INSTITUTO INFNET

ESCOLA SUPERIOR DE TECNOLOGIA

GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

Projeto de Bloco: Ciência da Computação

TP4

Daniel Gomes Lipkin

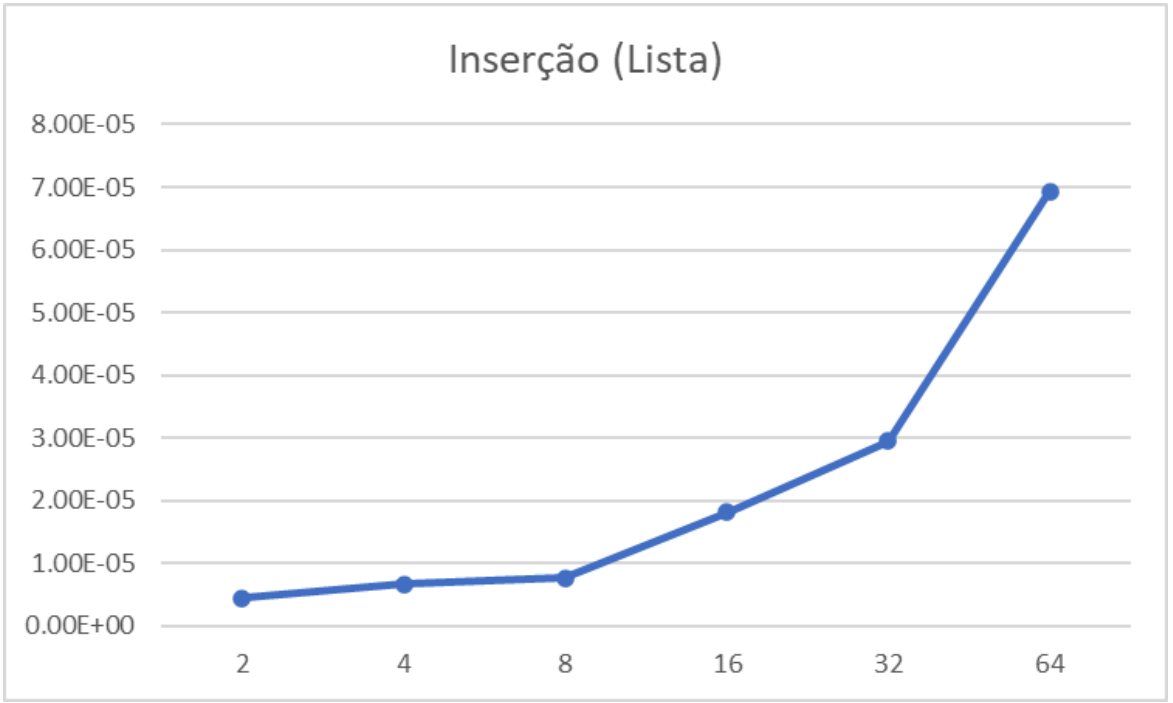
17 de mar. de 2025

Grupo 1

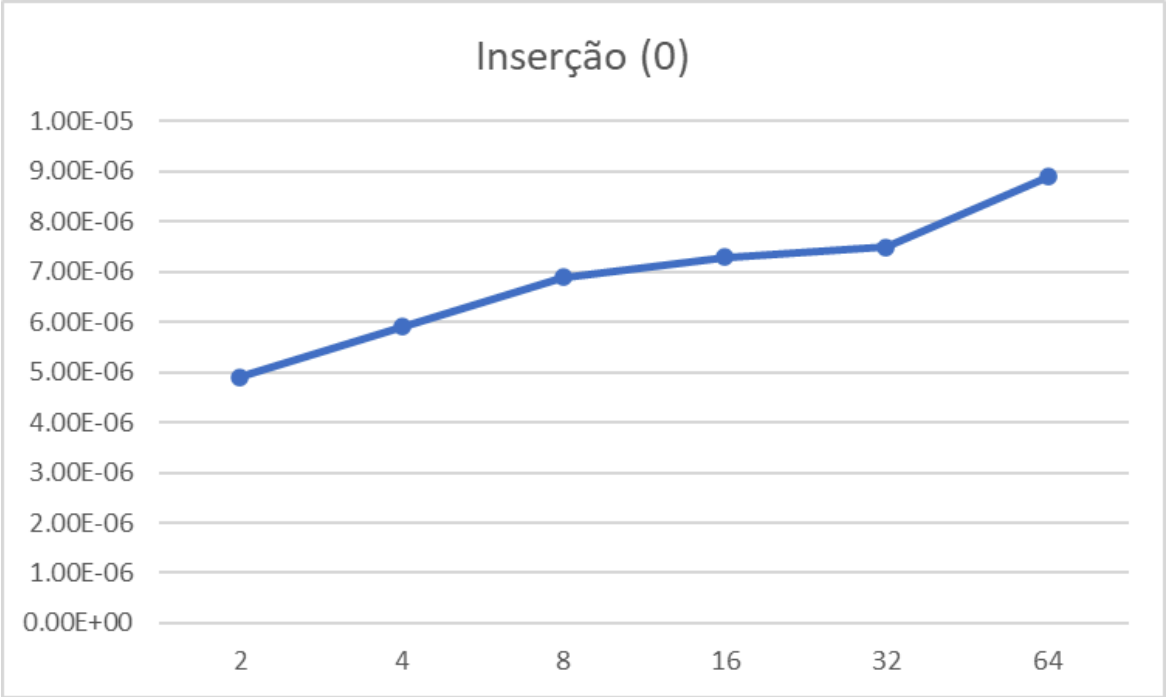
```
 2 elementos
4.500150680541992e-06  Segundos - Inserindo lista remexida
[1, 2]
4.90015372633934e-06  Segundos - Inserindo 0
[0, 2, 1]
True
0.010448200162500143  Segundos - Procurando 1
8.00006091594696e-06  Segundos - Apagando 1
[0, 2]

 4 elementos
6.6999346017837524e-06  Segundos - Inserindo lista remexida
[1, 2, 4, 3]
5.899928510189056e-06  Segundos - Inserindo 0
[0, 1, 4, 3, 2]
True
0.01037549995817244  Segundos - Procurando 2
7.0999376475811005e-06  Segundos - Apagando 2
[0, 1, 4, 3]

 8 elementos
7.699942216277122e-06  Segundos - Inserindo lista remexida
[1, 2, 6, 4, 3, 8, 7, 5]
6.8999361246824265e-06  Segundos - Inserindo 0
[0, 1, 6, 2, 3, 8, 7, 5, 4]
True
0.009145299904048443  Segundos - Procurando 4
7.899943739175797e-06  Segundos - Apagando 4
[0, 1, 6, 2, 3, 8, 7, 5]

 16 elementos
1.810002140700817e-05  Segundos - Inserindo lista remexida
[1, 2, 5, 7, 3, 8, 6, 15, 11, 12, 4, 13, 10, 14, 9, 16]
7.2999391704797745e-06  Segundos - Inserindo 0
[0, 1, 5, 2, 3, 8, 6, 7, 11, 12, 4, 13, 10, 14, 9, 16, 15]
True
0.009329000022262335  Segundos - Procurando 8
9.399838745594025e-06  Segundos - Apagando 8
[0, 1, 5, 2, 3, 10, 6, 7, 11, 12, 4, 13, 15, 14, 9, 16]
```

```
 32 elementos
2.950010821223259e-05  Segundos - Inserindo lista remexida
[1, 2, 6, 4, 3, 7, 10, 11, 5, 8, 9, 15, 12, 18, 16, 14, 31, 17, 21, 29, 30, 19, 1
7.500173524022102e-06  Segundos - Inserindo 0
[0, 1, 6, 2, 3, 7, 10, 4, 5, 8, 9, 15, 12, 18, 16, 11, 31, 17, 21, 29, 30, 19, 13,
True
0.008399599930271506  Segundos - Procurando 16
8.09994526207447e-06  Segundos - Apagando 16
[0, 1, 6, 2, 3, 7, 10, 4, 5, 8, 9, 15, 12, 18, 14, 11, 31, 17, 21, 29, 30, 19, 13,

 64 elementos
6.929994560778141e-05  Segundos - Inserindo lista remexida
[1, 3, 2, 11, 6, 8, 4, 14, 19, 10, 7, 9, 21, 18, 5, 27, 26, 28, 20, 41, 30, 16, 15
, 55, 60, 63, 25, 24, 36, 49, 32, 13, 62]
8.899951353669167e-06  Segundos - Inserindo 0
[0, 1, 2, 3, 6, 8, 4, 11, 19, 10, 7, 9, 21, 18, 5, 14, 26, 28, 20, 41, 30, 16, 15,
55, 60, 63, 25, 24, 36, 49, 32, 13, 62, 58]
True
0.009493099991232157  Segundos - Procurando 32
1.309998333454132e-05  Segundos - Apagando 32
[0, 1, 2, 3, 6, 8, 4, 11, 19, 10, 7, 9, 21, 18, 5, 14, 26, 28, 20, 41, 30, 16, 15,
55, 60, 63, 25, 24, 36, 49, 58, 13, 62]
```

