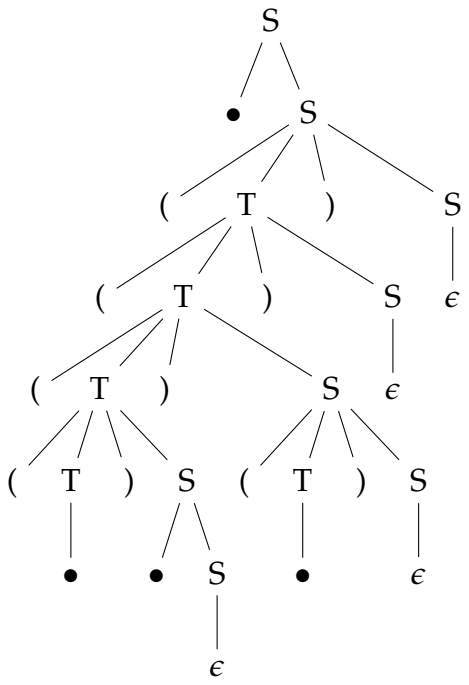
(1)



(2) One way of taking a look at it is the Sharing of Base-Pairs. If the Consensus Secondary Structure has a Pseudo-Knot, that would mean that the Frequency of the Crossing Base-Pairs is higher than 0,5. More than half of the Sample Set should have a Shared Base-Pair. The problem is that we began with a Sample Set that did not have a Pseudo-Knot. □

Furthermore, another way of looking at it is basing off the fact that multiple papers say that it is impossible (without extra symbols) to show a Pseudo-Knot in Dot-Bracket Notation [1]. We are given several Structures without Pseudo-Knots, and the only way to show a Consensus is to display what is shared among them. Because the samples themselves do not have Pseudo-Knots, there will not be any Pseudo-Knots in the Consensus Secondary Structure.

(3c) Below is a table of the collected data. For this question, I recognised that there could be integer-integer division when calculating the frequency. I then tried to use NumPy's `longdouble` type, and I got more or less the same thing. I finally then tried with integer-integer division, and the result was always evaluated to the same number. Thus proving my intuition. Lastly I also had to update the template. I was using the 2017 template, which didn't account for `lbox`. I spent a week trying to figure out why my errors were so high.

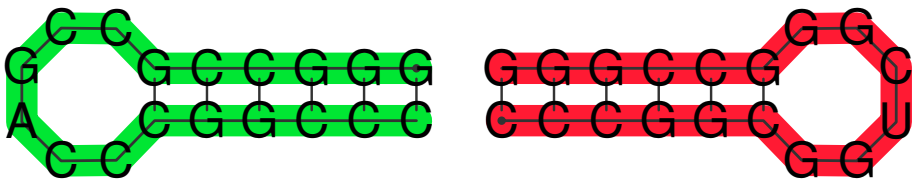
[1] `https://www.tbi.univie.ac.at/RNA/ViennaRNA/doc/html/rna_structure_notations.html`

	k=10	k=50	k=100	k=1000	k=10'000
1	1.5375059	0.9211347	0.3573519	0.1889793	0.0523643
2	1.3642018	0.3793504	0.4766675	0.1290219	0.0516527
3	1.0909705	0.3872036	0.4766295	0.1536227	0.0502646
4	1.4071216	0.5466988	0.6668144	0.1428715	0.0467222
5	1.0497515	0.6041981	0.5532631	0.2284039	0.0401635
6	1.8845581	0.4874197	0.3610638	0.1588597	0.0376211
7	1.1995103	0.7100801	0.3721780	0.1273484	0.0516988
8	1.1039671	0.7092426	0.4111781	0.1221495	0.0439793
9	1.7656729	0.6126232	0.6891705	0.1401970	0.0437631
10	1.7252548	0.6233073	0.4511786	0.0996786	0.0341162
MEAN	1.4128515	0.5981259	0.4815495	0.1491133	0.0452346
STDEV	0.3053044	0.1623946	0.1203693	0.0367549	0.0064323

As k increases, the error decreases. When $k = 200'000$, $error = 0.01$. Therefore, $k > 200'000$ in order for $error < 0.01$

(4) `GGGCCGCCGACCCGGCCC&CCCGGCGGUCGGGCCGGG`

`RNAcofold < omega.fasta` gives me the following output.



```
GGGCCGCCGACCCGGCCC
CCCGGCGGUCGGGCCGGG
```

