Nussinov's algorithm is a base pair maximization algorithm that predicts the RNA secondary structure by utilizing dynamic programming. The dynamic programming breaks down a problem into small subsequences, works its way and expand to tackle larger subsequences. In order to satisfy the conditions of this algorithm, the match pairs are nested non-intersecting. The more base pairs you have, the more stable RNA structure you get. The basic Nussinov's algorithm does not count the real experimental information into the folding model. So, the implementation here includes a constraint from the experiment. The constraint is given as a sequence of N nucleotides with a vector of N elements containing values between 0 and 1, where the value close to 1 is more towards unpairing. There are different ways to utilize experimental data to enhance the biological relevant structure.

The first one is to find the value in each nucleotide position and determined whether or not that position can pair. Based on the information from the experiment, when the value is close to 1, it is more unpaired. To find the value that is close to one from the interval between 0 and 1 can be tricky, so I decided to define the pair or unpair by using the midpoint of the interval, then add this constraint to the Nussinov by specifying each nucleotide that contains value less than 0.5 as pairable nucleotides, as in the formular c[i] and c[j] <= 0.5, while any nucleotide that contains value greater than the criteria is unpairable. For the recursion, c[i] and c[j] <= 0.5 must be added as a condition check before calculating the score of the matching pair. So, it has to satisfy a constrained value before adding a score to the matrix.

The second approach is to consider c[i] and c[j] as a pair rather than as individuals. In the pair, c[i] and c[j] should be weighted equally in the pair. The condition can be defined as $\frac{(C_i+C_j)}{2} \leq \mathbf{0.5}$

The strength of the first approach is that we can see the effect of constraint directly because each matching pair receives one point in the scoring scheme for matching pairs. The main disadvantage of this approach is losing the marginal points (potential pairs). We do not know the exact value of the cutting point in term of being pairable or unpairable. For example, if we have a pair of c[i] = 0.25 and c[j] = 0.53, for the first approach, this will be considered unpairable, even if the values are very close. The second approach is more compromise. It does not cut those marginal values out of matching pairs; after we take the average of c[i] = 0.25 and c[j] = 0.53, it is less than 0.5.

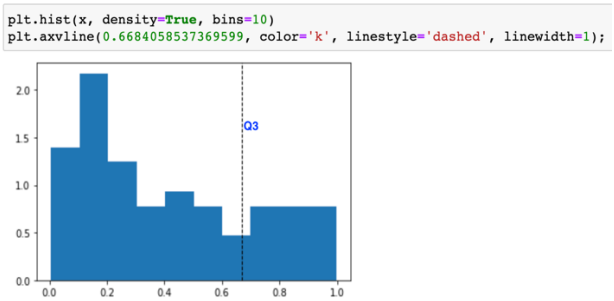

Figure 1 Distribution plot of constraint value of 1st sequence

According to the distribution plot of the constraint value of the first sequence, as shown in figure1, the mean = 0.41, the median = 0.35, and third quantile is at 0.67, which indicates that the majority of the nucleotides are pairable, and it is a very tight constraint to consider values above 0.5 as unpair able, so the second approach would be safe to mitigate the effect of overkilling those marginal pairs.

The implementation contains 6 functions; **def init_matrix(), def fill_matrix(), def backtracking(), def identifyBP(seq), def count_BP(list_1, list_2) and def hamming().** This implementation is a modified version of the basic Nussinov's algorithm exercise with no pseudoknots. The main idea is to find the maximum base pair, but this version includes a constraint as an extra criterion for the matching pair. Besides fulfilling the canonical Watson–Crick base, the matching pairs need to be checked for constraint value, if the constraint value indicates they are unpairable, there will be no score addition to the matrix. In the recursion step, we aim to find the maximum value from the down, left, diagonal, and the bifurcation term.
The recursion step can be explained as follows: Sequence:($x_i, \ldots . x_j$) Constraint: ($C_i, \ldots . C_j$)

$$\mathbf{E(i,\ j)} = \begin{cases} \boldsymbol{E(i, j-1)\ \ from\ the\ left} \\ \boldsymbol{E(i+1, j)\ from\ down} \\ \boldsymbol{E(i+1, j-1) + s(i,j) : add\ score\ s(i,j)\ if\ and\ only\ if\ \frac{(C_i + C_j)}{2} \leq 0.5} \\ \boldsymbol{max}_{\ i<k<j-1}\{\boldsymbol{E(i,k) + E(k+1, j)}\} \end{cases}$$

$$\mathbf{s(i,\ j) = max}\begin{cases} \boldsymbol{1\ if\ x[i] and\ x[j]\ pair} \\ \boldsymbol{0\ otherwise} \end{cases}$$

