Chuong Vu

Bioinformatics COMP5800

Homework 3

Prof. Kim Byung

HOMEWORK 3 – OVERLAP LAYOUT CONCENSUS

1. DATA STRUCTURE

For this homework, I use Python because python is object oriented programming which help me easy to trouble shoot and this will eliminate of memory allocation.

Fragments are randomly pick from the DNA sequence with the length is 400bp. Number of fragments to be read is determined by the following formula:

Fragments = (DNA Sequence’s length / 400) \* 10

My program is break down into 8 functions and each one do a certain job:

1. Remove the N and other character
2. Make a fragments file from the cleaned up file
3. Calculate the overlap and save to overlap graph from reads
4. Get the largest overlap graph from the graph file
5. Create path from the largest overlap graph
6. Calculate new DNA sequences from overlap path
7. Create one new DNA sequence from new DNA sequences
8. Compare new and original DNA

Each function will read an input file and save to an output file. I break it down because it will help me easy to keep track and I don’t have to run from the beginning of the OLC.

From the original dataset, I failed to do the OLC because the fragments are too big so the server can’t handle it and it ran out of memory. It was about over 1 million of fragments. I tried to run the dataset and left it over 2 days and it still can’t finished it and it out of memory when it try to calculate the overlap data (step 6 to 8 was never done after few days). So I had no choice to stop the program and break the original dataset into a smaller dataset.

I extracted 200/1000 lines from the original dataset to test it out. With the same method that read data, fragments’ length 400bp. Once the fragments table is created, I built the overlap map and the graph between fragments into array of Dictionary (which I can keep track the keys and values from the DNA). Each item in Dictionary contained index of next fragment and overlap character number of current fragment’s suffix and next fragment’s prefix (this is done in function 4). And then I built new path from based on the highest overlap values that get from the Dictionary’s array. From the array, a new array that contain map of paths of adjacent overlap fragments will be built (function 6). Finally, from the new fragments table, a new DNA sequence will be created based on the max of columns (2-d table matrix) and the length of the new DNA will be the length of the longest new fragments.

1. COMPARISON

Because the time consuming, I don’t have a chance and machine that can handle the whole dataset, I only run the small dataset. I did the test with 200/1000lines from both of the original DNAs. Four comparisons are done.

1. Original DNA\_1 vs New DNA\_2
2. Original DNA\_2 vs New DNA\_2
3. Original DNA\_1 vs New DNA\_3
4. Original DNA\_2 vs New DNA\_3

Notes: The DNA\_3 is the combination from the Original DNA\_1 and Original DNA\_2