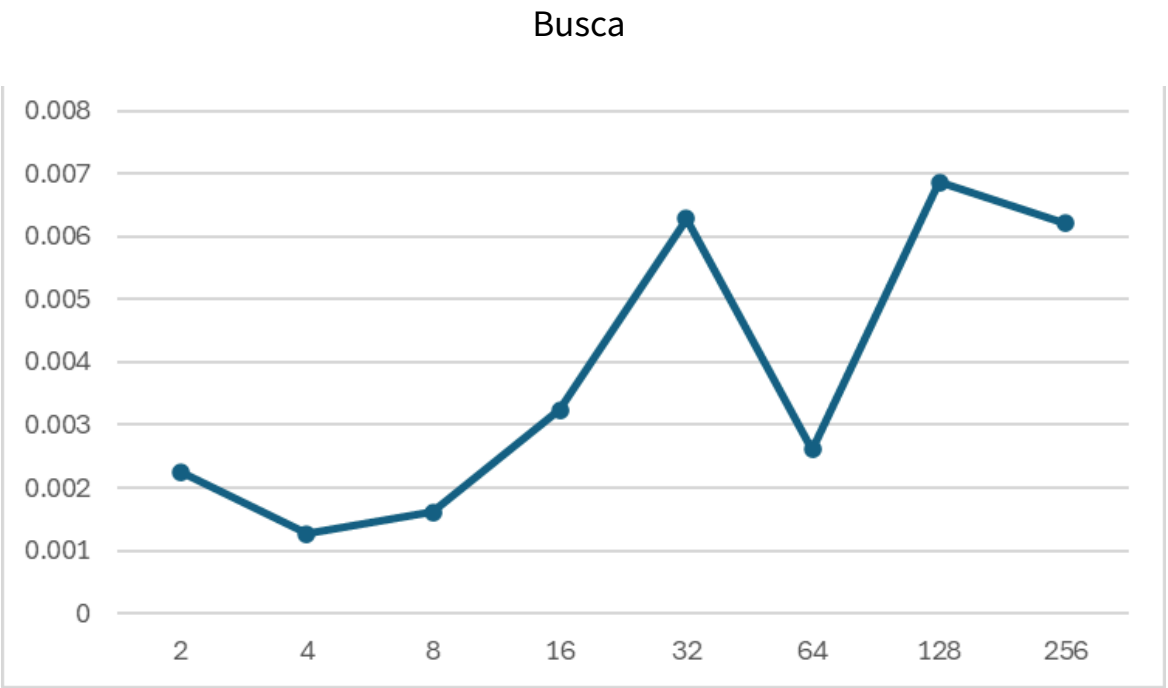
Uma lista de ranges remexida com random.shuffle é inserida. Para as operações de procurar e apagar, o valor é n/2.



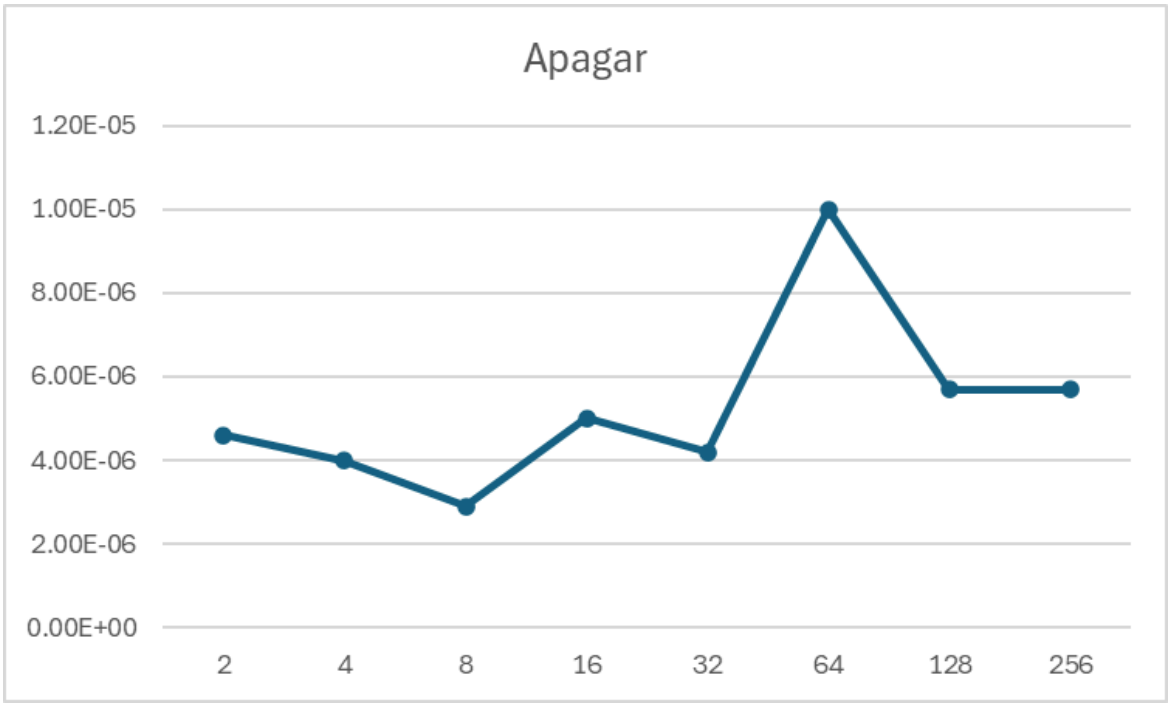
O processo de inserção usa o metodo “bubble up”, iniciando o processo no ultimo indice e comparando recursivamente seu peso com o peso do pai, trocando posições caso seja menor para manter a regra da min-heap onde valores menores ficam em niveis mais altos.
Parece que o tempo corresponde a O(n^2) pois percorre a lista de elementos e, para cada elemento, percorre a arvore linearmente (ou seja, a heap em forma de lista) de forma a reorganizar in-place os índices e inserir o novo elemento.