`RNAup < omega.txt` gives me the following output.

```
# RNAup --include_both
# 18
# GGGCCGCCGACCCGGCCC
# 18
# CCCGGCGGUCGGGCCGGG
#     pos      u4S       dG
       1       NA    0.000
       2       NA   -0.134
       3       NA   -0.134
       4    6.167   -0.134
       5    6.166   -0.134
       6    8.799   -1.614
       7    7.611  -11.998
       8    6.355  -11.998
       9    1.739  -11.998
      10    0.000  -11.998
      11    0.000  -11.998
      12    0.001  -11.998
      13    1.739   -7.290
      14    6.514   -2.117
      15    9.173   -2.117
      16    9.143   -2.117
      17    7.141   -2.117
      18    7.141    0.000
#     pos      u4S
       1       NA
       2       NA
       3       NA
       4    8.920
       5    8.931
       6   10.880
       7   10.045
       8    7.422
       9    4.792
      10    0.000
      11    0.000
      12    0.001
      13    4.799
      14    8.687
      15   11.014
      16    9.376
```

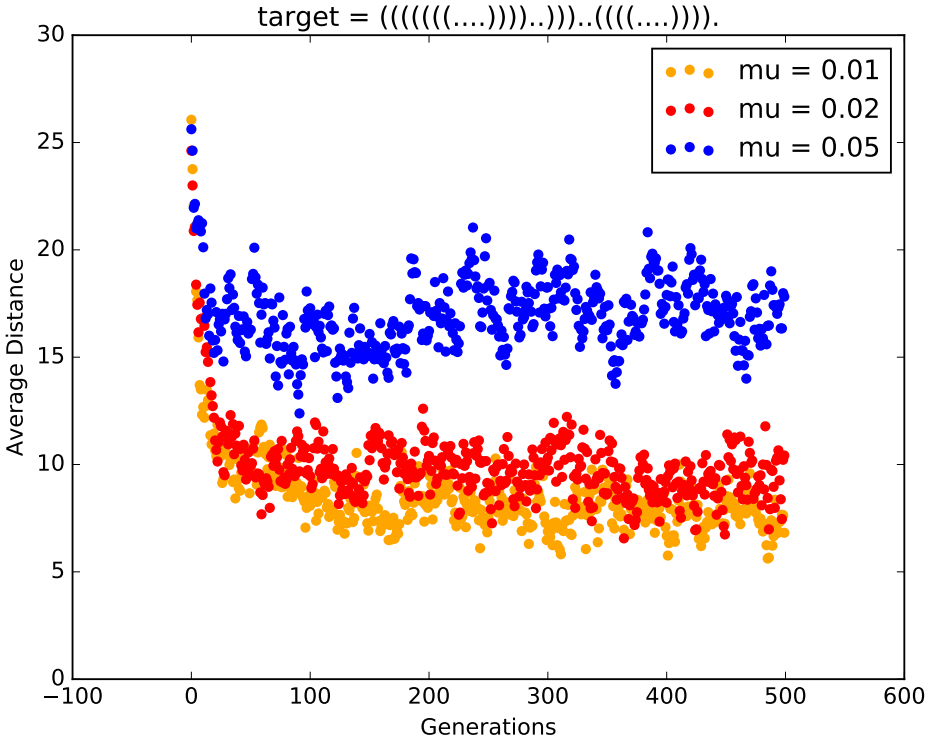
```
17      7.176
18      7.175
```

The predictions are different because they run different algorithms. Basically, the difference in calculation arises due to the internal computations. `RNAup` looks for the best interaction site based off the calculations of the opening energies and the interaction energies. `RNAcofold` establishes a common secondary structure using modified energies.

