

INSTITUTO INFNET
ESCOLA SUPERIOR DE TECNOLOGIA
GRADUAÇÃO EM ENGENHARIA DE SOFTWARE



Projeto de Bloco: Ciência da Computação

TP5

Daniel Gomes Lipkin

17 de mar. de 2025

1.1)

```
vert = ['A','B','C','D','E','F']

ares = [
    [('A','B', 3)],
    [('A','B', 3), ('B','C', 7)],
    [('A','B', 3), ('B','C', 7), ('C','D', 2)],
    [('A','B', 3), ('B','C', 7), ('C','D', 2), ('D','E', 5)],
    [('A','B', 3), ('B','C', 7), ('C','D', 2), ('D','E', 5), ('E','F', 1)],
    [('A','B', 3), ('B','C', 7), ('C','D', 2), ('D','E', 5), ('E','F', 1), ('A','C', 4)],
    [('A','B', 3), ('B','C', 7), ('C','D', 2), ('D','E', 5), ('E','F', 1), ('A','C', 4), ('B','D', 6)],
    [('A','B', 3), ('B','C', 7), ('C','D', 2), ('D','E', 5), ('E','F', 1), ('A','C', 4), ('B','D', 6), ('C','E', 9)],
    [('A','B', 3), ('B','C', 7), ('C','D', 2), ('D','E', 5), ('E','F', 1), ('A','C', 4), ('B','D', 6), ('C','E', 9), ('D','F', 8)],
    [('A','B', 3), ('B','C', 7), ('C','D', 2), ('D','E', 5), ('E','F', 1), ('A','C', 4), ('B','D', 6), ('C','E', 9), ('D','F', 8), ('A','D', 10)],
    [('A','B', 3), ('B','C', 7), ('C','D', 2), ('D','E', 5), ('E','F', 1), ('A','C', 4), ('B','D', 6), ('C','E', 9), ('D','F', 8), ('A','D', 10), ('B','E', 12)],
    [('A','B', 3), ('B','C', 7), ('C','D', 2), ('D','E', 5), ('E','F', 1), ('A','C', 4), ('B','D', 6), ('C','E', 9), ('D','F', 8), ('A','D', 10), ('B','E', 12), ('C','F', 11)],
    [('A','B', 3), ('B','C', 7), ('C','D', 2), ('D','E', 5), ('E','F', 1), ('A','C', 4), ('B','D', 6), ('C','E', 9), ('D','F', 8), ('A','D', 10), ('B','E', 12), ('C','F', 11), ('A','E', 13)],
    [('A','B', 3), ('B','C', 7), ('C','D', 2), ('D','E', 5), ('E','F', 1), ('A','C', 4), ('B','D', 6), ('C','E', 9), ('D','F', 8), ('A','D', 10), ('B','E', 12), ('C','F', 11), ('A','E', 13), ('F', 14)]
]
```

A lista total de vertices é A,B,C,D,E, F. Foi executado com quantidades diferentes de vertices e com combinações de arestas diferentes.

```
2 Vertices

1 arestas
[('A','B', 3)]
0.007727599935606122 segundos - MST
[('A','B /3', '3')]
0.008273700019344687 segundos - A -> B

3 Vertices

1 arestas
[('A','B', 3)]
0.007050399901345372 segundos - MST
None
0.008177099982276559 segundos - A -> C

2 arestas
[('A','B', 3), ('B','C', 7)]
0.007492100121453404 segundos - MST
[('A','B /3', 'C /10', '10')]
0.007857799995690584 segundos - A -> C

3 arestas
[('A','B', 3), ('A','C', 4)]
0.007786900037899613 segundos - MST
[('A','C /4', '4')]
0.007334199966862798 segundos - A -> C

4 Vertices

1 arestas
[('A','B', 3)]
0.007362700067460537 segundos - MST
None
0.007137200096622109 segundos - A -> D

2 arestas
```

5 arestas
[(A', 'B', 3), (B', 'C', 7), (C', 'D', 2), (D', 'E', 5), (E', 'F', 1)]
0.01582830003462732 segundos - MST
[(A', 'B' /3', 'C' /10', 'D' /12', 'E' /17', 'F' /18'], '18')
0.008046000031754375 segundos - A -> F