Vemos o tempo linear de inserção para um elemento individual.



O tempo de buscar parece ser O(log n) com o aumento inicial alto e a regularizada ao chegar no fim da linha. Isso se da conta pois ele percorre a lista como se fosse uma BST, pulando para o filho do índice presente.



Um pouco incompreensível mas o começo alto com a regulação no final indica o mesmo tempo que o de procura, portanto O(log n)
O processo de apagar procura o valor na heap e apaga a node, reajustando a Heap com a função de "bubble down" (minHeapify), que inicia do indice apagado e compara recursivamente o peso com seus filhos para a manter a regra da min-heap com substiuições.

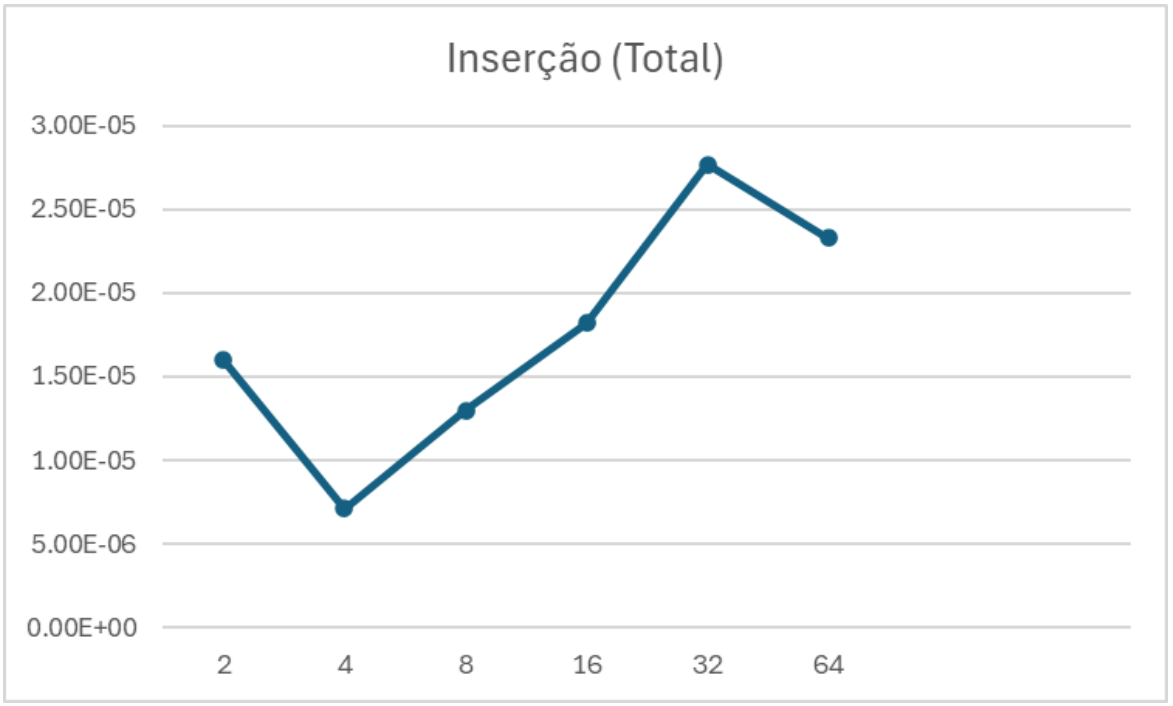
Grupo 2

```
for i in range(1, 6):
    i_n = 2**i
    print("\n\n", i_n, " palavras")

    words = [
        "casa", "casamento", "casulo", "cachorro", "casaco", "casinha", "casbah", "casca",
        "cadeira", "caderno", "cadeado", "café", "caixa", "caju", "cama", "caminho",
        "camisa", "caneta", "caneca", "canção", "canguru", "cantar", "capaz", "capitão",
        "carro", "cartão", "casa", "castelo", "cavalo", "caverna", "cabelo", "cabide",
        "cacto", "cadáver", "cafofo", "caipira", "calça", "caldo", "calma", "cama"
    ][:i_n]
```

```
words_n = ["casa", "calma", "casulo", "caipira", "caipiras", "casamento", "casamentos"]
```

