IP–MAC. Caso o mesmo IP seja associado a MACs diferentes, exiba um alerta indicando possível ARP spoofing.

Exemplo:
Se durante a monitoração o IP 192.168.1.10 for visto associado primeiramente ao MAC "aa:bb:cc:dd:ee:ff" e depois a outro MAC, por exemplo "11:22:33:44:55:66", o script exibirá:

```
Alerta: Possível ARP Spoofing detectado para IP 192.168.1.10! MAC
anterior: aa:bb:cc:dd:ee:ff, MAC atual: 11:22:33:44:55:66
```

Grupo 5: Coleta de Informações de Domínios e Servidores

Exercício 5.1 - Coleta de Registros DNS (simulando DNSRecon)

Utilize a biblioteca `dnspython` para consultar e coletar informações DNS (registros A, MX e NS) de um domínio.

Exemplo:
Para o domínio "example.com", o script exibirá os registros DNS, por exemplo:

```
Registros A:
    93.184.216.34
Registros MX:
    0 .
Registros NS:
    a.iana-servers.net.
    b.iana-servers.net.
```

Observação:
Instale a biblioteca com: `pip install dnspython`.

Exercício 5.2 - Varredura de Portas com Nmap via Subprocess

Utilize o módulo `subprocess` para executar o comando Nmap e identificar serviços (versões dos serviços) em um host alvo. O script executa o Nmap e exibe a saída com as portas abertas e os serviços identificados.

Exemplo:
Ao executar o comando:

```
python group5_exercicio2.py 192.168.1.1
```

O script executará o Nmap para o host 192.168.1.1 e exibirá a varredura com as portas abertas e os serviços detectados.

Observação:
Certifique-se de ter o Nmap instalado em seu sistema.