GP_lab3

March 19, 2020

In [1]:
```
#Algorytm Felsensteina
import math
import numpy as np
```

In [2]:
```
def pr(nucleotide1: str, nucleotide2:str , time: float, alpha = .25):

    match = nucleotide1 == nucleotide2

    probability = (1/4 + (3/4 if match else -1/4) * math.exp(-4 * alpha * time))

    return(probability)
```

In [3]:
```
def isLeaf(k: int, structure: list):
    return(structure[k] == ())
```

In [4]:
```
def prL(k: int, a: str, u: int, x: list, t: int, s: list, prL_number = 0):
    ACTG = ['A', 'C', 'T', 'G']

    if isLeaf(k,s):
        if x[k][u] == a:
            prL_number = 1

        else:
            prL_number = 0
    else:
        sons = [i for i in s[k]]
        for b in ACTG:
            for c in ACTG:
                prL_number += (pr(a,b,t[sons[0]]) * \
                               prL(sons[0], b, u, x, t, s) * \
                               pr(a,c,t[sons[1]]) * \
                               prL(sons[1], c, u, x, t, s))
    return(prL_number)
```

In [5]:
```
def prLBis(u, x, t, s):

    q = 0.25
    tree_probability = 0
```

```
            ACTG = ['A', 'C', 'T', 'G']

            for n in ACTG:
                tree_probability += prL(len(s) - 1, n, u, x, t, s) * q

            return(tree_probability)
```

In [6]:
```
def treeLogLikelihood(x, t, s):

    result = 0

    for u in range(len(x[0])):
        result += math.log(prLBis(u,x,t,s))

    return(result)
```

In [7]:
```
def testuj():
    """
    Procedura testowa dla alg. Felsensteina.
    """
    x = [ "AACACA", "AACGCA", "ATTACA", "AACGTG" ]
    leaf = ()
    s = [leaf]*4 + [ (0,1), (2,3), (4,5) ]
    t = [ 1.0, 2.5, 1.0, 1.0, 3.0, 4.5, -1.0 ]
    print(treeLogLikelihood(x, t, s)) # -33.25
```

In [8]:
```
testuj()
```

```
-33.253205712534225
```

In [ ]: