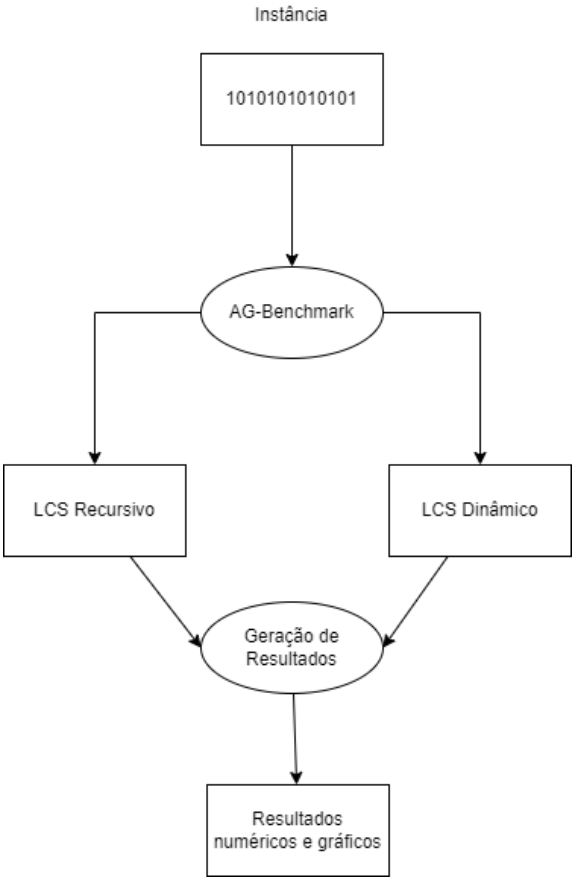
Universidade Federal do Ceará
***Campus* Quixadá**
Programa de Pós-Graduação em Computação - PCOMP
QPC0015 - ANÁLISE DE DESEMPENHO
Professor: Emanuel Ferreira Coutinho

Francisco Victor da Silva Pinheiro - 513770

1. **Nome do Benchmark:** GA-Benchmark

2. **Descrição do benchmark:** O Genetic Algorithm Benchmark é um benchmark para análise de desempenho de algoritmos de comparação de cadeias genéticas de DNA, sua análise é realizada com base nos tempos execução dado o tamanho das strings de entrada para os algoritmos. As cargas de trabalho estão divididas em três níveis: baixo, moderado e alto, sendo que é de escolha do usuário. A quantidade de repetição e crescimento das entradas é solicitada ao usuário pelo programa durante a execução.

3. **Uma imagem que descreva a arquitetura do benchmark**



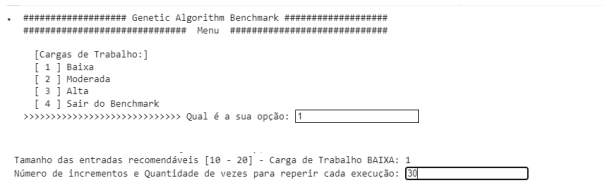
4. **Detalhes de código do benchmark:** Segue em anexo

5. **Métricas (descrição)**
    a) **Velocidade de Execução:** Tempo necessário para que o algoritmo realize sua execução de acordo com a carga de trabalho a ele fornecida.

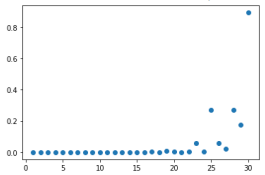
6. **Manual de instalação:**
    a) O programa foi escrito na linguagem de programação python, e para ser executado é necessário que a máquina do usuário possua o python3 instalado, caso não tenha, instalar o mesmo verificando o modo de instalação do seu sistema operacional.
    b) O arquivo utiliza algumas bibliotecas, que se caso não estejam instaladas, é necessário instalá-las. As bibliotecas são: matplolib, subprocess, numpy, time, psutil e os.

7. **Manual de operação**
    a) O seguinte menu é mostrado ao usuário, para que o mesmo escolha qual carga de trabalho utilizar:
    [Cargas de Trabalho:]
    [1] Baixa
    [2] Moderada
    [3] Alta
    [4] Sair do Benchmark
    b) Após a escolha, alguns comandos devem ser feitos pelo usuário, que são:
        - Tamanho das entradas recomendáveis de acordo com a carga de trabalho escolhida
        - Número de incrementos e Quantidade de vezes para repetir cada execução
    c) Após isso, o programa realiza a execução dos algoritmos, coletados os dados necessários para os resultados, e em seguida é mostrado os resultados individuais de cada algoritmo e depois o resultado geral.