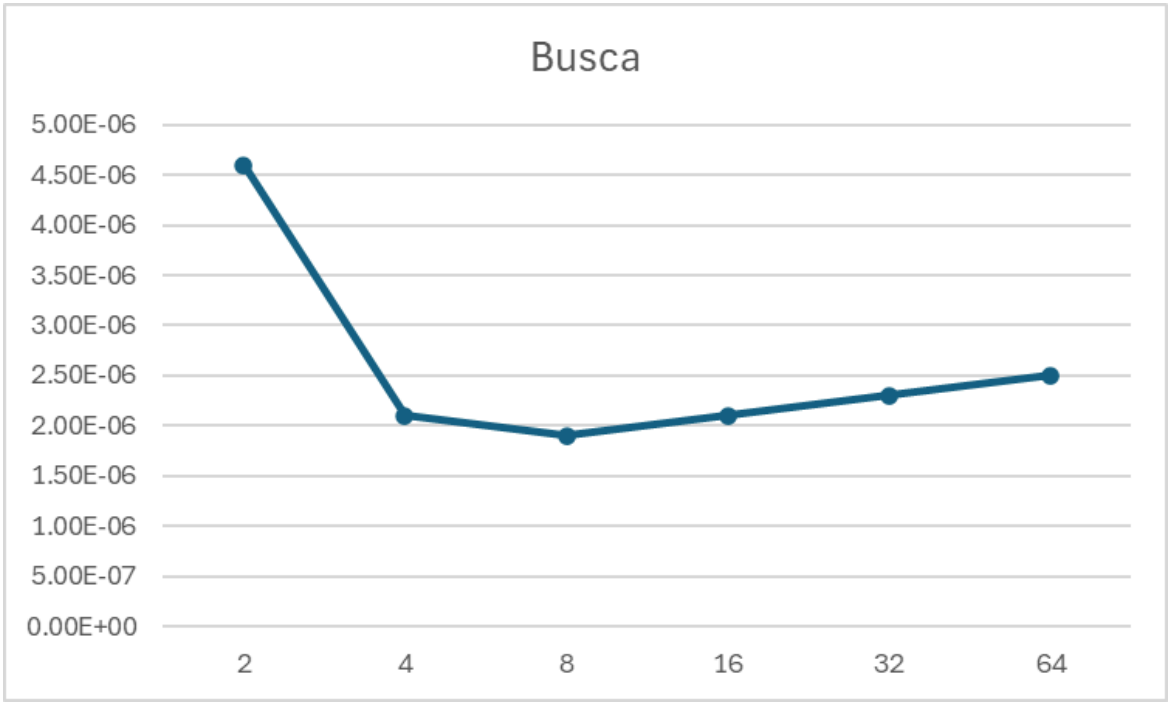
```
 2  palavras
1.4999997802078724e-05 Segundos - Inserindo (total)

 3.100023604929447e-06 Segundos - Buscando casa Achada?: True

 2.500019036233425e-06 Segundos - Inserindo casa com 4 letras

['casa', 'casamento']
 0.0033171999966725707 Segundos - Autocomplete para c

['casa', 'casamento']
 0.0028710999758914113 Segundos - Autocomplete para ca

['casa', 'casamento']
 0.0045334999449551105 Segundos - Autocomplete para cas


 1.479999627918005e-05 Segundos - Apagando casa


 4  palavras
7.1999384090304375e-06 Segundos - Inserindo (total)

 3.00002284348011e-06 Segundos - Buscando calma Achada?: False

 3.400025889277458e-06 Segundos - Inserindo calma com 5 letras

['casa', 'casamento', 'casulo', 'cachorro', 'calma']
 0.002918099984526634 Segundos - Autocomplete para c

['casa', 'casamento', 'casulo', 'cachorro', 'calma']
 0.0036820999812334776 Segundos - Autocomplete para ca

['casa', 'casamento', 'casulo']
 0.0031607000855728984 Segundos - Autocomplete para cas
```

```
 16  palavras
1.850002445280552e-05 Segundos - Inserindo (total)

 2.300017513334751e-06 Segundos - Buscando caipira Achada?: False

 3.700028173625469e-06 Segundos - Inserindo caipira com 7 letras

['casa', 'casamento', 'casaco', 'casulo', 'casinha', 'casbah', 'casca', 'cachor
o', 'cadeira', 'caderno', 'cadeado', 'café', 'caixa', 'caipira', 'caju', 'cama'
 'caminho']
 0.007415300002321601 Segundos - Autocomplete para c

['casa', 'casamento', 'casaco', 'casulo', 'casinha', 'casbah', 'casca', 'cachor
o', 'cadeira', 'caderno', 'cadeado', 'café', 'caixa', 'caipira', 'caju', 'cama'
 'caminho']
 0.006914100027643144 Segundos - Autocomplete para ca

['casa', 'casamento', 'casaco', 'casulo', 'casinha', 'casbah', 'casca']
 0.0032417000038549304 Segundos - Autocomplete para cas


 1.0199961252510548e-05 Segundos - Apagando casinha


 32  palavras
0.00015470001380890608 Segundos - Inserindo (total)

 2.800021320581436e-06 Segundos - Buscando caipiras Achada?: False

 4.699919372797012e-06 Segundos - Inserindo caipiras com 8 letras

['casa', 'casamento', 'casaco', 'casulo', 'casinha', 'casbah', 'casca', 'castel
', 'cachorro', 'cadeira', 'caderno', 'cadeado', 'café', 'caixa', 'caipiras', 'c
ju', 'cama', 'caminho', 'camisa', 'caneta', 'caneca', 'canção', 'canguru', 'car
ar', 'capaz', 'capitão', 'carro', 'cartão', 'cavalo', 'caverna', 'cabelo', 'cab
de']
 0.009992999956011772 Segundos - Autocomplete para c
```

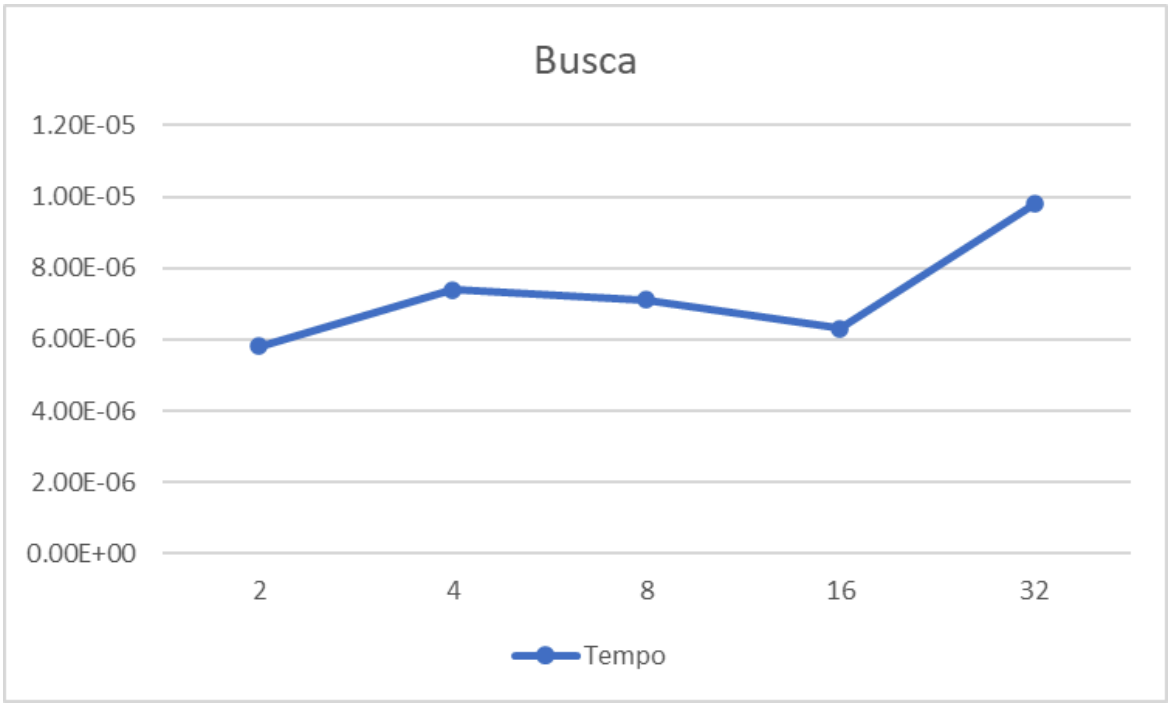


Vemos que o tempo aumenta linearmente ao colocar mais palavras.