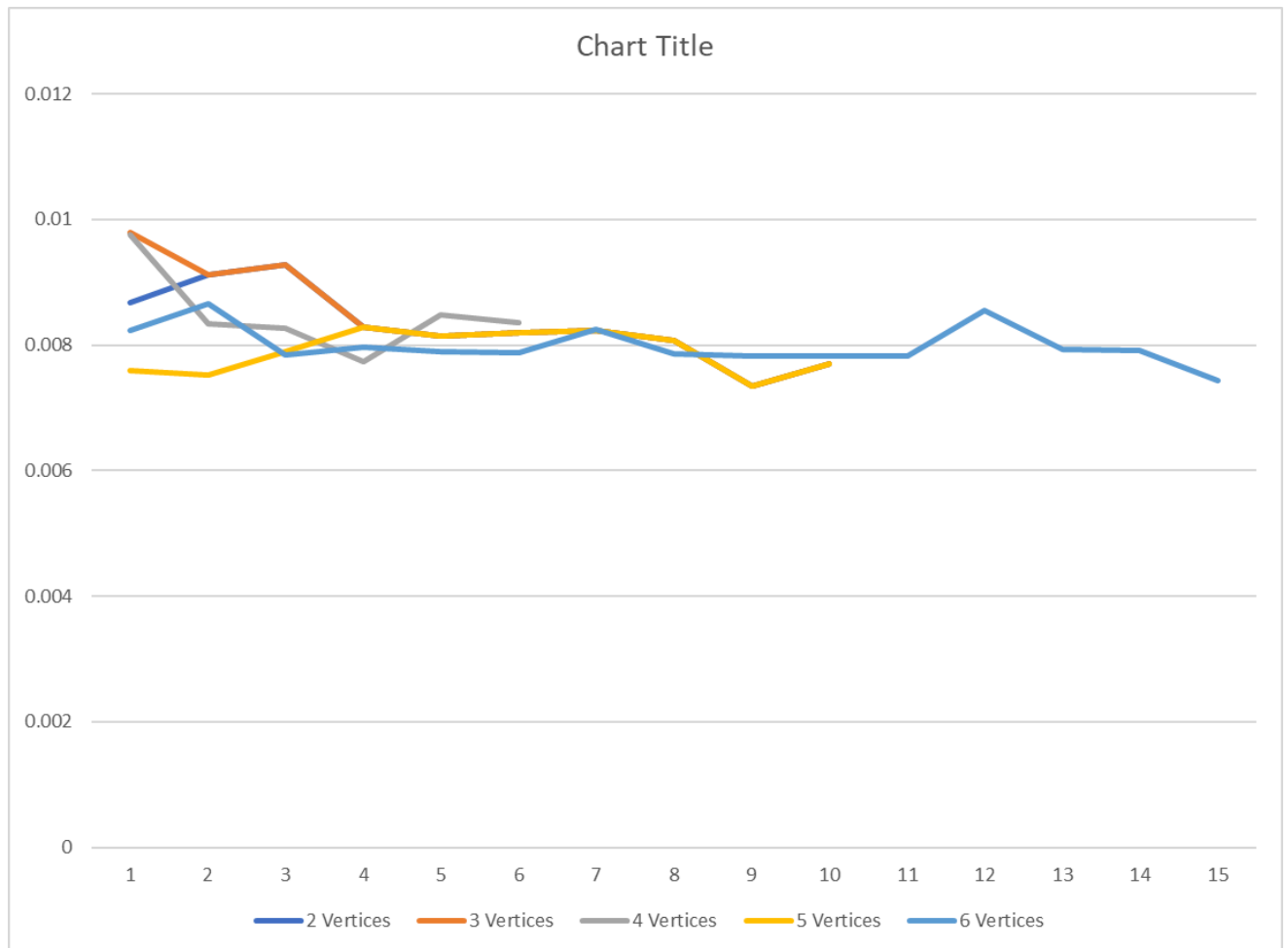
6 arestas
[(A', 'B', 3), (A', 'C', 4), (C', 'D', 2), (D', 'E', 5), (E', 'F', 1)]
0.00811930000782013 segundos - MST
[(A', 'C' /4', 'D' /6', 'E' /11', 'F' /12'], '12')
0.009206100134178996 segundos - A -> F