1. RESULTS

*With 200 lines*

cvu@cs3:~/BIO/HWT$ time python3 HW5.py

1. Clean Up - Done

2. Get Fragment - Done

3. Calculate Overlap - Done

4. Get Largest Overlap Graph - Done

5. Get Path - Done

6. Get New DNA Sequences - Done

7. Create one new DNA sequence - Done

Org DNA 1 vs New OLC DNA 1

Org DNA 1 Length = 14070

New DNA 1 Length = 13974

Number's Overlap = 4159

Org DNA 2 vs New OLC DNA 2

Org DNA 2 Length = 14070

New DNA 2 Length = 13974

Number's Overlap = 4747

Org DNA 1 vs New OLC DNA 3

Org DNA 1 Length = 28140

New DNA 3 Length = 13974

Number's Overlap = 4404

Org DNA 2 vs New OLC DNA 3

Org DNA 2 Length = 28140

New DNA 3 Length = 13974

Number's Overlap = 4747

8. Compare two DNAs - Done

real 7m46.181s

user 7m42.512s

sys 0m0.584s

*With 1000 lines*

cvu@cs3:~/BIO/HWT$ time python3 HW5.py

1. Clean Up - Done

2. Get Fragment - Done

3. Calculate Overlap - Done

4. Get Largest Overlap Graph - Done

5. Get Path - Done

6. Get New DNA Sequences - Done

7. Create one new DNA sequence - Done

Org DNA 1 vs New OLC DNA 1

Org DNA 1 Length = 70070

New DNA 1 Length = 69980

Number's Overlap = 21014

Org DNA 2 vs New OLC DNA 2

Org DNA 2 Length = 70070

New DNA 2 Length = 69945

Number's Overlap = 20615

Org DNA 1 vs New OLC DNA 3

Org DNA 1 Length = 140140

New DNA 3 Length = 69980

Number's Overlap = 21373

Org DNA 2 vs New OLC DNA 3

Org DNA 2 Length = 140140

New DNA 3 Length = 69945

Number's Overlap = 20615

8. Compare two DNAs - Done

real 200m27.680s (3h30)

user 199m4.168s

sys 0m35.660s

From the results we can see that the new OLC sequence is consistently overlapping about 1/3 of original DNA. New OLC DNA sequence from the mixed both original DNA is less overlapping than just one but the number is not really wide apart from the test runs.