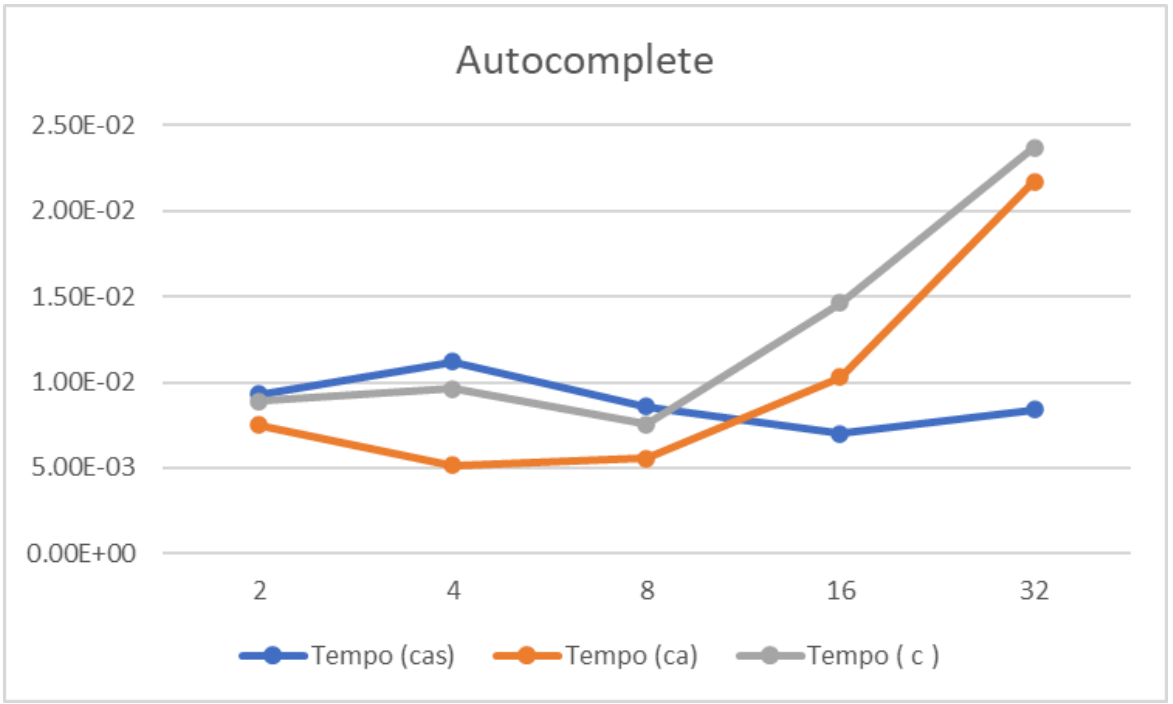
Tempo x Letras. Parece que com menos elementos na Trie e poucas letras, o tempo é drasticamente mais alto que quando tem mais elementos e mais letras.



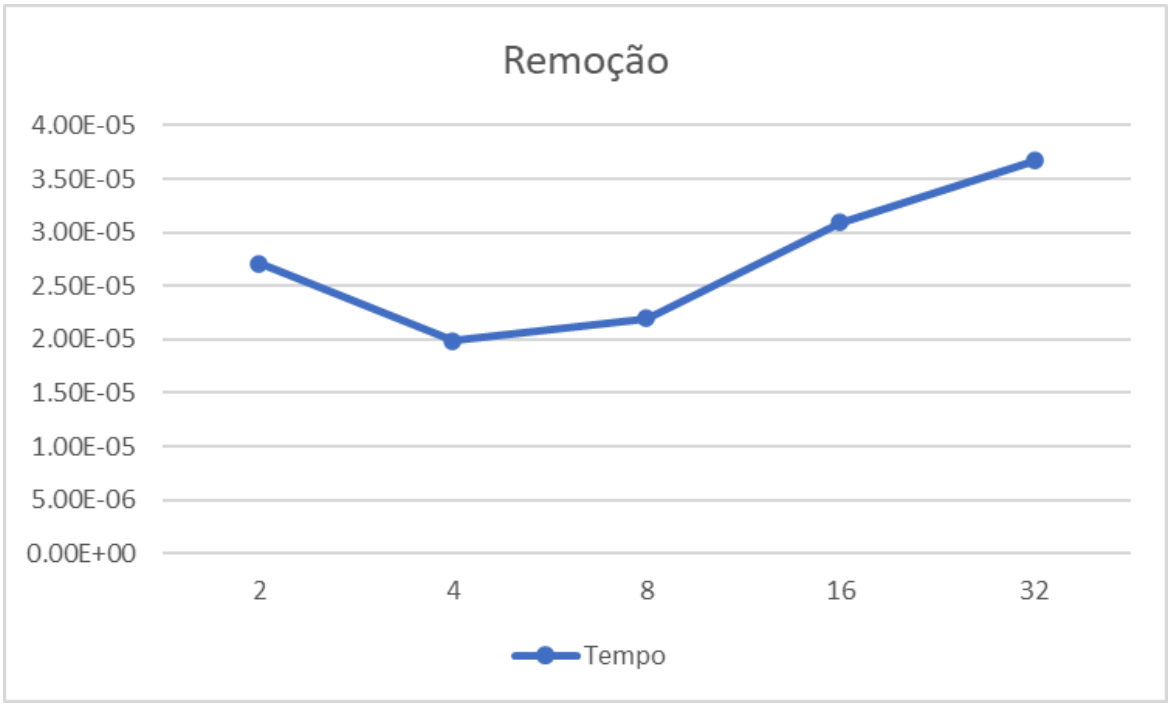
Para cada numero de palavras, a palavra a buscar possui uma letra a mais como feito no exemplo de inserção individual. O tempo parece aumentar linearmente com o tamanho da palavra buscada e o numero de elementos.



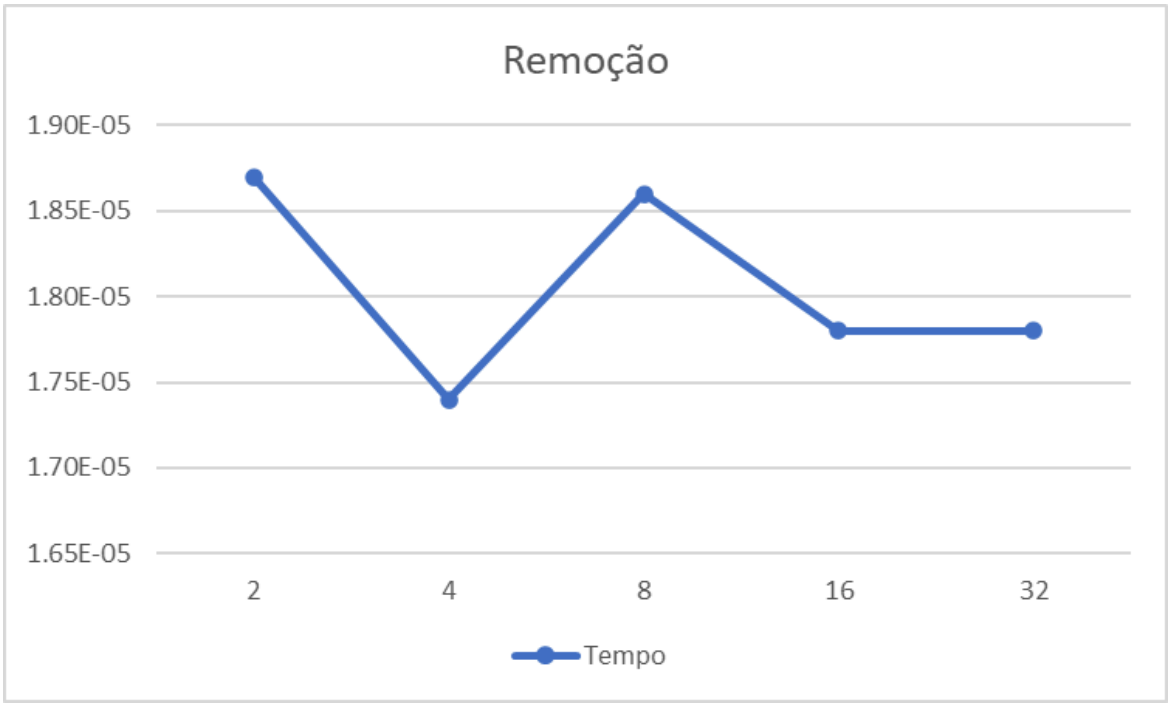
Sem alterar a palavra buscada, somente o numero de palavras. Tem uma aparência ligeiramente logarítmica.



O tempo parece diminuir quanto maior é a palavra e aumentar quanto mais palavras tiver a Trie.