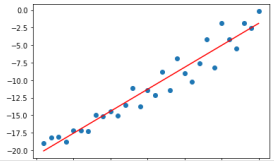
8. **Exemplos de utilização**
9. **Telas de resultados dos exemplos**



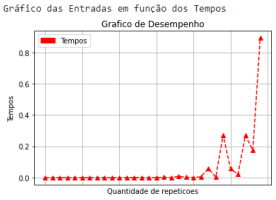
```
######################### Relatório ############################
############## Dados Gerais de cada algoritmo ###################

######################### LCS Recursivo ############################
Maior tamanho da entrada:  31
Quantidade de execuções para cada entrada:  30
Entradas [('', 'C'), ('T', 'A'), ('T', 'AA'), ('AG', 'GG'), ('AG', 'ACT'), ('GAA', 'GAC'), ('AAG', 'ATGA'), ('TGTG', 'ATTC'), ('CTGG', 'GGGGT'), ('TGACA', 'AAAGC'), ('TCTGG', 'TGCGGT'), ('ACGCGT', 'TGGTTT'), ('AACGAA', 'GTACCTC'), ('C
Tamanhos da Entradas [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
Tempos de cada execução [1.9073486328125e-06, 3.337860107421875e-06, 3.5762786865234375e-06, 2.1457672119140625e-06, 6.914138793945312 5e-06, 6.914138793945312 5e-06, 6.198883056640625e-06, 3.147125244140625e-05, 2.6464462280273438e-05,
Tempo Total da Execução 0.8926782608032227
Média dos tempos de execução 0.029755942026774087
Uso de CPU em porcentagem:  2.4 %
Uso médio de memória RAM em porcentagem:  13.2 %

######################### Gráficos ############################

Gráfico das Entradas em função dos Tempos
```



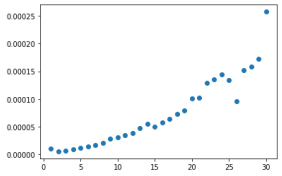
```
Regressão linear - Gráfico das Entradas em função dos Tempos
slope: [0.62729608]
```



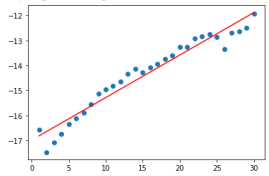
```
Gráfico das Entradas em função dos Tempos
```




```
##########################################################################

######################### LCS Dinâmico ############################
Maior Tamanho da entrada:  31
Quantidade de execuções para cada entrada:  30
Entradas [('', ''), ('', 'G'), ('T', 'T'), ('G', 'AA'), ('TA', 'AA'), ('CT', 'GTC'), ('GTT', 'TTT'), ('GTG', 'GCCA'), ('ACCA', 'GTAT'), ('GGCT', 'CTTAG'), ('ACTAG', 'AACCG'), ('GTGAC', 'ATCTTT'), ('CGTACA', 'AGCACG'), ('CCCCTT', 'TAAA
Tamanhos da Entradas [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
Tempos de cada execução [1.0251998901367188e-05, 5.4836273193359375e-06, 7.152557373046875e-06, 9.059906005859375e-06, 1.1920928955078125e-05, 1.4066696166992188e-05, 1.6450881958007812e-05, 2.0742416381835938e-05, 2.8133392333984375e
Tempo Total da Execução 0.0002574920654296875
Média dos tempos de execução 8.583068847656250e-06
Uso de CPU em porcentagem:  57.8 %
Uso médio de memória RAM em porcentagem:  13.2 %

######################### Gráficos ############################

Gráfico das Entradas em função dos Tempos
```



```
Regressão linear - Gráfico das Entradas em função dos Tempos
slope: [0.17037739]
```



```
slope conf interval: [0.15466882 0.18608596]

Gráfico das Entradas em função dos Tempos
```


```
##########################################################################
```