7 arestas
[(A', 'B', 3), (A', 'C', 4), (C', 'D', 2), (D', 'E', 5), (E', 'F', 1)]
0.009268000023439527 segundos - MST
[(A', 'C' /4', 'D' /6', 'E' /11', 'F' /12'], '12')
0.008554300060495734 segundos - A -> F

8 arestas
[(A', 'B', 3), (A', 'C', 4), (C', 'D', 2), (D', 'E', 5), (E', 'F', 1)]
0.009095000103116035 segundos - MST
[(A', 'C' /4', 'D' /6', 'E' /11', 'F' /12'], '12')
0.008329400094226003 segundos - A -> F

[(vertice1, vertice2, peso)] - MST

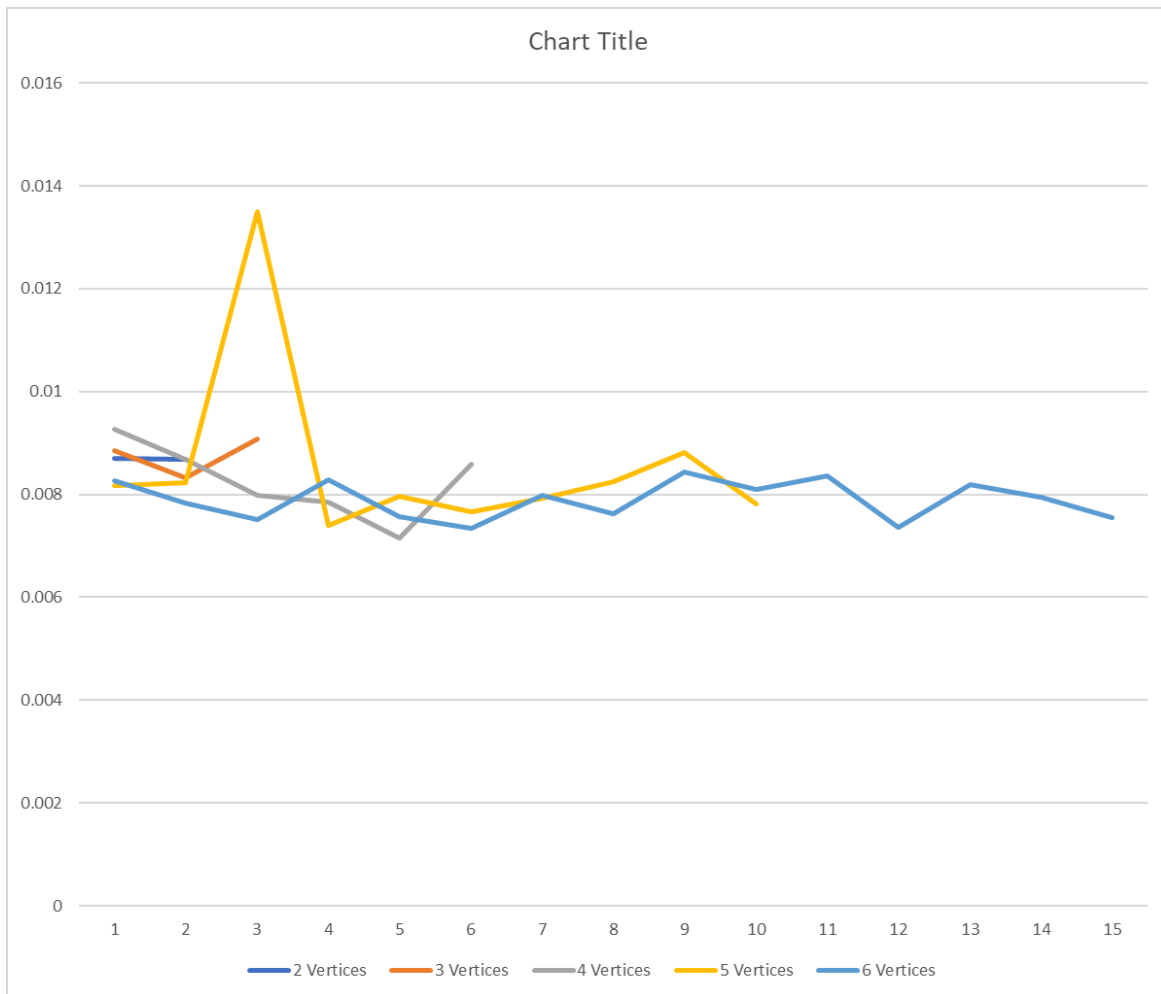
[vertice origem, vertice destino, peso acumulado] - Dijkstra