Parece ser similar ao tempo de busca, porem com mais fidelidade. É possivel observar um tempo linear.



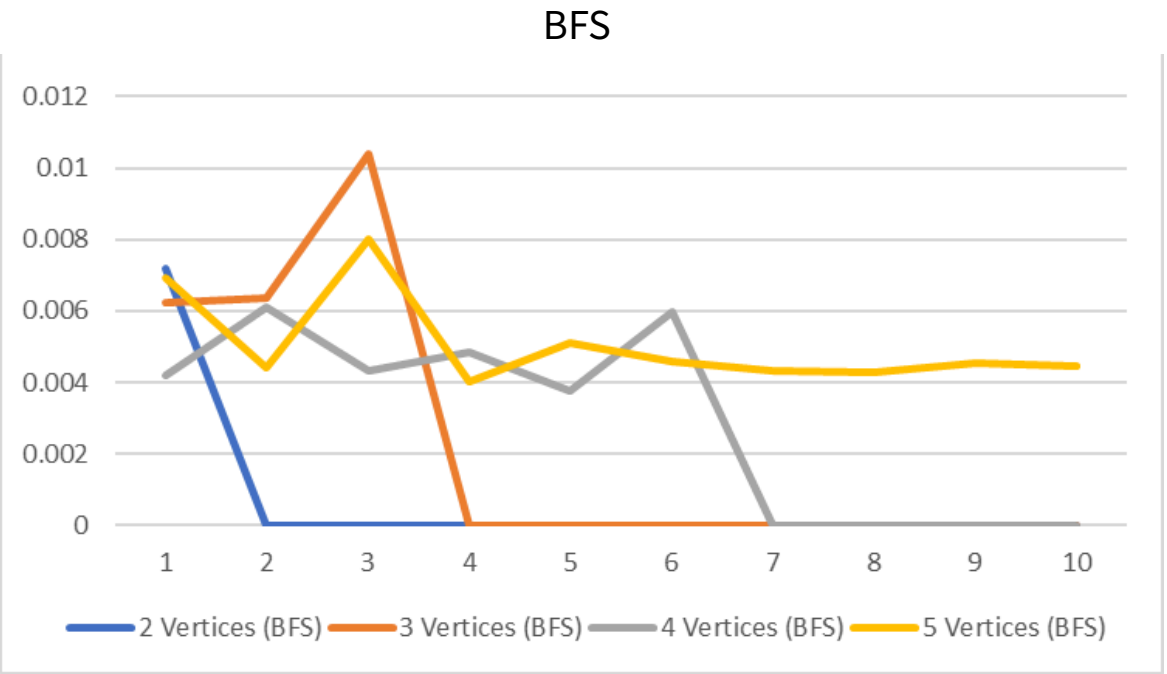
Sem alterar a palavra buscada, somente o numero de palavras.

Grupo 3

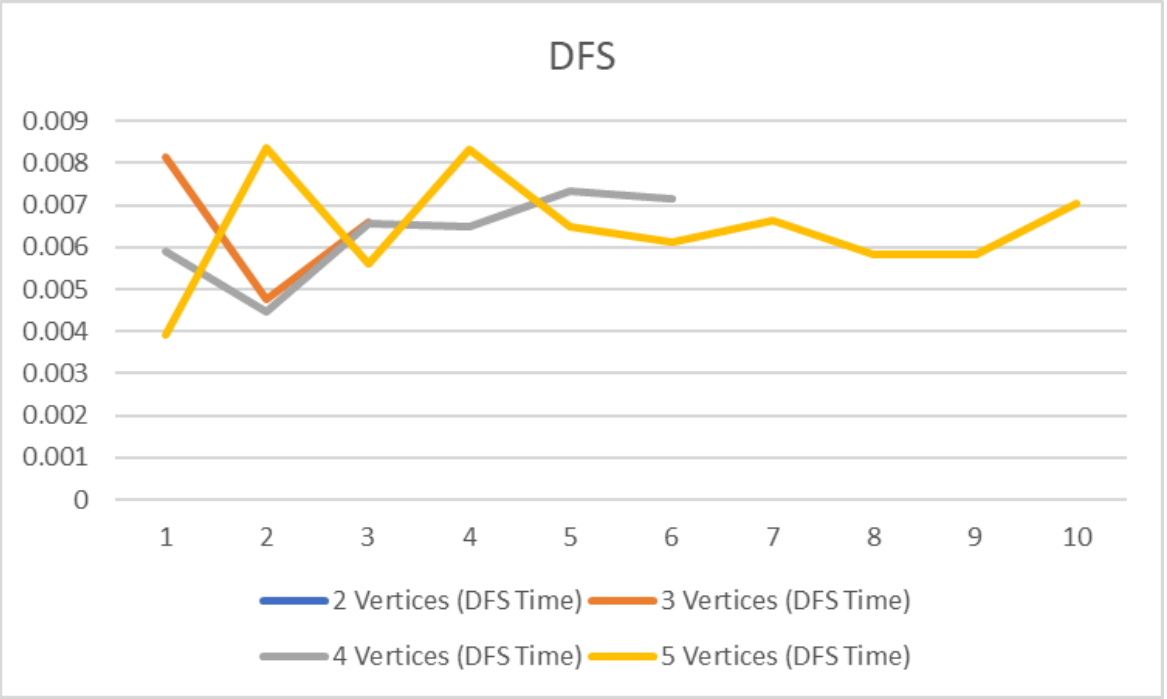
```
vert = ["A","B","C","D","E"]

ares = [
    [('A', 'B')],
    [('A', 'B'), ('B', 'C')],
    [('A', 'B'), ('B', 'C'), ('C', 'D')],
    [('A', 'B'), ('B', 'C'), ('C', 'D'), ('D', 'E')],
    [('A', 'B'), ('B', 'C'), ('C', 'D'), ('D', 'E'), ('A', 'C')],
    [('A', 'B'), ('B', 'C'), ('C', 'D'), ('D', 'E'), ('A', 'C'), ('A', 'D')],
    [('A', 'B'), ('B', 'C'), ('C', 'D'), ('D', 'E'), ('A', 'C'), ('A', 'D'), ('A', 'E')],
    [('A', 'B'), ('B', 'C'), ('C', 'D'), ('D', 'E'), ('A', 'C'), ('A', 'D'), ('A', 'E'), ('B', 'D')],
    [('A', 'B'), ('B', 'C'), ('C', 'D'), ('D', 'E'), ('A', 'C'), ('A', 'D'), ('A', 'E'), ('B', 'D'), ('B', 'E')],
    [('A', 'B'), ('B', 'C'), ('C', 'D'), ('D', 'E'), ('A', 'C'), ('A', 'D'), ('A', 'E'), ('B', 'D'), ('B', 'E'), ('C', 'E')]
]
```