10. Um estudo de caso mais completo (cenário, projeto, execução, análises)

- Cenário: 2 algoritmos de comparação de strings, um projeto de forma recursiva e outro iterativa utilizando o método de programação dinâmica
- Projeto: Entrada [Tamanho 1]; Número de crescimento da entrada [40]; número de execuções para cada entrada [5];
- Execuções: 5 repetições da execução para cada tamanho de entrada
- Resultados - Numéricos

```
############################## Relatório ##############################
############### Dados Gerais de cada algoritmo ####################

########################## LCS Recursivo
##############################
Maior tamanho da entrada:  41
Quantidade de execuções para cada entrada:  40
Entradas [('', 'T'), ('G', 'G'), ('A', 'CT'), ('CA', 'TT'), ('TC',
'TCG'), ('GAA', 'CCC'), ('GGT', 'CAGC'), ('TCGG', 'CCCT'), ('TGAC',
'GATCC'), ('CTCAA', 'TTCCT'), ('CCCGT', 'GCGCTC'), ('ATCTCT',
'ATCTGT'), ('GTCTCG', 'CGAATAC'), ('ATTTCGC', 'TCACTTA'),
('GGAATGA', 'TAATTGGT'), ('AATTGTTA', 'GTCCGGGC'), ('TCATCCCC',
'CGGGACTGT'), ('GCCATGCAT', 'GGCAACGAG'), ('GGAGGATAG',
'GATCCTCATG'), ('CGTTCGCGGC', 'CCTGAGCGGC'), ('GGGCGATCAA',
'CCGACCTTAGA'), ('ACGCAACCGTT', 'CCGGCCAGCCA'), ('CCGTTGCTTGG',
'ATTGTTCAACGG'), ('GTGCCCCTGACC', 'TTTACTCGACAA'), ('CATTCCGTGTGA',
'TACTCTAAGGCTT'), ('GACTTGTTATTTC', 'CCATCGTACCAGC'),
('TGGGTGTCCGAAA', 'AATAGGTGATGAAC'), ('CGAGACAAGATTTG',
'TTACTTGGTATGAA'), ('GAAAGCCCGCAGCT', 'GATTAAAGTTGTCGT'),
('GTTATCATGGTCCAA', 'ATATTGCCAAAGGGG'), ('CCAGGTTTAGGAGAA',
'AAGTTCATTGCGCCCT'), ('AGGGTAATTGTGATCG', 'AGTAGAGTACCAAGTT'),
('CTTTAAAAAACCTTGA', 'GTACCGAGCTGGCTTGC'), ('GACTAGTCAAAGTTAGC',
'GACCAGAGGAGGTCCTT'), ('GTCTCTGGATTTGGTGG', 'GATAACCGATTCGGATTT'),
('GTCGCGAGAGCCAGTACA', 'AGGCGGATATATTGTGTA'),
('AGACTTAAGTTGTACACA', 'TGACCGTTTGCACGGGGGG'),
('TCGGAACGCAACACGCCGT', 'CGTGCTTAGGGGGCGCACA'),
('CGTATACGCGGAACAAGGC', 'CACTTGACTCGCCTACTTCT'),
('TGCAGCGTCCGTACAGCCGG', 'ATACCGCCTCGTGGCTCGCC')]
Tamanhos da Entradas [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
31, 32, 33, 34, 35, 36, 37, 38, 39, 40]
```

```
Tempos de cada execução [1.6689300537109375e-06,
2.6226043701171875e-06, 7.3909759521484375e-06,
9.775161743164062e-06, 9.298324584960938e-06,
2.5987625122070312e-05, 2.3126602172851562e-05,
3.600120544433594e-05, 2.6702880859375e-05, 8.511543273925781e-05,
3.981590270996094e-05, 3.719329833984375e-05,
0.0005345344543457031, 0.0006251335144042969,
0.0006396770477294922, 0.006042003631591797, 0.0025053024291992188,
0.0011491775512695312, 0.001117706298828125, 4.315376281738281e-05,
0.002126455307006836, 0.004964590072631836, 0.0038170814514160156,
0.01200723648071289, 0.025881528854370117, 0.04380226135253906,
0.046105146408081055, 0.4506046772003174, 0.2279045581817627,
1.5135300159454346, 4.126950979232788, 0.6865451335906982,
11.724200248718262, 5.9643778800964355, 10.712137460708618,
3.355220079421997, 101.0223445892334, 22.46193790435791,
62.792768716812134, 4.3423779010772705]
Tempo Total da Execução 4.3423779010772705
Média dos tempos de execução 0.10855944752693177
Uso de CPU em porcentagem:  40.7 %
Uso médio de memória RAM em porcentagem:  13.3 %
```

