MBI Homework 3 for CS students

Autor: Marián Kravec

a)

To compute structure with maximal number of base pairs without any not nested cases we will use slightly modified Nussinov algorithm.

```
def nussin(seq):
    n = len(seq)
    A = np.zeros((n,n))
    B = np.zeros((n,n))
    for length in range(1, n):
        for i in range(n):
            j = i + length
            if j < n:
                pos_val = 0
                A[i, j], B[i, j] = max([(A[i+1, j], 1),
                                        (A[i, j-1], 2),
                                        (A[i+1, j-1]+pair(seq[i], seq[j]), 3)
                                        ], key= lambda x: x[0])
    return(A[0, n-1], B)
```

This algorithm is filling out table A (starting from diagonal and getting closer to upper-top corner) where $A[i, j]$ is number maximal number of nested base pair of sub-sequence starting from position i and ending on position j (final solution is in cell $A[0, n]$ where n is length of sequence), it's a bit simple than standard Nussinov algorithm because it does not take into account situation where branching of secondary structure create more base pairs, so situation where we are looking for position k between positions i and j such that $A[i, k] + A[k + 1, j]$ is bigger than any other solution. We omit this rules because breaching creates not nested pairs.

So to compute $A[i, j]$ our algorithm choose maximal value out of three possibilities: $A[i + 1, j]$, $A[i, j - 1]$ or $A[i + 1, j - 1] + pair(x_i, x_j)$.

- $A[i + 1, j]$ - first base of sub-sequence in unpaired and rest is optimally paired
- $A[i + 1, j]$ - last base of sub-sequence in unpaired and rest is optimally paired
- $A[i + 1, j - 1] + pair(x_i, x_j)$ - interpretation of this depends on value of $pair(x_i, x_j)$ if x_i and x_j creates pair then this is situation where we take optimal pairing of bases between them plus their own, if they do not create pair then this is situation where we take optimal pairing ob bases between them and first and last base is unpaired

This algorithm compute half of values of matrix with size $n \times n$ and for each position if finds maximum of constant amount of possibilities (only 3), so asymptotic running time of this algorithm is $O(n^2)$.

b)

Similarly to algorithm from part a), our stochastic context-free grammar needs only one nonterminal which we will call S and to have only three types of rules (three types does not mean three rules), and that is rules to create unpaired bases before nonterminal, rules to create unpaired bases after nonterminal and rules to create pairs (and lastly there will be rule to terminate sequence generation):

$$
\begin{aligned}
S \rightarrow & aS|uS|cS|gS| \\
& Sa|Su|Sc|Sg| \\
& aSu|uSa|cSg|gSc| \\
& \epsilon
\end{aligned}
$$