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

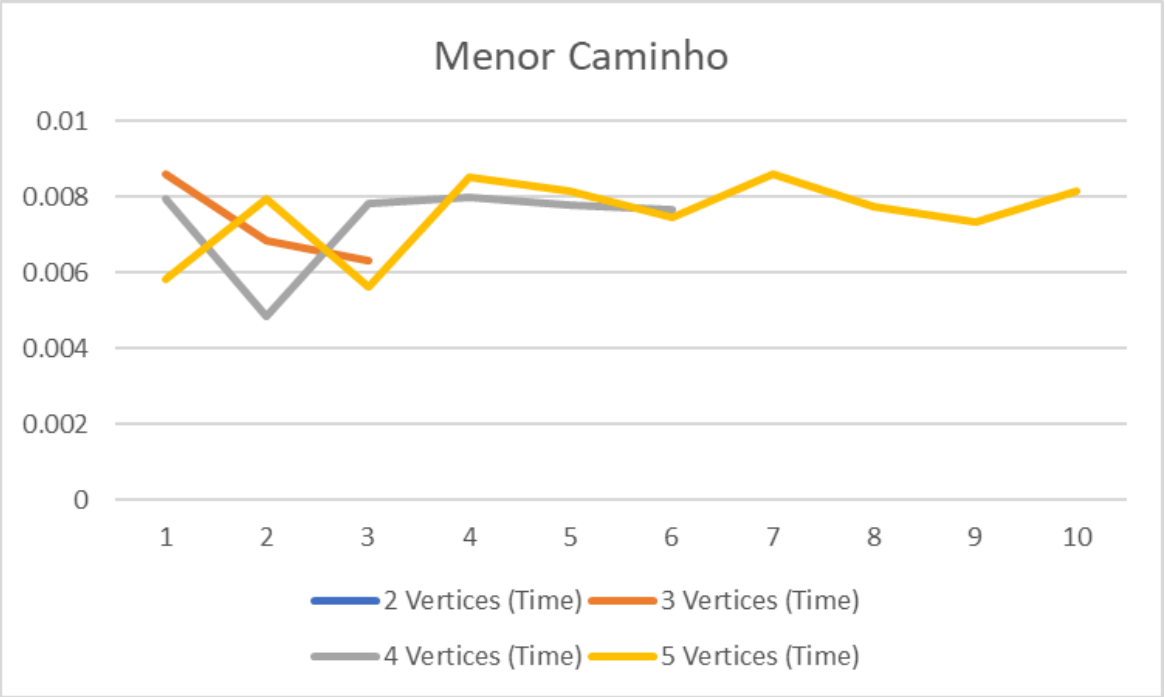
```
 2 Vertices

 1 arestas
['A', 'B']
0.006004799855872989 segundos - DFS
['A', 'B']
0.007186899892985821 segundos - BFS
['A', 'B']
0.004384299973025918 segundos - A -> B

 3 Vertices

 1 arestas
['A', 'B']
0.008151200134307146 segundos - DFS
['A', 'B']
0.0062478999607264996 segundos - BFS

 2 arestas
['A', 'B', 'C']
0.004767300095409155 segundos - DFS
['A', 'B', 'C']
0.006352300057187676 segundos - BFS
['A', 'B', 'C']
0.004656899953261018 segundos - A -> C

 3 arestas
['A', 'B', 'C']
0.006589399883523583 segundos - DFS
['A', 'B', 'C']
0.01038610003888607 segundos - BFS
['A', 'C']
0.0039773001335561275 segundos - A -> C

 4 Vertices

 1 arestas
['A', 'B']
0.005890199914574623 segundos - DFS
['A', 'B']
0.0041970000602304935 segundos - BFS

 2 arestas
['A', 'B', 'C']
0.004478299990296364 segundos - DFS
['A', 'B', 'C']
0.00608640001155436 segundos - BFS
```



Tempo x Arestas. Curiosamente, o tempo diminui com a quantidade maior de arestas e aumenta com uma quantidade menor de vértices, o'que não corresponde muito ao tempo O(V + A), senão seria O(V - A).

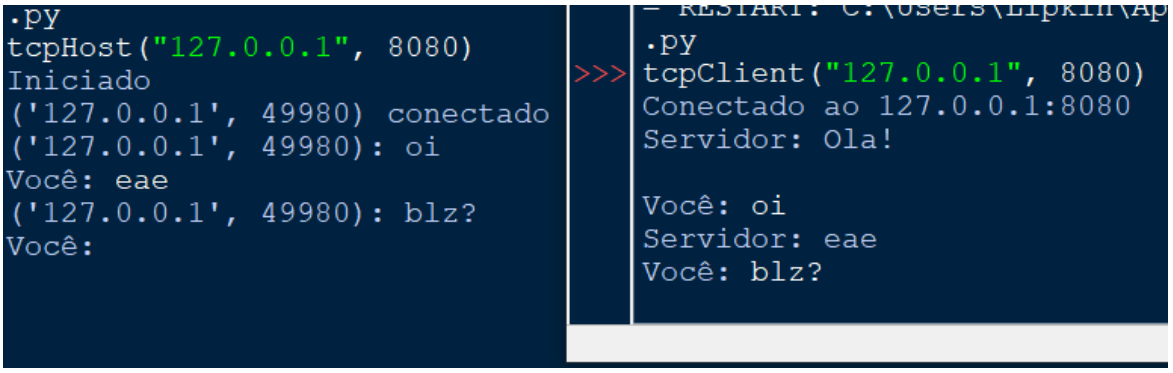


Aqui a complexidade de tempo faz mais sentido com o O(V+E) esperado, aumentando gradativamente no final de cada linha.

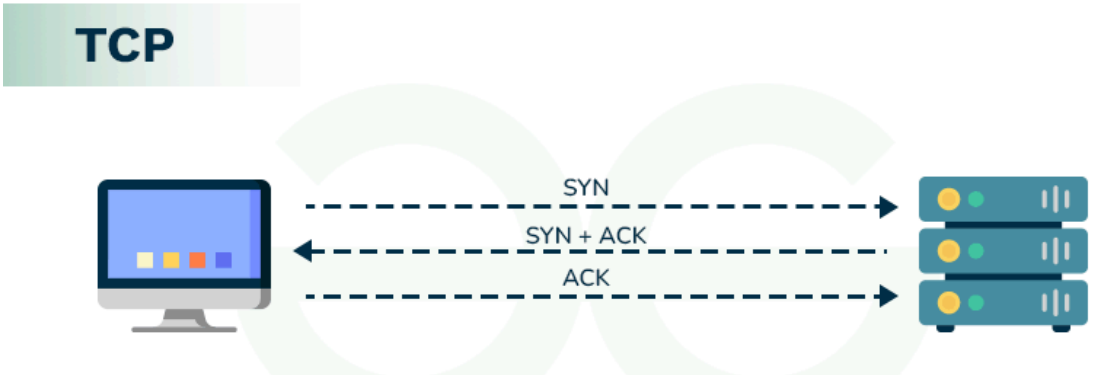


O tempo aumenta drasticamente em alguns lugares com o aumento do numero de arestas. A inconsistência causada pelo numero de vertices em alguns lugares complica a analise efetiva, ainda mais por ser um problema NP (não polinomial).