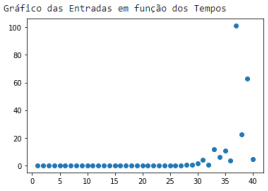
```
###################### LCS Dinâmico ############################
Maior Tamanho da entrada:  41
Quantidade de execuções para cada entrada:  40
Entradas [('', ''), ('', 'A'), ('G', 'G'), ('T', 'CT'), ('CC',
'TG'), ('GT', 'CGA'), ('CTT', 'CAT'), ('TAT', 'GCAT'), ('ACAT',
'AATG'), ('TCAG', 'GCCGT'), ('GAAAC', 'GACCA'), ('AGAGA',
'TTTGGA'), ('GACCGT', 'GTTATA'), ('AACCCC', 'TCCACCC'), ('ACATGCG',
'CCCATTT'), ('TCAAATC', 'AACGCAAG'), ('TTCGCGTT', 'GGCCGAGG'),
('ACAGCGTC', 'GCTCCTCTT'), ('GTAAGATCA', 'CGGTTAAGC'),
('ATAGGTGCG', 'TGACTGTCGT'), ('CTTTGCGGTT', 'CACCCTGGAG'),
('GGCCGCATGG', 'TGAAACCTCCA'), ('CTTACGAGTCG', 'TCCACCCCGTT'),
('GAAATTCCATT', 'GGAAACATGTAC'), ('TTAGGCCTTACC', 'GATTAGTTGTCC'),
('AGCCAATCTGGG', 'AATCGCTAAGGAG'), ('GCCGGCAGTGTTT',
'GCCTGCTTAACAC'), ('TTTAGGTCAAAAG', 'AGGCTTCTCTCACG'),
('TAACCAGATGTTTC', 'AAATTCAAAAGAAG'), ('TGCGAGACCCACGA',
'CGCTATAAGGGCACT'), ('ACAATAACGGCGGAA', 'CGGCAGCAATTCCGA'),
('TTACGGGTCCCTAGT', 'CTATGAACTGATCATA'), ('ATGGGAGACAAGGCCG',
'GTAGGTTGGTACGCGA'), ('TCGCAATCAGAAGTAG', 'CAGGTGTACAAGCTATG'),
('AGCGCGCACTGCAGTTG', 'GACTACGGTGCACGCCC'), ('CGAATCAACTGAAATAT',
'GGCTATGGTTATATCCTC'), ('CAGGTTTGAATGTGAACA',
'CGGAGCAAGCGCTATGAA'), ('AGTCACTCGGAGCCCTGT',
'CGCGCTGTTGGGTCCCGAT'), ('CAGGATCCGGTTCTGTGCC',
'TTTGCTTCCGTTATACCTG'), ('CCACCCCAGGTCAGGAAAA',
'TTCCCATCCGCCTTCGGTCA')]
Tamanhos da Entradas [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
31, 32, 33, 34, 35, 36, 37, 38, 39, 40]
Tempos de cada execução [1.0728836059570312e-05,
5.7220458984375e-06, 7.3909759521484375e-06, 9.775161743164062e-06,
1.2159347534179688e-05, 1.430511474609375e-05,
1.811981201171875e-05, 2.193450927734375e-05,
2.6226043701171875e-05, 2.8133392333984375e-05,
3.6716461181640625e-05, 4.4345855712890625e-05,
4.863739013671875e-05, 4.982948303222656e-05,
5.173683166503906e-05, 6.198883056640625e-05,
7.390975952148438e-05, 7.867813110351562e-05,
8.153915405273438e-05, 6.389617919921875e-05,
6.604194641113281e-05, 7.033348083496094e-05,
```

```
7.748603820800781e-05, 8.511543273925781e-05,
0.00011110305786132812, 0.00010156631469726562,
0.00010395050048828125, 0.00014138221740722656,
0.00011539459228515625, 0.00014591217041015625,
0.0001342296600341797, 0.00018715858459472656,
0.0001506805419921875, 0.00016951560974121094,
0.00016808509826660156, 0.00018596649169921875,
0.0001938343048095703, 0.00019621849060058594,
0.00020742416381835938, 0.0002276897430419922]
Tempo Total da Execução 0.0002276897430419922
Média dos tempos de execução 5.692243576049805e-06
Uso de CPU em porcentagem:  55.3 %
Uso médio de memória RAM em porcentagem:  13.3 %
```

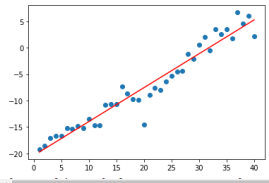
```
################ Resultados Gerais ####################
Tempo total de execução:  4.3426055908203125
Tempo médio de execução:  0.10856513977050782
Uso de CPU em porcentagem:  56.3 %
Uso médio de memória RAM em porcentagem:  13.2 %
```

- Resultados - Gráficos e Números:

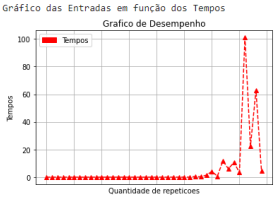
```
################## Genetic Algorithm Benchmark ##################
##############################  Menu  ############################

  [Cargas de Trabalho:]
  [ 1 ] Baixa
  [ 2 ] Moderada
  [ 3 ] Alta
  [ 4 ] Sair do Benchmark
>>>>>>>>>>>>>>>>>>>>>>>>>>> Qual é a sua opção: 2
Tamanho das entradas recomendáveis [20 - 30] - Carga de Trabalho BAIXA: 1
Número de incrementos e Quantidade de vezes para reperir cada execução: 40

########################## Relatório ############################
############### Dados Gerais de cada algoritmo ##################

######################## LCS Recursivo ##########################
Maior tamanho da entrada:  41
Quantidade de execuções para cada entrada:  40
Entradas [('', 'T'), ('G', 'G'), ('A', 'CT'), ('CA', 'TT'), ('TC', 'TCG'), ('GAA', 'CCC'), ('GGT', 'CAGC'), ('TCGG', 'CCCT'), ('TGAC', 'GATCC'), ('CTCAA', 'TTCCT'), ('CCCGT', 'GCGCTC'), ('ATCTCT', 'ATCTGT'), ('GTCTCG', 'CGAATAC'), ('ATTTCGC', 'TCA
Tamanhos da Entradas [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]
Tempos de cada execução [1.6689300537109375e-06, 2.622604370117875e-06, 7.3909759521484375e-06, 9.775161743164062e-06, 9.298324584960938e-06, 2.5987625122070312e-05, 2.3126602172851562e-05, 3.600120544433594e-05, 2.6702880859375e-05, 8.511543273
Tempo Total da Execução 4.3423779010772705
Média dos tempos de execução 0.10855944752693177
Uso de CPU em porcentagem:  40.7 %
Uso médio de memória RAM em porcentagem:  13.3 %

########################## Gráficos #############################

Gráfico das Entradas em função dos Tempos
```



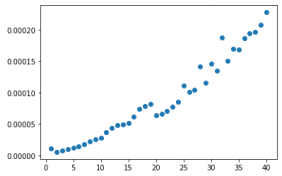
```
Regressão linear - Gráfico das Entradas em função dos Tempos
slope: [0.64098937]
```



```
Gráfico das Entradas em função dos Tempos
```




```
########################################################################

###################### LCS Dinâmico #############################
Maior Tamanho da entrada:  41
Quantidade de execuções para cada entrada:  40
Entradas [('', ''), ('', 'A'), ('G', 'G'), ('T', 'CT'), ('CC', 'TG'), ('GT', 'CGA'), ('CTT', 'CAT'), ('TAT', 'GCAT'), ('ACAT', 'AATG'), ('TCAG', 'GCCGT'), ('GAAAC', 'GACCA'), ('AGAGA', 'TTTGGA'), ('GACCGT', 'GTTATA'), ('AACCCC', 'TCCACCC'), ('ACA
Tamanhos da Entradas [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]
Tempos de cada execução [1.0728836059570312e-05, 5.7220458984375e-06, 7.390975952148437 5e-06, 9.775161743164062e-06, 1.2159347534179688e-05, 1.4305114746093 75e-05, 1.8119812011718 75e-05, 2.193450927734375e-05, 2.622604370117875e-05, 2.8133392333
Tempo Total da Execução 0.0002276897430419922
Média dos tempos de execução 5.692243576049805e-06
Uso de CPU em porcentagem:  55.3 %
Uso médio de memória RAM em porcentagem:  13.3 %

########################## Gráficos #############################

Gráfico das Entradas em função dos Tempos
```



```
Regressão linear - Gráfico das Entradas em função dos Tempos
slope: [0.12032972]
```

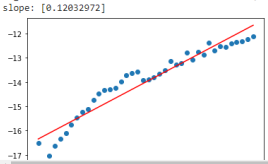
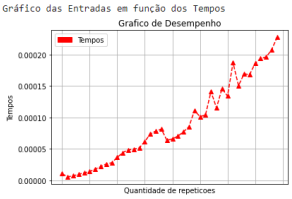


Gráfico das Entradas em função dos Tempos




```
######################################################################


############### Resultados Gerais ####################
Tempo total de execução:  4.3426055908203125
Tempo médio de execução:  0.10856513977050782
Uso de CPU em porcentagem:  56.3 %
Uso médio de memória RAM em porcentagem:  13.2 %

############################## Gráficos ###################################
```

Gráfico dos Tempos de Execução Recusivo

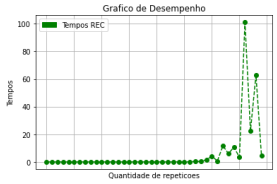
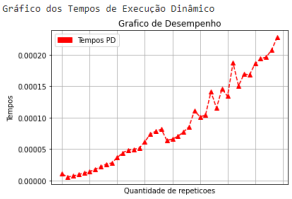