1. SOURCE CODE PYTHON
2. #
3. #        Chuong Vu
4. #        Bioinformatics
5. #        Homework 5
6. #        Prof Kim Byung
7. #
8. #
9. **import** pprint
10. **import** random
11. **import** json
12. **from** array **import** array
13. **from** collections **import** OrderedDict
14. **import** timeit
15. #from collections import deque
16. # reads with average 400 bp
17. reads **=** **400**
18. ## Working files
19. # I manually copy 200 line from the removed N
20. # to write the test because the original file is too long
21. # it cause my program out of memory
22. DNA\_File1\_FASTA **=** "chr01.fasta"
23. DNA\_File1\_RmN **=** "chr01.RemoveN.orig"
24. DNA\_File1\_Orig **=** "chr01.orig"
25. DNA\_File1\_Frag **=** "chr01.frag"
26. DNA\_File1\_Graph **=** "chr01.graph"
27. DNA\_File1\_Graph\_Largest **=** "chr01.graph.largest"
28. DNA\_File1\_Path **=** "chr01.path"
29. DNA\_File1\_NewDNA\_Seqs **=** "chr01.newdna.seqs"
30. DNA\_File1\_NewDNA **=** "chr01.newdna"
31. DNA\_File2\_FASTA **=** "chr21.fasta"
32. DNA\_File2\_RmN **=** "chr21.RemoveN.orig"
33. DNA\_File2\_Orig **=** "chr21.orig"
34. DNA\_File2\_Frag **=** "chr21.frag"
35. DNA\_File2\_Graph **=** "chr21.graph"
36. DNA\_File2\_Graph\_Largest **=** "chr21.graph.largest"
37. DNA\_File2\_Path **=** "chr21.path"
38. DNA\_File2\_NewDNA\_Seqs **=** "chr21.newdna.seqs"
39. DNA\_File2\_NewDNA **=** "chr21.newdna"
40. # Combined from both file
41. DNA\_File3\_RmN **=** "chrc"
42. DNA\_File3\_Orig **=** "chrc.orig"
43. DNA\_File3\_Frag **=** "chrc.frag"
44. DNA\_File3\_Graph **=** "chrc.graph"
45. DNA\_File3\_Graph\_Largest **=** "chrc.graph.largest"
46. DNA\_File3\_Path **=** "chrc.path"
47. DNA\_File3\_NewDNA\_Seqs **=** "chrc.newdna.seqs"
48. DNA\_File3\_NewDNA **=** "chrc.newdna"
49. #
50. # 1. Remove all N character and SPACE
51. #
52. **def** CleanUpOriginalFileToNewFile**(**InputFile**,** OutputFile**):**
53. with open**(**InputFile**,** "r"**)** as infile**,** open**(**OutputFile**,** "w"**)** as outfile**:**
54. data **=** infile**.**read**()**
55. data **=** data**.**replace**(**'N'**,** ''**)**
56. data **=** data**.**replace**(**' '**,** ''**)**
57. outfile**.**write**(**data**)**
58. **return**
59. #    2.
60. #    Function use to create fragment file from short file
61. #    This is get random by length of string \* 10 and / reads
62. #
63. **def** CreateFragFile**(**InputFile**,** OutputFile**,** OutputFile1**):**
64. FileRead **=** open**(**InputFile**,** "r"**)**
66. lines **=** FileRead**.**readlines**()**
67. DNA\_Seq **=** ''
69. #DNA\_Seq = infile.read()
70. #DNA\_Seq = DNA\_Seq.replace('\n', '')
71. **for** line **in** lines**:**
72. line **=** line**.**replace**(**'\n'**,**''**)**
73. **if** **not** line**.**startswith**(**'>'**):**
74. DNA\_Seq **+=** line
76. FileWrite **=** open**(**OutputFile1**,** "w"**)**
77. FileWrite**.**write**(**DNA\_Seq**)**
78. FileWrite**.**close**()**
79. # get fragment numbers
80. frag\_num **=** **(**len**(**DNA\_Seq**)** **\*** **10)** **/** reads
81. # genete the uniquie start points from DNA sequence based on length from 0 to length - 400
82. unum **=** random**.**sample**(**range**(0,** len**(**DNA\_Seq**)** **-** reads**),** int**(**frag\_num**))**
83. #sorted the array
84. #unum.sort()
85. # Open file to write
86. File\_Write **=** open**(**OutputFile**,** "w"**)**
87. #print(DNA\_Seq)
88. #print(unum)
89. #print(len(unum))
90. # based on the start points, get the fragment sequence and save to output file
91. **for** i **in** range**(**len**(**unum**)):**
92. frag **=** ''
93. frag\_end **=** unum**[**i**]** **+** reads
94. frag **=** DNA\_Seq**[**unum**[**i**]:**frag\_end**]**
95. File\_Write**.**write**(**frag **+** '\n'**)**
96. # close file
97. File\_Write**.**close**()**
98. **return**
99. **def** JoinFile**(**Input1**,** input2**,** output**):**
100. filenames **=** **[**Input1**,** input2**]**
101. with open**(**output**,** 'w'**)** as outfile**:**
102. **for** fname **in** filenames**:**
103. with open**(**fname**)** as infile**:**
104. **for** line **in** infile**:**
105. outfile**.**write**(**line**)**
106. **return**
107. #
108. # 3.Create overlap graphs between all fragments
109. #
110. **def** CreateOverlapGraph**(**InputFile**,** OutputFile**):**
112. OverlayValue **=** **0**
113. FileRead **=** open**(**InputFile**,** "r"**)**