Tempo x Arestas. Parece que com poucos vertices e poucas arestas, o algoritmo demora mais, provavelmente devido a inicialização. Depois disso é quase impossível medir o tempo do algoritmo pois varia demais cada linha de aresta e também se trata de um problema NP, ou seja, de tempo não polinomial. A única maneira possível de se medir o tempo seria fazendo a versão força bruta do problema por ser NP-Completo, porém estaria na ala de complexidade $O(2^V)$ ou até $O(V!)$.

Pelo menos é possível saber que a complexidade de espaço será $O(V)$ pois armazena todos os vertices pelo menos 1 vez no algoritmo para demarcar visitados.

1.2)



Usando as mesmas entradas do exercício anterior, vemos o mesmo problema. Complexidade de espaço $O(V)$ para armazenar os menores pesos dos vértices e para montar a MST.

2.1)

```
items = {
  "item1":(2,40),
  "item2":(3,50),
  "item3":(5,100),
  "item4":(4,90),
  "item5":(3, 40),
  "item6":(2,50)
}
```

```

Capacidade 2
3 items
(['item1'], 2, 40)
4 items
(['item1'], 2, 40)
5 items
(['item1'], 2, 40)
6 items
(['item1'], 2, 40)

Capacidade 4
3 items
(['item1'], 2, 40)
4 items
(['item1'], 2, 40)
5 items
(['item4'], 4, 90)
6 items
(['item4'], 4, 90)

Capacidade 6
3 items
(['item1', 'item2'], 5, 90)
4 items
(['item1', 'item2'], 5, 90)
5 items
(['item4', 'item1'], 6, 130)
6 items
(['item4', 'item1'], 6, 130)

```

```

Capacidade 8
3 items
(['item1', 'item2'], 5, 90)
4 items
(['item1', 'item3'], 7, 140)
5 items
(['item4', 'item1'], 6, 130)
6 items
(['item4', 'item1'], 6, 130)

Capacidade 10
3 items
(['item1', 'item2'], 5, 90)
4 items
(['item1', 'item3', 'item2'], 10, 190)
5 items
(['item4', 'item1', 'item2'], 9, 180)
6 items
(['item4', 'item1', 'item2'], 9, 180)

```

(itens, peso total, valor)

Essa heurística tem o problema de nunca realmente encher a capacidade total da mochila. Se fosse possível quebrar os itens em partes menores, o knapsack fracionário poderia ser aplicado para colocar valores fracionados dos itens, aproximando-se a uma solução mais ótima do problema. Complexidade de espaço $O(n)$ pois armazena

separadamente os itens ordenados.

2.2)

```
vertices = {  
  "A": (0, 0),  
  "B": (1, 5),  
  "C": (5, 2),  
  "D": (6, 6),  
  "E": (8, 3),  
  "F": (2, 4)  
}
```

2 vertices

A -> ([A, 'B', 'A'], 10.198039027185569)

B -> ([B, 'A', 'B'], 10.198039027185569)

3 vertices

A -> ([A, 'B', 'C', 'A'], 15.484184320727287)

B -> ([B, 'C', 'A', 'B'], 15.48418432072729)

C -> ([C, 'B', 'A', 'C'], 15.484184320727287)

4 vertices

A -> ([A, 'B', 'C', 'D', 'A'], 22.707406513449016)

B -> ([B, 'C', 'D', 'A', 'B'], 22.707406513449016)

C -> ([C, 'D', 'B', 'A', 'C'], 19.70630945993773)

D -> ([D, 'C', 'B', 'A', 'D'], 22.707406513449016)