Gráfico dos Tempos de Execução Dinâmico



ANEXO 1 - PARTE CÓDIGO FONTE

```
1.
2.      tamanho_entrada = int(input('Tamanho das entradas recomendáveis [10 - 20] - Carga de Trabalho BAIXA: '))
3.      #crescimento_entradas = int(input('Número de incrementosQuantidade de vezes para reperir cada execução'))
4.      quantidade_execucoes = int(input('Número de incrementos e Quantidade de vezes para reperir cada execução: '))
5.      tempoTotalDasExecucoes = 0
6.      tempoTotalDasExecucoes1 = 0
7.      tempoMedioTotaldasExecucoes = 0
8.
9.      temposrec = []
10.     tempospd = []
11.     print('')
12.
13.                         print("############################      Relatório ##############################")
14.             print("###############   Dados   Gerais   de   cada   algoritmo ####################")
15.
16.     # LCS recursivo
17.     def lcs(X, Y, m, n):
18.         if m == 0 or n == 0:
19.             return 0
20.         elif X[m-1] == Y[n-1]:
21.             return lcs(X, Y, m-1, n-1) + 1
22.         else:
23.             return max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n))
24.
25.     # LCS prog dinamica
26.     def lcs_pd(X, Y, m, n):
27.       L1 = [[0]*(n + 1) for i in range(m + 1)]
28.
29.       for i in range(m + 1):
30.           for j in range(n + 1):
31.               if i == 0 or j == 0 :
32.                   L1[i][j] = 0
33.               elif X[i-1] == Y[j-1]:
```

```
                L1[i][j] = L1[i-1][j-1]+1
            else:
                L1[i][j] = max(L1[i-1][j], L1[i][j-1])

    return L1[m][n]

# tempo de execução do recursivo
def time_out_re(X,Y,m,n):
    start = time.time()
    lcs(X, Y, m, n)
    end = time.time()
    tempo = end - start
    return tempo

# tempo de execução do prog dinamica
def time_out_pd(X,Y,m,n):
    start = time.time()
    lcs_pd(X, Y, m, n)
    end = time.time()
    tempo = end - start
    return tempo

# função que calcula o tempo total da execução
def time_total_de_execucao(X,Y,m,n):
    start_rec = time.time()
    lcs(X, Y, m, n)
    end_rec = time.time()
    tempo_execucao_rec = end_rec - start_rec

    start_pd = time.time()
    lcs_pd(X, Y, m, n)
    end_pd = time.time()
    tempo_execucao_pd = end_pd - start_pd

    tempoTotal = (tempo_execucao_rec + tempo_execucao_pd)

    return tempoTotal

