Andreas, Christian, Lærke

Status of work

For this project we ended up using the program “rfdist” from the previous project, and ended up completing the program “discountNJ.py”, as well using the two different tree building methods (QuickTree and RapidNJ). This all worked as expected and we achieved satisfactory results (including getting the correct output from “example_slide4.phy”). In general, the project has gone ahead smoothly and without too many hiccups.

Implementation of NJ

Firstly we decided on implementing in Python, as none of us are confident in our proficiency with other languages.
We implemented the version by Saitou and Nei, presented to us during the lectures, although without keeping track of the branch length as these wouldn’t be necessary for the rf dist. Furthermore, instead of filling the N matrix (as described by S&N) and then searching for the minimal entry, we just kept track of the best candidate while calculating the matrix. (we also realized that by doing this, we had no reason to keep the N matrix, so that can be ignored if space is a concern).
We implemented the distance matrix using Numpy Arrays, hoping that this would speed up calculations of row/column sums.
The running time turned out to be rather unimpressive, so we thought about some optimizations we could try, but didn’t have time for. Right now we remove two columns from the array, which requires a lot of updating in the array, this could be reduced to one, if we decided to reuse one of the outgoing columns for the new. Furthermore, for the implementation of the tree, we used python lists to contain the children of a Node, Which with large input size might also slow the running time. We considered trying a linked list implementation for easy removal of elements from the tree, but did not end up implementing it.
The only dependency is Numpy which is used as described above. furthermore we use the system and dataclass modules.
The program takes one positional argument (sys argv 1) which is a phylip format distance matrix, as described in the project description. example of program call:
python discountNJ.py matrix.phy

Description of machine

Machine specs:
Processor Intel i5-7200u @ 2.71GHz
8GB Ram
All programs were run on Ubuntu (20.04.4 LTS) using windows subsystem for linux

Time measurement were automated in python, by using the timeit module to log system times, and subprocess module to run the programs and assert that they actually completed. All the output was piped to .nwk files, and used to calculate rf distances using our own program.

Andreas, Christian, Lærke

Results

Nr of seqs	*Running time QuickTree (s)*	*Running time RapidNJ (s)*	*Running time DiscountNJ (s)*	*QuickTree/DiscountNJ*	*RapidNJ/Discount NJ*	*RF dist Quicktree/DiscountNJ*	*Rfdist RapidNJ/DiscountNJ*	*Rfdist RapidNJ/QuickTree*
89	*0.096*	*0.043*	*2*	*0.048*	*0.0217*	*9*	*21*	*18*
214	*0.224*	*0.076*	*8*	*0.028*	*0.0094*	*16*	*35*	*38*
304	*0.364*	*0.110*	*21*	*0.017*	*0.0052*	*8*	*56*	*59*
401	*0.634*	*0.157*	*57*	*0.011*	*0.0027*	*23*	*75*	*94*
494	*0.732*	*0.190*	*95*	*0.007*	*0.0020*	*204*	*529*	*530*
608	*0.977*	*0.309*	*172*	*0.0057*	*0.0018*	*13*	*10*	*17*
777	*1.351*	*0.430*	*243*	*0.0056*	*0.0018*	*126*	*396*	*431*
877	*2.067*	*0.592*	*384*	*0.0054*	*0.0015*	*22*	*27*	*43*
1347	*6.864*	*1.429*	*1136*	*0.0060*	*0.0012*	*1*	*3*	*1*
1493	*7.187*	*1.480*	*1483*	*0.0048*	*0.0009*	*19*	*72*	*70*
1560	*8.271*	*1.662*	*1604*	*0.0051*	*0.0010*	*44*	*119*	*128*
1689	*10.287*	*1.998*	*2019*	*0.0051*	*0.0009*	*24*	*82*	*84*
1756	*10.470*	*2.191*	*2302*	*0.0045*	*0.0009*	*14*	*54*	*51*
1849	*13.160*	*2.280*	*2753*	*0.0048*	*0.0008*	*57*	*202*	*213*

Andreas, Christian, Lærke

Graph of running time in seconds as a function of number of species for the 3 programs.