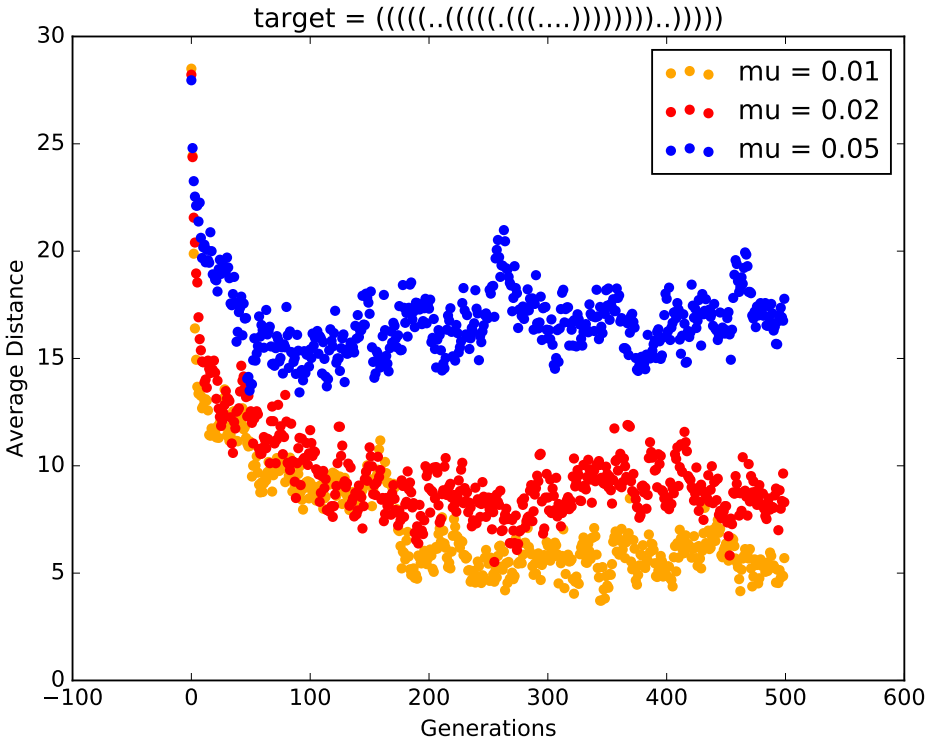
Obviously each algorithm has its advantages and limitations. `RNAcofold` is used more towards the investigation of the concentration dependency of dimerization. `RNAup` is used more towards the investigations of the binding of regulatory RNA molecules with their target RNAs[2].

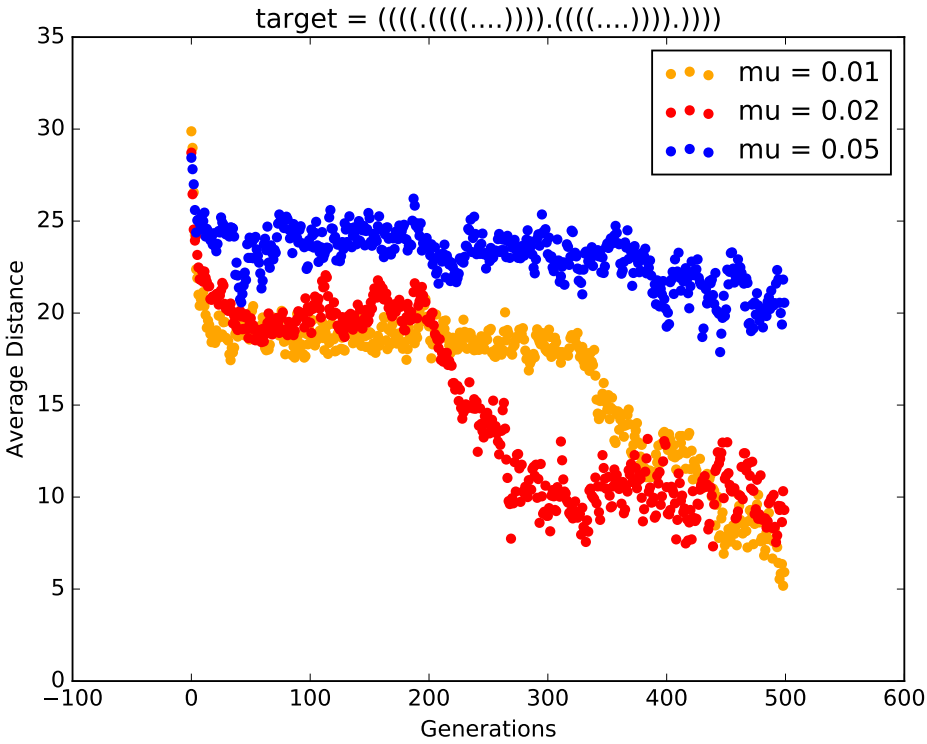
(5) Due to the election in October 2019, I have decided to make my colours based off the Party Colours. IE: Red is Liberal, Blue is Conservative and Orange is NDP.

This question actually took a long time to run. Each target structure took almost 30 minutes. Therefore, I spent 90 minutes or 1,5 hours staring at a computer screen trying to entertain myself with other websites. My Macbook Air only has 4GB of RAM, so I decided to run and gather data on the Trottier Machines, which to my knowledge have around 12 GB of RAM.



[2] `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3319429/`


target = ((((..((((((.(((....)))))))).))))))
mu = 0.01
mu = 0.02
mu = 0.05
Average Distance
Generations
0
5
10
15
20
25
30
−100
0
100
200
300
400
500
600


target = ((((.((((....)))).((((....)))).))))
mu = 0.01
mu = 0.02
mu = 0.05
Average Distance
Generations
0
5
10
15
20
25
30
35
−100
0
100
200
300
400
500
600