5 vertices

A -> ([A, 'B', 'C', 'E', 'D', 'A'], 25.35212982346372)

B -> ([B, 'C', 'E', 'D', 'A', 'B'], 25.352129823463727)

C -> ([C, 'E', 'D', 'B', 'A', 'C'], 22.351032769952443)

D -> ([D, 'E', 'C', 'B', 'A', 'D'], 25.35212982346372)

E -> ([E, 'C', 'D', 'B', 'A', 'E'], 26.027426058289137)

6 vertices

A -> ([A, 'F', 'B', 'C', 'E', 'D', 'A'], 26.139459827243613)

B -> ([B, 'F', 'C', 'E', 'D', 'A', 'B'], 25.371894661300807)

C -> ([C, 'E', 'D', 'F', 'B', 'A', 'C'], 23.138362773732336)

D -> ([D, 'E', 'C', 'F', 'B', 'A', 'D'], 25.371894661300807)

E -> ([E, 'C', 'F', 'B', 'A', 'D', 'E'], 25.371894661300807)

F -> ([F, 'B', 'C', 'E', 'D', 'A', 'F'], 26.139459827243613)

(caminho, peso total)

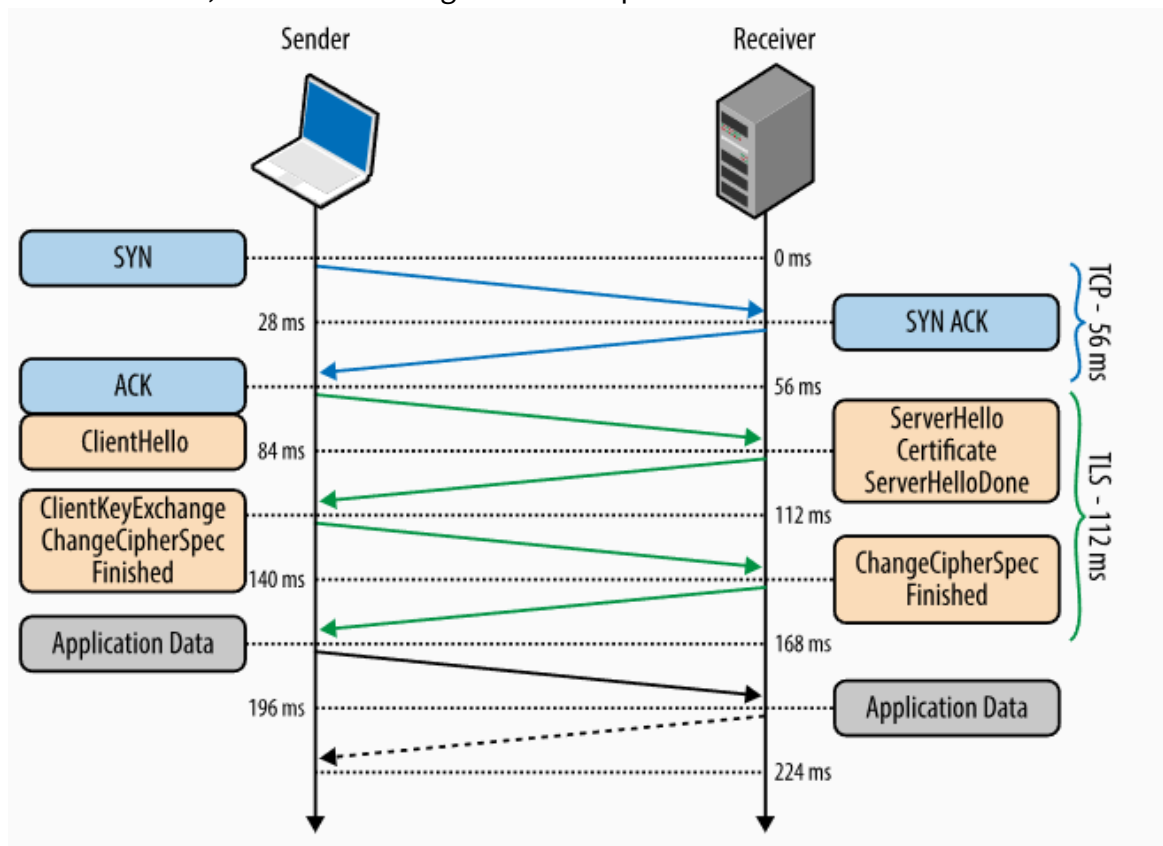
O(V) complexidade de espaço para armazenar as menores distâncias. De qualquer maneira o tempo aparenta ser logarítmico pois aumenta drasticamente com baixos números de vértices e normaliza quando chega a um certo ponto como pode ser visto começando do mapa de 4 vértices.