```

```
72.  # função de execução da LCS recursivo
73.  def executor_re(tamanho_entrada):
74.    conteudo = []
75.    conteudo2 = []
76.    conteudo3 = []
77.    tempos = []
78.
79.
80.    temp_m = 0
81.    medias = 0
82.    entradas = []
83.    tamanhos = []
84.
85.    for i in range(quantidade_execucoes):
86.        V = ["A","T","C","G"]
87.        W = ""
88.        for i in range(tamanho_entrada):
89.            W = W + V[np.random.choice(range(len(V)))]
90.        meio = len(W)//2
91.        X = W[:meio]
92.        Y = W[meio:]
93.        m = len(X)
94.        n = len(Y)
95.        entradas.append((X, Y))
96.        tamanhos.append(tamanho_entrada)
97.        temp = time_out_re(X, Y, m, n)
98.        global tempoTotalDasExecucoes
99.        tempoTotalDasExecucoes = temp
100.          tempos.append(temp)
101.          global temposrec
102.          temposrec.append(temp)
103.          temp_m = temp / quantidade_execucoes
104.
105.          medias = (temp_m / quantidade_execucoes)
106.          tamanho_entrada = tamanho_entrada + 1
107.
108.      print('')
```

```
109.                         print("#########################    LCS    Recursivo
    ##############################")
110.         print("Maior tamanho da entrada: ", tamanho_entrada)
111.                 print('Quantidade  de  execuções  para  cada  entrada:  ',
    quantidade_execucoes)
112.         print('Entradas', entradas)
113.         print('Tamanhos da Entradas', tamanhos)
114.         print('Tempos de cada execução', tempos)
115.         print('Tempo Total da Execução', temp)
116.         print('Média dos tempos de execução', temp_m)
117.         print('Uso de CPU em porcentagem: ', psutil.cpu_percent(),'%')
118.                 print('Uso  médio  de  memória  RAM  em  porcentagem:  ',
    psutil.virtual_memory().percent,'%')
119.         print('')
120.                             print("#############################    Gráficos
    ################################")
121.         print('')
122.         print('Gráfico das Entradas em função dos Tempos')
123.         x = np.array(tamanhos)
124.         y = np.array(tempos)
125.
126.         lx = x
127.         ly = y
128.
129.         plt.scatter(x, ly)
130.         plt.show()
131.
132.         print('')
133.            print('Regressão  linear  -  Gráfico  das  Entradas  em  função  dos
    Tempos')
134.         x = np.array(tamanhos)
135.         y = np.array(tempos)
136.
137.         lx = x
138.         ly = np.log2(y)
139.
140.           #regressão linear, tranformar a curva do gráfico em uma reta a
    partir de um certo ponto
141.         from sklearn.linear_model import LinearRegression
```

```
142.        model = LinearRegression().fit(lx.reshape(-1, 1), ly)
143.        print('slope:', model.coef_)
144.
145.        plt.scatter(lx, ly)
146.        plt.plot(lx, model.intercept_ + model.coef_ * lx, 'r')
147.        plt.show()
148.
149.        # intervalo de confiança, grau do polinomio
150.        import statsmodels.api as sm
151.        lx = sm.add_constant(lx)
152.        res = sm.OLS(ly, lx).fit()
153.        print('slope conf interval:', res.conf_int(0.05)[1])
154.
155.        print('')
156.        print('Gráfico das Entradas em função dos Tempos')
157.        #plt.plot(tamanhos, 'go')  # green bolinha
158.        #plt.plot(tamanhos, 'k--', color='green')  # linha pontilha
159.
160.        plt.plot(tempos, 'r^')  # red triangulo
161.        plt.plot(tempos, 'k--', color='red')  # linha tracejada
162.
163.        plt.title("Grafico de Desempenho")
164.        plt.tick_params(axis='x', which='both', bottom=False,
165.                        top=False, labelbottom=False)
166.        red_patch = mpatches.Patch(color='red', label='Tempos')
167.        #green_patch = mpatches.Patch(color='green', label='Tempos')
168.        plt.legend(handles=[red_patch])
169.
170.        plt.grid(True)
171.        plt.xlabel("Quantidade de repeticoes")
172.        plt.ylabel("Tempos")
173.        plt.show()
174.
175.
    print('#########################################################################
    ####')
176.
177.      # função de execução da LCS recursivo
178.      def executor_pd(tamanho_entrada):
```

```
179.        conteudo = []
180.        conteudo2 = []
181.        conteudo3 = []
182.        tempos = []
183.
184.        temp_m = 0
185.        medias = 0
186.        entradas = []
187.        tamanhos = []
188.
189.
190.        #for j in range(crescimento_entradas):
191.        for i in range(quantidade_execucoes):
192.            V = ["A","T","C","G"]
193.            W = ""
194.            for i in range(tamanho_entrada - 1):
195.                W = W + V[np.random.choice(range(len(V)))]
196.            meio = len(W)//2
197.            X = W[:meio]
198.            Y = W[meio:]
199.            m = len(X)
200.            n = len(Y)
201.            entradas.append((X, Y))
202.            tamanhos.append(tamanho_entrada)
203.            temp = time_out_pd(X, Y, m, n)
204.            global tempoTotalDasExecucoes1
205.            tempoTotalDasExecucoes1 = temp
206.            tempos.append(temp)
207.            global tempospd
208.            tempospd.append(temp)
209.            temp_m = temp / quantidade_execucoes
210.            medias = (temp_m / quantidade_execucoes)
211.
212.
213.            #global crescimento_entradas
214.
215.            #if():
216.              #opp = False
```

```
217.                 #break
218.             #else:
219.             tamanho_entrada = tamanho_entrada + 1
220.
221.                 #print('tamanho entrada', tamanho_entrada)
222.
223.         print('')
224.                             print("########################    LCS    Dinâmico
    ############################")
225.         print("Maior Tamanho da entrada: ", tamanho_entrada)
226.                  print('Quantidade  de  execuções  para  cada  entrada:  ',
    quantidade_execucoes)
227.         print('Entradas', entradas)
228.         print('Tamanhos da Entradas', tamanhos)