114. FileWrite **=** open**(**OutputFile**,** "w"**)**
115. # Read all lines and store to array
116. frag **=** FileRead**.**readlines**()**
117. #print(frag[1][1:10])
118. # Get numbers of fragment
119. frag\_num **=** len**(**frag**)**
120. #print(frag\_num)
121. #Graphs = [{} for x in range(frag\_num)]
122. #print(Graphs)
123. # This first loop is get the first DNA Sequence
124. **for** i **in** range**(0,** frag\_num**):**
125. # Create dict object
126. overlap **=** **{}**
127. # write the line to file (ex 0:)
128. #FileWrite.write(str(i) + ": ")
129. # this loop get the second DNA Sequence
130. **for** j **in** range**(0,** frag\_num**):**
131. # if they are same line, ignore and continues
132. **if** **(**i **==** j**):**
133. **continue**
134. overlap\_index **=** **0**
135. # this loop is use to compare the first and seoncd DNA Sequences
136. # this run from position 0 to 399 (Mean from 1 to 400)
137. **for** k **in** range**(0,** reads **-** **1):**
138. # check if the suffix and preffix is same
139. **if(**frag**[**i**][-**k**-2:-1]** **==** frag**[**j**][0:**k**+1]):**
140. overlap\_index **=** k
142. **if(**overlap\_index **>** **0):**
143. # add to dict object
144. overlap**[**j**]** **=** overlap\_index
145. #print(overlap)
146. json**.**dump**(**overlap**,** FileWrite**)**
147. FileWrite**.**write**(**'\n'**)**
148. #print(Graphs)
149. FileRead**.**close**()**
150. FileWrite**.**close**()**
151. **return**
152. #
153. #    4.Get largest overlap graph
154. #
155. **def** CreateLargestOverlapGraph**(**InputFile**,** OutputFile**):**
156. FileRead **=** open**(**InputFile**,** "r"**)**
157. FileWrite **=** open**(**OutputFile**,** "w"**)**
158. # Read all lines and store to array
159. Graphs\_X **=** FileRead**.**readlines**()**
160. #print(Graphs\_X[0])
161. graph\_num **=** len**(**Graphs\_X**)**
162. Graphs **=** **[{}** **for** i **in** range**(**graph\_num**)]**
163. #Convert Graphs\_X to dict and store to Graphs
164. **for** i **in** range**(0,** graph\_num**):**
165. Graphs**[**i**]** **=** json**.**loads**(**Graphs\_X**[**i**])**
166. graph\_l **=** **[{}** **for** i **in** range**(**graph\_num**)]**
167. **for** i **in** range**(0,** graph\_num**):**
168. #Keep tract other
169. #FileWrite.write(str(i) + " ")
170. overlap\_value **=** **0**
171. overlap\_index **=** **0**
172. **for** key**,**value **in** Graphs**[**i**].**items**():**
173. # get max value from graph
174. **if** **(**overlap\_value **<** value**):**
175. overlap\_value **=** value
176. overlap\_index **=** key
177. graph\_l**[**i**][**overlap\_index**]** **=** overlap\_value
178. json**.**dump**(**graph\_l**[**i**],** FileWrite**)**
179. FileWrite**.**write**(**'\n'**)**
180. FileRead**.**close**()**
181. FileWrite**.**close**()**
182. **return**
183. #
184. #    5.Create path from the largest overlap graph
185. #
186. **def** CreatePath**(**InputFile**,** OutputFile**):**
187. FileRead **=** open**(**InputFile**,** "r"**)**
188. FileWrite **=** open**(**OutputFile**,** "w"**)**
190. graph\_X **=** FileRead**.**readlines**()**
191. Graph\_num **=** len**(**graph\_X**)**
192. graph\_l **=** **[{}** **for** i **in** range**(**Graph\_num**)]**
193. #Convert Graphs\_X to dict and store to Graphs
194. **for** i **in** range**(0,** Graph\_num**):**
195. graph\_l**[**i**]** **=** json**.**loads**(**graph\_X**[**i**])**
196. #Creat object path with subclass OrderedDict()
197. Graph\_Path **=** **[**OrderedDict**()** **for** i **in** range**(**Graph\_num**)]**
198. #print(Graph\_Path)
200. **for** i **in** range**(**Graph\_num**):**
201. path **=** **[0** **for** j **in** range**(**Graph\_num**)]**
202. path**[**i**]** **=** **1**
203. lap\_weight **=** **0**
204. next\_vertex **=** i
205. **while** True**:**
206. current\_vertex **=** next\_vertex
207. graph **=** graph\_l**[**current\_vertex**]**
208. next\_vertex **=** int**(**list**(**graph**.**keys**())[0])**
209. lap\_weight **=** list**(**graph**.**values**())[0]**
210. #print(graph, next\_vertex, lap\_weight)
211. **if** **(**path**[**next\_vertex**]** **==** **0):**
212. Graph\_Path**[**i**][**current\_vertex**]** **=** lap\_weight
213. #set path to 1 so it wont read it again
214. path**[**next\_vertex**]** **=** **1**
215. lap\_weight **=** **0**
216. **else:**
217. **break**
218. **del** path
219. json**.**dump**(**Graph\_Path**[**i**],** FileWrite**)**
220. FileWrite**.**write**(**'\n'**)**
221. #print(Graph\_Path[0])
222. FileRead**.**close**()**
223. FileWrite**.**close**()**
224. **return**
225. #
226. # 6.create new DNA based on overlap path
227. #
228. **def** CreateNewDNASequence**(**InputFile1**,** InputFile2**,** OutputFile**):**
229. FileRead **=** open**(**InputFile1**,** "r"**)**
230. FileWrite **=** open**(**OutputFile**,** "w"**)**
231. FileRead1 **=** open**(**InputFile2**,** "r"**)**
232. Fragment **=** FileRead1**.**readlines**()**
233. #print(Fragment[0])
235. graph\_X **=** FileRead**.**readlines**()**
236. Graph\_num **=** len**(**graph\_X**)**
237. #print(graph\_X[0])
238. Graph\_Path **=** **[{}** **for** i **in** range**(**Graph\_num**)]**
239. #Convert Graphs\_X to dict and store to Graphs
240. **for** i **in** range**(0,** Graph\_num**):**
241. Graph\_Path**[**i**]** **=** json**.**loads**(**graph\_X**[**i**],** object\_pairs\_hook**=**OrderedDict**)**
242. #print(graph\_X[0])
243. newDNA\_num **=** len**(**Graph\_Path**)**
244. newDNA\_seq **=** **[**'' **for** i **in** range**(**newDNA\_num**)]**
245. #print(newDNA\_num, newDNA\_seq)
246. next\_seq **=** **0**
247. **for** path **in** Graph\_Path**:**
248. i **=** int**(**list**(**path**)[0])**
249. #print("index:", i)
250. overlap **=** int**(**list**(**path**.**values**())[0])**
251. #print("overlap:", overlap)
252. newDNA\_seq**[**next\_seq**]** **=** Fragment**[**i**].**strip**(**'\n'**)**
253. #print(newDNA\_seq[next\_seq])
254. **for** i**,** nextlap **in** list**(**path**.**items**())[1:]:**
255. newDNA\_seq**[**next\_seq**]** **+=** Fragment**[**int**(**i**)][**overlap**+1:].**strip**(**'\n'**)**
256. overlap **=** nextlap
257. json**.**dump**(**newDNA\_seq**[**next\_seq**],** FileWrite**)**
258. FileWrite**.**write**(**'\n'**)**
259. next\_seq **+=** **1**
260. FileRead**.**close**()**
261. FileRead1**.**close**()**
262. FileWrite**.**close**()**
263. **return**
264. **def** CreateDNASequence**(**InputFile**,** OutputFile**):**
265. FileRead **=** open**(**InputFile**,** "r"**)**
266. FileWrite **=** open**(**OutputFile**,** "w"**)**
267. newDNA\_logest\_len **=** **0**
268. Sequences **=** FileRead**.**readlines**()**
269. # Remove " sign from read string
270. **for** i **in** range**(**len**(**Sequences**)):**
271. Sequences**[**i**]** **=** Sequences**[**i**].**replace**(**'"'**,** ''**)**
272. Sequences**[**i**]** **=** Sequences**[**i**].**replace**(**'\n'**,** ''**)**
273. # Calculate the longest DNA sequences
274. **for** i **in** Sequences**:**
275. **if** **(**newDNA\_logest\_len **<** len**(**i**)):**
276. newDNA\_logest\_len **=** len**(**i**)**
277. #seq = i
278. #print(Sequences[0])
279. # The new DNA sequence will have the length of the longest DNA sequences
280. # Calculate column by column to fine get the new letter
281. newDNA\_consensus **=** ''
282. newDNA\_consensus\_len **=** len**(**Sequences**)**
283. **for** i **in** range**(**newDNA\_logest\_len**):**
284. consensus **=** **{**"A"**:** **0,** "C"**:** **0,** "G"**:** **0,** "T"**:** **0}**
285. # Get each string in array of string
286. **for** new\_seq **in** Sequences**:**
287. **if** **(**len**(**new\_seq**)** **>** i**):**
288. # increase letter counting
289. consensus**[**new\_seq**[**i**]]** **+=** **1**
290. newDNA\_consensus **+=** max**(**consensus**,** key**=**consensus**.**get**)**
291. FileWrite**.**write**(**newDNA\_consensus**)**
292. FileRead**.**close**()**
293. FileWrite**.**close**()**
294. **return**
295. **def** CompareDNAs**(**InputFile1**,** InputFile2**,** Org**,** New**):**
296. FileRead1 **=** open**(**InputFile1**,** "r"**)**
297. FileRead2 **=** open**(**InputFile2**,** "r"**)**
298. # Get DNA sequence
299. New\_DNA **=** FileRead1**.**readlines**()[0]**
300. Org\_DNA **=** FileRead2**.**readlines**()[0]**
301. # Get lenght
302. New\_DNA\_len **=** len**(**New\_DNA**)**
303. Org\_DNA\_len **=** len**(**Org\_DNA**)**
305. most\_ol **=** **0**
306. **for** i **in** range**(**Org\_DNA\_len**):**
307. k **=** i
308. overlap **=** **0**
309. **for** j **in** range**(**New\_DNA\_len**):**
310. **if** **(**k **<** Org\_DNA\_len**):**
311. **if** **(**Org\_DNA**[**k**]** **==** New\_DNA**[**j**]):**