3.1)

```
>>> sslClient("localhost", 8000) >>> sslHost("localhost", 8000)
Conectado localhost:8000          Iniciando localhost:8000
Servidor: Eae cria!              ('127.0.0.1', 62762) conectado
                                  ('127.0.0.1', 62762): oi
                                  Servidor: oie
                                  Servidor: oie
Voc : oi
Servidor: oie
Voc :
```

Iniciados atrav s de : `python -m server_cienciacomp_tp5` e `python -m client_cienciacomp_tp5`

O protocolo SSL/TLS adiciona uma camada de seguran a a comunica  o TCP do aplicativo antes de estabelecer a conex  o e enquanto mandam dados entre eles. Primeiro   um processo de handshake entre o cliente e o servidor, onde o cliente valida o certificado (server.pem nesse caso) e recebe a chave publica, ambos concordam usar uma cipher suite (RSA, AES, etc) para criptografar mensagens e geram chaves de no n vel de Sess o, trocando mensagens de teste para validar a conex  o.



<https://hpbn.co/transport-layer-security-tls/>

4.1)

192.168.0.1/24


```

Rede: 192.168.0.1/24
Begin emission
.....
.....*.....
Finished sending 256 packets
.....*.....
.....
.....
.....
.....
.....
.....*.....
.....
Received 1550 packets, got 3 answers, remaining 253 packets
3 hosts
192.168.0.49|40:ae:30:64:d9:fa - Ativo
192.168.0.1|98:77:e7:0c:2e:18 - Ativo
192.168.0.55|14:7d:da:d1:6a:e7 - Ativo

```

192.168.1.64/20

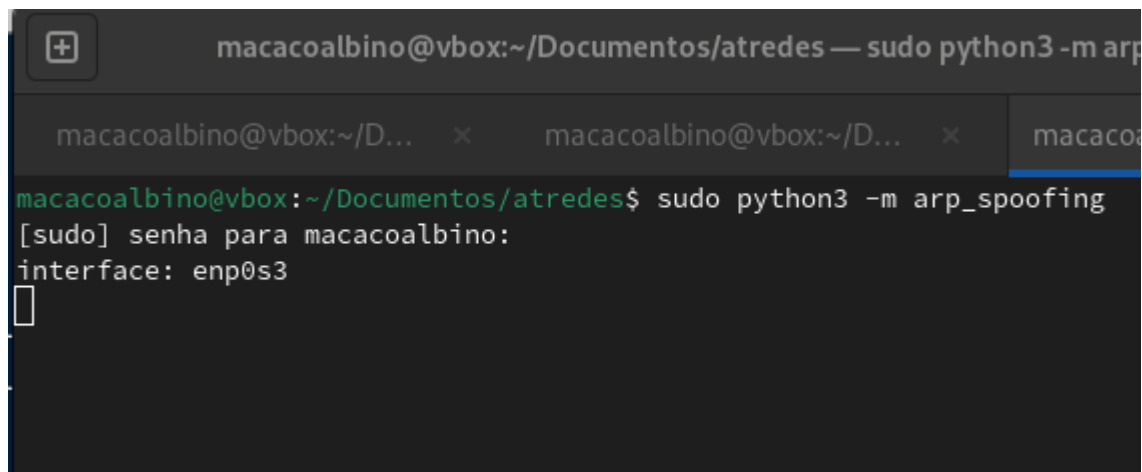
```

Finished sending 4096 packets
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
Received 3838 packets, got 4 answers, remaining 4092 packets
4 hosts
192.168.0.49|40:ae:30:64:d9:fa - Ativo
192.168.0.1|98:77:e7:0c:2e:18 - Ativo
192.168.0.55|14:7d:da:d1:6a:e7 - Ativo
192.168.0.13|98:06:3c:f5:29:28 - Ativo

```

O programa manda uma mensagem SYN para todos os endereços da tabela ARP ligadas ao range de endereços, marcando ativo ou inativo para aqueles que respondem ou não.