4.1 e 4.2



O protocolo TCP estabelece uma conexão entre cliente e servidor, e se qualquer um deles fecharem, nenhum pode mais receber dados do outro. Quando um lado envia um dado, o outro manda uma confirmação (ACK) de recebimento para que ele possa enviar seus proprios dados conforme a ilustração. Esse processo também é feito para estabelecer a conexão inicial.



https://media.geeksforgeeks.org/wp-content/uploads/20240226095336/TCP-Gif.gif

4.3 e 4.4

```
udpHost("127.0.0.1", 8080)
Iniciado
('127.0.0.1', 63998): oi
('127.0.0.1', 63998): oi
('127.0.0.1', 63998): oi
('127.0.0.1', 63998): AAAA
```

```
>>> udpClient("127.0.0.1", 8080)
Você: oi
('127.0.0.1', 63998) - ACK!
Você: oi
('127.0.0.1', 63998) - ACK!
Você: oi
('127.0.0.1', 63998) - ACK!
Você: AAAA
('127.0.0.1', 63998) - ACK!
Você:
```

O protocolo UDP não precisa estabelecer nenhuma conexão para enviar dados entre os dois lados. Dessa forma, ao contrario do TCP, pacotes perdidos ou enviados na “hora” errada ou serão recebidos mais tarde ou descartados sem que haja muita importância para a ordem ou validação deles.

https://media.geeksforgeeks.org/wp-content/uploads/20240226104348/UDP-gif.gif

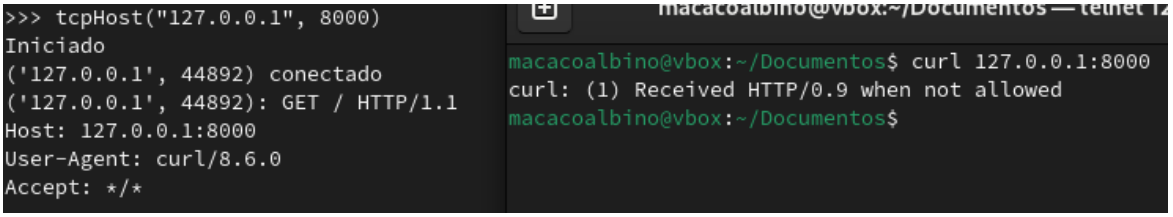
5.1

```
>>> tcpHost("127.0.0.1", 6000)
Iniciado
('127.0.0.1', 36850) conectado
('127.0.0.1', 36850): eae

Você: blz?
```

```
macacoalbino@vbox:~/Documentos$ telnet 127.0.0.1 6000
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
Ola!
eae
blz?
```

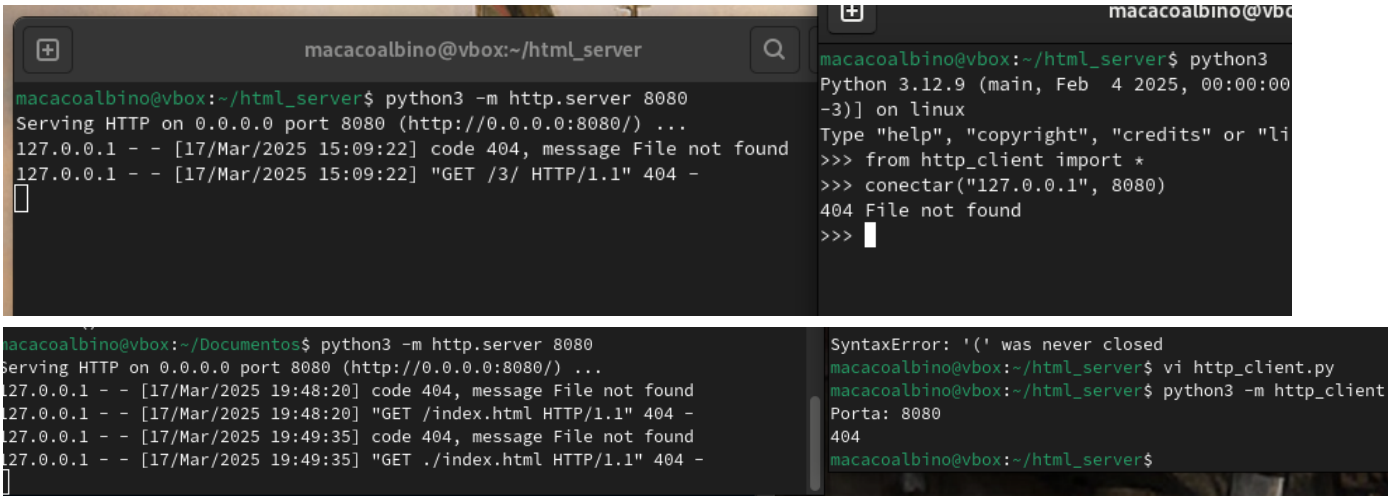
O telnet é um protocolo antigo que visava estabelecer conexão de um host com um servidor, enviando comandos e esperando a resposta do estado do servidor como se fosse um terminal remotamente controlado. Aqui ele envia e recebe os dados em bytes enquanto a conexão esta ativa, funcionando da mesma forma que no Grupo 4, onde cada lado espera receber dados depois de enviar uma mensagem.

5.2

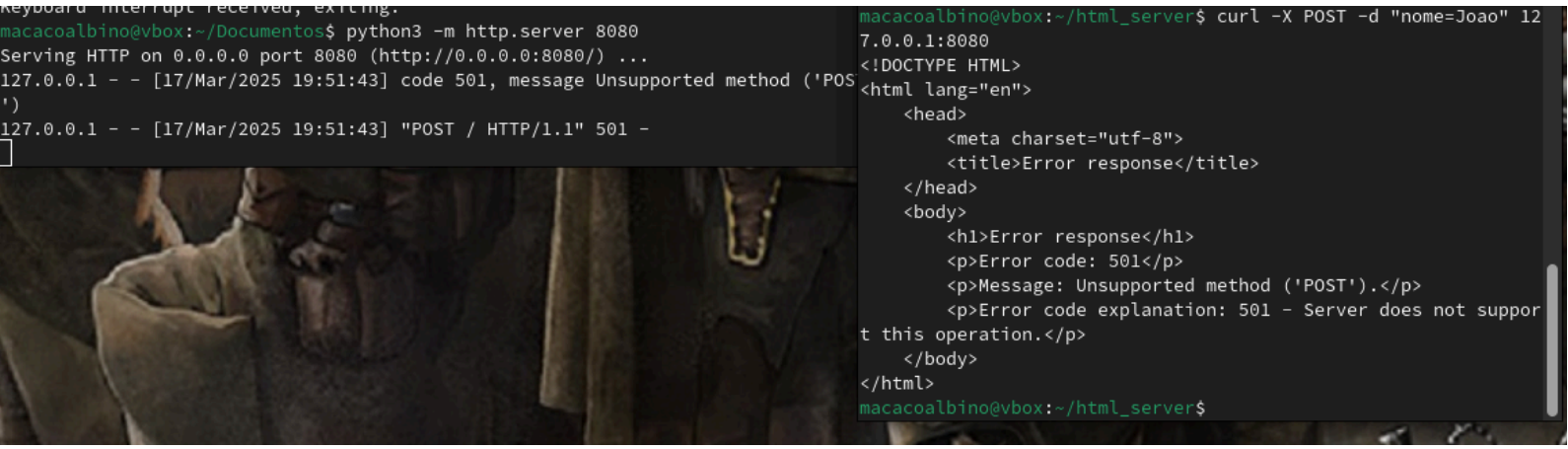


Ele manda o corpo de uma requisição HTTP nos bytes enviados e o servidor envia uma resposta que não é em formato HTTP, algo que o curl não espera.

5.3



5.4



O servidor da uma resposta invalida de HTML pois nenhuma função foi criada e associadas a um PATH que recebe requisições POST de HTTP.