Backtracking step is an implementation of pseudo-code given in lecture (As in Durbin et al. 1998) with a bit of modification, else if M(i+1, j-1) + d[i, j] **and** $\frac{(C_i + C_j)}{2} \leq 0.5$ **and M(i+1, j-1) + s(i,j)** = M(i, j) then record base, push(i+1,j-1) to stack.

The last part of this implementation contains functions that compare base pair and Hammer distances, which I implemented in the last exercise session.

Result:
Seq1 with no constraint
AUCUAUAUAGUAUAAAAGUAUAUUUGACUUCCAAUCAUAAGGUCUAUUAAUUAAUAGUAUAG
AUA 65
((((((((((((((.(...)))))))((((.((..(((.((...))..)))))))))))))).))))))).
Score: 25
Seq1 with constraint
AUCUAUAUAGUAUAAAAGUAUAUUUGACUUCCAAUCAUAAGGUCUAUUAAUUAAUAGUAUAG
AUA 65
((.(.(((((((((.(...)))((((((...)..))..)))...)((((((...))))))))))))).
Score: 22
Base pair distance: 20
Hamming distance: 37

Seq2 with no constraint
UUUCCUGUCCCUUACAUGCAGUGCUUUAAGAGGCUAACACAGAAGGGUAAAGUAAAUCUCCA
CGAAACCCAGAGAAGAGAUUUUUAAAACUCCUCUUUGGAUCCUGUCUGGAGUCACAGCU 121
.((..(((((((((.(.((.(((.(((...)))))))..))..)))))))((...)))(.((((((.(..(((
(((((.((((.(((...)))))))).)))))))(...)))))))))))).))..
Score: 44

Seq2 with constraint

UUUCCUGUCCCUUACAUGCAGUGCUUUAAGAGGCUAACACAGAAGGGUAAAGUAAAUCUCCA
CGAAACCCAGAGAAGAGAUUUUUAAAACUCCUCUUUGGAUCCUGUCUGGAGUCACAGCU 121
. ((. . ((((((((. (. (((. . .) . ((. . . (. . . .)))) . .)))))) ((. . .)) .)) . ((. . .)) .) (((
. (((((((((. ((. . .)) . .)) .))))) ((. . .)) .) .))) .) . .) .) . . .
Score: 37
Base pair distance: 32
Hamming distance: 57

For the first sequence, the base pair and hamming distances are 20 and 37 respectively, while the second sequence is longer, that yields higher distances between the two. Overall observations from dot bracket notation, they look similar, and the result shows that the prediction with constraint generates more hairpin loops than the basic nussinov. The hairpin loop is crucial for the RNA folding process, interacts with ribozyme, protects mRNA, and functions as a motif for proteins. However, when comparing the base pair and hamming distances with the real structure from the Rfram website, the basic Nussinov performs better than the Nussinov with constraint, as shown in the table 1.

Table 1 comparison with the real structure

	Nussinov		**with constraint**	
	Seq 1	**Seq2**	**Seq 1**	**Seq2**
Base pair distance	**12**	**17**	**14**	**32**
Hamming distance	**30**	**45**	**31**	**68**

From a biological point of view, the nussinov algorithm predicts secondary structure by finding maximum base pairs associated with the canonical Watson–Crick base, which means that “A” is needed to pair with “U” as always whenever we run the algorithm. But it is not a reflection of a natural RNA folding process. “A” does not need to pair with “U” all the time; some hidden factors may influence this pairing. The algorithm with constraint, on the other hand, helps to model one step closer to a natural process, but we still require some more experimental data and optimization of the criteria. The implementation with constraint is easy to adjust the criteria and fast, but there are some weaknesses; for example, this implementation does not work well with longer sequences, and it does not include hairpin-loop and pseudoknots.