312. overlap **+=** **1**
313. k **+=** **1**
314. **else:**
315. **break**
316. **if** **(**overlap **>** most\_ol**):**
317. most\_ol **=** overlap
318. **print(**"Org DNA"**,** Org**,** "vs New OLC DNA"**,** New**)**
319. **print(**"Org DNA"**,** Org**,** "Length ="**,** Org\_DNA\_len**)**
320. **print(**"New DNA"**,** New**,** "Length ="**,** New\_DNA\_len**)**
321. **print(**"Number's Overlap ="**,** most\_ol**)**
322. **print(**''**)**
323. FileRead1**.**close**()**
324. FileRead2**.**close**()**
325. **return**
326. **def** main**():**
327. #========================================
328. #    1. Remove the N and other character
329. #========================================
330. CleanUpOriginalFileToNewFile**(**DNA\_File1\_FASTA**,** DNA\_File1\_RmN**)**
331. CleanUpOriginalFileToNewFile**(**DNA\_File2\_FASTA**,** DNA\_File2\_RmN**)**
332. JoinFile**(**DNA\_File1\_RmN**,** DNA\_File2\_RmN**,** DNA\_File3\_RmN**)**
333. **print(**"1. Clean Up - Done"**)**
334. #========================================
335. #    2. From short file (200 lines) make a fragment file (Manually made)
336. #========================================
337. CreateFragFile**(**DNA\_File1\_RmN**,** DNA\_File1\_Frag**,** DNA\_File1\_Orig**)**
338. CreateFragFile**(**DNA\_File2\_RmN**,** DNA\_File2\_Frag**,** DNA\_File2\_Orig**)**
339. CreateFragFile**(**DNA\_File3\_RmN**,** DNA\_File3\_Frag**,** DNA\_File3\_Orig**)**
340. **print(**"2. Get Fragment - Done"**)**
341. #========================================
342. #    3. Calculate the overlap and save to overlap graph from reads
343. #========================================
344. CreateOverlapGraph**(**DNA\_File1\_Frag**,** DNA\_File1\_Graph**)**
345. CreateOverlapGraph**(**DNA\_File2\_Frag**,** DNA\_File2\_Graph**)**
346. CreateOverlapGraph**(**DNA\_File3\_Frag**,** DNA\_File3\_Graph**)**
347. **print(**"3. Calculate Overlap - Done"**)**
348. #========================================
349. #    4. Get the largest overlap graph from the graph\_short file
350. #========================================
351. CreateLargestOverlapGraph**(**DNA\_File1\_Graph**,** DNA\_File1\_Graph\_Largest**)**
352. CreateLargestOverlapGraph**(**DNA\_File2\_Graph**,** DNA\_File2\_Graph\_Largest**)**
353. CreateLargestOverlapGraph**(**DNA\_File3\_Graph**,** DNA\_File3\_Graph\_Largest**)**
354. **print(**"4. Get Largest Overlap Graph - Done"**)**
355. #========================================
356. #    5. Create path from the largest overlap graph
357. #========================================
358. CreatePath**(**DNA\_File1\_Graph\_Largest**,** DNA\_File1\_Path**)**
359. CreatePath**(**DNA\_File2\_Graph\_Largest**,** DNA\_File2\_Path**)**
360. CreatePath**(**DNA\_File3\_Graph\_Largest**,** DNA\_File3\_Path**)**
361. **print(**"5. Get Path - Done"**)**
362. #========================================
363. #    6. Calculate new DNA sequences from overlap path
364. #========================================
365. CreateNewDNASequence**(**DNA\_File1\_Path**,** DNA\_File1\_Frag**,** DNA\_File1\_NewDNA\_Seqs**)**
366. CreateNewDNASequence**(**DNA\_File2\_Path**,** DNA\_File2\_Frag**,** DNA\_File2\_NewDNA\_Seqs**)**
367. CreateNewDNASequence**(**DNA\_File3\_Path**,** DNA\_File3\_Frag**,** DNA\_File3\_NewDNA\_Seqs**)**
368. **print(**"6. Get New DNA Sequences - Done"**)**
369. #========================================
370. #    7. Create one new DNA sequence from new DNA sequences
371. #========================================
372. CreateDNASequence**(**DNA\_File1\_NewDNA\_Seqs**,** DNA\_File1\_NewDNA**)**
373. CreateDNASequence**(**DNA\_File2\_NewDNA\_Seqs**,** DNA\_File2\_NewDNA**)**
374. CreateDNASequence**(**DNA\_File3\_NewDNA\_Seqs**,** DNA\_File3\_NewDNA**)**
375. **print(**"7. Create one new DNA sequence - Done"**)**
376. #========================================
377. #    8. Compare new and orignal DNA
378. #========================================
379. CompareDNAs**(**DNA\_File1\_NewDNA**,** DNA\_File1\_Orig**,** "1"**,** "1"**)**
380. CompareDNAs**(**DNA\_File2\_NewDNA**,** DNA\_File2\_Orig**,** "2"**,** "2"**)**
381. CompareDNAs**(**DNA\_File1\_NewDNA**,** DNA\_File3\_Orig**,** "1"**,** "3"**)**
382. CompareDNAs**(**DNA\_File2\_NewDNA**,** DNA\_File3\_Orig**,** "2"**,** "3"**)**
383. **print(**"8. Compare two DNAs - Done"**)**
384. main**()**