229.         print('Tempos de cada execução', tempos)
230.         print('Tempo Total da Execução', temp)
231.         print('Média dos tempos de execução', temp_m)
232.         print('Uso de CPU em porcentagem: ', psutil.cpu_percent(),'%')
233.                  print('Uso  médio  de  memória  RAM  em  porcentagem:  ',
    psutil.virtual_memory().percent,'%')
234.         print('')
235.                             print("##############################    Gráficos
    ################################")
236.         print('')
237.         print('Gráfico das Entradas em função dos Tempos')
238.         x = np.array(tamanhos)
239.         y = np.array(tempos)
240.
241.         lx = x
242.         ly = y
243.
244.         plt.scatter(x, ly)
245.         plt.show()
246.
247.         print('')
248.            print('Regressão  linear  -  Gráfico  das  Entradas  em  função  dos
    Tempos')
249.         x = np.array(tamanhos)
250.         y = np.array(tempos)
```

```
251.
252.         lx = x
253.         ly = np.log2(y)
254.
255.          #regressão linear, tranformar a curva do gráfico em uma reta a
    partir de um certo ponto
256.         from sklearn.linear_model import LinearRegression
257.         model = LinearRegression().fit(lx.reshape(-1, 1), ly)
258.         print('slope:', model.coef_)
259.
260.         plt.scatter(lx, ly)
261.         plt.plot(lx, model.intercept_ + model.coef_ * lx, 'r')
262.         plt.show()
263.
264.         # intervalo de confiança, grau do polinomio
265.         import statsmodels.api as sm
266.         lx = sm.add_constant(lx)
267.         res = sm.OLS(ly, lx).fit()
268.         print('slope conf interval:', res.conf_int(0.05)[1])
269.
270.         print('')
271.         print('Gráfico das Entradas em função dos Tempos')
272.         #plt.plot(tamanhos, 'go')  # green bolinha
273.         #plt.plot(tamanhos, 'k--', color='green')  # linha pontilha
274.
275.         plt.plot(tempos, 'r^')  # red triangulo
276.         plt.plot(tempos, 'k--', color='red')  # linha tracejada
277.
278.         plt.title("Grafico de Desempenho")
279.         plt.tick_params(axis='x', which='both', bottom=False,
280.                         top=False, labelbottom=False)
281.         red_patch = mpatches.Patch(color='red', label='Tempos')
282.         #green_patch = mpatches.Patch(color='green', label='Tempos')
283.         plt.legend(handles=[red_patch])
284.
285.         plt.grid(True)
286.         plt.xlabel("Quantidade de repeticoes")
287.         plt.ylabel("Tempos")
288.         plt.show()
```

```
289.
290.
    print('#####################################################################
    ####')
291.
292.        executor_re(tamanho_entrada)
293.        executor_pd(tamanho_entrada)
294.                tempoTotalDasExecucoesTotal    =    tempoTotalDasExecucoes    +
    tempoTotalDasExecucoes1
295.            tempoMedioTotaldasExecucoes   =   (tempoTotalDasExecucoesTotal   /
    quantidade_execucoes)
296.        print('')
297.        print('')
298.        print('')
299.        print("############### Resultados Gerais ####################")
300.        print('Tempo total de execução: ', tempoTotalDasExecucoesTotal)
301.        print('Tempo médio de execução: ', tempoMedioTotaldasExecucoes)
302.        #print('Variança do tempo de execução: ')
303.        print('Uso de CPU em porcentagem: ', psutil.cpu_percent(),'%')
304.              print('Uso   médio   de   memória   RAM   em   porcentagem:   ',
    psutil.virtual_memory().percent,'%')
305.        print('')
306.                            print("############################      Gráficos
    ################################")
307.
308.        print('')
309.        print('Gráfico dos Tempos de Execução Recusivo')
310.        plt.plot(temposrec, 'go')  # green bolinha
311.        plt.plot(temposrec, 'k--', color='green')  # linha pontilha
312.
313.        #plt.plot(tempospd, 'r^')  # red triangulo
314.        #plt.plot(tempospd, 'k--', color='red')  # linha tracejada
315.
316.        plt.title("Grafico de Desempenho")
317.        plt.tick_params(axis='x', which='both', bottom=False,
318.                        top=False, labelbottom=False)
319.        #red_patch = mpatches.Patch(color='red', label='Tempos REC')
320.        green_patch = mpatches.Patch(color='green', label='Tempos REC')
321.        plt.legend(handles=[green_patch])
```

```
322.
323.        plt.grid(True)
324.        plt.xlabel("Quantidade de repeticoes")
325.        plt.ylabel("Tempos")
326.        plt.show()
327.
328.
329.        print('')
330.        print('Gráfico dos Tempos de Execução Dinâmico')
331.        #plt.plot(temposrec, 'go')  # green bolinha
332.        #plt.plot(temposrec, 'k--', color='green')  # linha pontilha
333.
334.        plt.plot(tempospd, 'r^')  # red triangulo
335.        plt.plot(tempospd, 'k--', color='red')  # linha tracejada
336.
337.        plt.title("Grafico de Desempenho")
338.        plt.tick_params(axis='x', which='both', bottom=False,
339.                        top=False, labelbottom=False)
340.        red_patch = mpatches.Patch(color='red', label='Tempos PD')
341.        #green_patch = mpatches.Patch(color='green', label='Tempos REC')
342.        plt.legend(handles=[red_patch])
343.
344.        plt.grid(True)
345.        plt.xlabel("Quantidade de repeticoes")
346.        plt.ylabel("Tempos")
347.        plt.show()
348.
349.        print('')
350.
351.        #Opções depois do relatório gerado
352.        print('Opções: ')
353.        print('''
354.        [1] - Continuar
355.        [2] - Sair''')
356.
357.        op = 0
358.        op = int(input('>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Deseja continuar, para
   mais execuções? '))
359.
```

```
360.      if op == 1:
361.        print('')
362.        print('Continuando...')
363.        print('')
364.
365.      elif op == 2:
366.        print('')
367.        print('Saindo...')
368.        print('')
369.        break
370.
371.      else:
372.        print('Opção inválida!!!')
```