4.2)



```
macacoalbino@vbox:~/Documentos/atredes — sudo python3 -m arp
macacoalbino@vbox:~/D... x macacoalbino@vbox:~/D... x macaco
macacoalbino@vbox:~/Documentos/atredes$ sudo python3 -m arp_spoofing
[sudo] senha para macacoalbino:
interface: enp0s3
█
```

É feito um sniff sem filtro de uma interface escolhida, se o pacote tiver a camada ARP e for um ARP do tipo resposta (op 2), ele manda um pacote com o destino do IP do emissor e compara o MAC do pacote analisado com o MAC do pacote novo. Se os endereços não coincidem, quer dizer que alguém está tentando envenenar a tabela ARP para forjar mensagens com o disfarce de um IP confiável, aka, ARP spoofing. É meio complicado simular ARP spoofing em mim mesmo já que meu IP naturalmente corresponde ao meu endereço físico (MAC), como se eu fosse a vítima e o man-in-the-middle ao mesmo tempo. E isso sem mencionar as questões éticas e legais de executar algo assim em dispositivos que não pertencem ao desenvolvedor.

5.1)

```

macacoalbino@vbox:~/Documentos/atredes$ python3 -m dns_resolver
Escreva o dominio que quer analisar: google.com

Record A
142.251.135.110

Record AAAA
2800:3f0:4004:806::200e

Record NS
ns4.google.com.
ns1.google.com.
ns2.google.com.
ns3.google.com.

Record SOA
ns1.google.com. dns-admin.google.com. 740276574 900 900 1800 60

Record MX
10 smtp.google.com.

Record TXT
"apple-domain-verification=30afIBcvSuDV2PLX"
"MS=E4A68B9AB2BB9670BCE15412F62916164C0B20BB"
"v=spf1 include:_spf.google.com ~all"
"docusign=1b0a6754-49b1-4db5-8540-d2c12664b289"
"onetrust-domain-verification=de01ed21f2fa4d8781cbc3fffb89cf4ef"
"cisco-ci-domain-verification=479146de172eb01ddee38b1a455ab9e8bb51542ddd7f1fa298557dfa7b22d963"
"facebook-domain-verification=22rm551cu4k0ab0bxsw536tlds4h95"
"google-site-verification=TV9-DBe4R80X4v0M4U_bd_J9cp0JM0nikft0jAgjmsQ"
"google-site-verification=4ibFUGB-wXLQ_S7vsXVomSTVamu0XBivAazpR5IZ87D0"
"google-site-verification=wD8N7i1JTNTkezJ49swvWW48f8_9xveREV4oB-0Hf5o"
"docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e"
"globalsign-smime-dv=CDYX+XFHUw2wml6/Gb8+59BsH31KzUr6c1l2BPvqKX8="
The DNS response does not contain an answer to the question: google.com. IN CNAME
The DNS response does not contain an answer to the question: google.com. IN PTR

```

O record A corresponde ao IPV4 do dominio, o AAAA o IPV6, o SOA os dados do servidor DNS responsável pelo host, o MX o servidor de email e TXT informações de texto extras. CNAME não foi achado mas mapeia um nome de domínio com outro nome.

5.2)

```
macacoalbino@vbox:~/Documentos/atredes$ sudo python3 -m nmap_vulner
Porta inicial: 512
Porta final: 1024
Digite o host: localhost
Starting Nmap 7.92 ( https://nmap.org ) at 2025-03-26 21:50 -03
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000030s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 512 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
631/tcp   open  ipp      CUPS 2.4
|_http-server-header: CUPS/2.4 IPP/2.1

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.26 seconds
```

Um subprocesso do Nmap é criado com o script vulners, imprimindo o resultado. Aqui vemos que 512 portas estão fechadas e a porta 631/tcp está aberta.