# References

Gifford, D. K. (n.d.). Genome Assembly. *Foundations of Computational Systems Biology*. Retrieved from https://ocw.mit.edu/courses/biology/7-91j-foundations-of-computational-and-systems-biology-spring-2014/lecture-slides/MIT7\_91JS14\_Lecture6.pdf

Graph Algorithms in Bioinformatics . (n.d.). *An Introduction to Bioinformatics Algorithms* . Retrieved from http://genetics.cs.ucla.edu/cs124/lecture/Lecture8.pdf

Huson, D. (2010, February 4). Bioinformatics I. Retrieved from http://kam.mff.cuni.cz/~horacek/projekty/shotgun/materialy/sequence\_assembly.pdf

Langmead, B. (n.d.). De Bruijn Graph assembly. Retrieved from http://www.cs.jhu.edu/~langmea/resources/lecture\_notes/assembly\_dbg.pdf

Langmead, B. (n.d.). Overlap Layout Consensus assembly. Retrieved from http://www.cs.jhu.edu/~langmea/resources/lecture\_notes/assembly\_olc.pdf

Lederman, R. (2012, December 18). Building approximate overlap graphs for DNA assembly using random-permutations-based search. New Haven, CT, 06511. Retrieved from http://www.cs.yale.edu/publications/techreports/tr1470.pdf

Sequence assembly. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Sequence\_assembly