Programming Lab I
Handout 8

Dr. Martin Vogt
`martin.vogt@bit.uni-bonn.de`

Jun 23, 2020

Deadline: Jul 8, 2020 **Next Handout: Jun 30, 2020**

The dynamic programming principle (Some background)

The dynamic programming principle can be applied if a problem exhibits the following two properties:

1. The optimal substructure property,
2. the overlapping subproblem property.

The *optimal substructure property* states that an optimal solution can be constructed from the optimal solution of subproblems. For the Needleman-Wunsch sequence alignment the subproblems are the alignments of the sequences without the last symbol in either sequence. So when aligning $(x_i)_{i=1...n}$ $(y_i)_{i=1...m}$ the three subproblems to solve are:

1. Align $(x_i)_{i=1...n-1}$ and $(y_i)_{i=1...m}$,
2. align $(x_i)_{i=1...n}$ and $(y_i)_{i=1...m-1}$ and
3. align $(x_i)_{i=1...n-1}$ and $(y_i)_{i=1...m-1}$.

Considering the solutions to these subproblems and the score for either introducing a gap or a match yields the optimal solution for the full problem.

The *overlapping subproblem* states that the solution to the subproblems can be *reused* in constructing the optimal solution for the problem. For Needleman-Wunsch solutions to the subproblems are stored in a matrix. Entries calculated there are reused in the calculation of the subsequent elements of the matrix. Recall that matrix entry $M(k,l)$ for the Needleman-Wunsch pairwise global alignment algorithm gives the score for the optimal alignment of subsequences $(x_i)_{i=1...k}$ and $(y_i)_{i=1...l}$. The solution $M(k,l)$ is reused in calculating $M(k+1,l)$, $M(l,k+1)$, and $M(l+1,k+1)$.

RNA secondary structure prediction: The Nussinov algorithm

The *Nussinov algorithm* for secondary RNA structure prediction is another example of a dynamic programming algorithm in bioinformatics.

RNA is a single-stranded sequence of nucleic acids and typically folds unto itself so that complementary base pairs are paired.

Consider the RNA sequence

```
AUCGGAGCAUUUUUUGCUCCGACGCAGCCUCAUGCUUUUUU
```

We would like to fold this sequence so that as many complementary base pairs `A-U` or `C-G` as possible align. E.g.:

```
A
UCGGAGCAUU
||||||||   U
AGCCUCGUUU
C
GCAGCC
|||   U
CGUAC
U
U
U
U
U
U
```

Another way to visualize a folding is to think of complementary bases as matching parentheses. Then a folding can be represented by a properly formed parenthesized string where dots indicate unmatched bases:

```
AUCGGAGCAUUUUUUGCUCCGACGCAGCCUCAUGCUUUUUU
.((((((((.....)))))))).(((......)))......
```

Note, that the folding shown here is not necessarily the optimal one.

Thus one is looking for a properly parenthesized string with the maximum numbers of parentheses, which have to correspond to complementary bases. There is caveat however, matching parentheses should probably not be next to each other. This would constitute an impossible folding of the RNA sequence. There would need to be at least one or maybe two bases between matching complementary bases.

The problem, as formulated here, is a textbook example for dynamic programming. Similar to the Needleman-Wunsch algorithm, subsequences are considered and the global solution is constructed from solutions of the subsequences. In sequence alignment we had two sequences and the global solution was constructed from initial segments of the two sequences. Here, there is only one sequence, however instead of considering initial segments we will consider subsequences starting at any position i and ending at any position j with $1 \leq i \leq j \leq n$ where n is the overall length of the RNA sequence. For brevity, such subsequences will be denoted by $i : j$ in the following.

The idea is to calculate a matrix $N(i,j)_{1 \leq i \leq j \leq n}$ recording the maximum number of possible matched base pairs the subsequence $i : j$.

The idea is to first consider all subsequences of length 1 and 2 and then formulate a recursive relation for all longer subsequences.

We know that a single base cannnot form a pair so for every subsequence of length 1 we have:

$$N(i,i) = 0, \text{ for } 1 \leq i \leq n$$

Furthermore, due to physical constraints, subsequence of length 2 should also never form paired bases, Thus:

$$N(i,i+1) = 0, \text{ for } 1 \leq i < n$$

If we now consider an arbitrary subsequence $i : j$ of length $j - i + 1 \geq 3$, a base pair matching can be constructed from a smaller subsequences in four different ways:

1. an unpaired base at position i is added to the best base-pairing for subsequence $i + 1 : j$
2. an unpaired base at position j is added to the best base-pairing for subsequence $i : j - 1$
3. a base pair (i,j) is added to the best base-pairing for subsequence $i + 1 : j - 1$
4. The base-pairing for sequence $i : j$ is constructed from two optimal pairings for subsequence $i : k$ and $k + 1 : j$ for $i \leq k < j$.[1]

[1] Note that because $N(i,i) = 0$ the first two cases are just a special case of this one where $k = i$ or $k = j - 1$.

The recursion can be formulated in mathematical terms in the following way:

$$N(i,j) = \max \begin{cases} N(i+1,j-1) + \delta(i,j) \\ \max_{i \le k < j}[N(i,k) + N(k+1,j)] \end{cases}$$

where $\delta(i,j) = 0$ if bases at position i and j do not form a base pair and $\delta(i,j) = 1$ if the bases at i and j are complementary bases.

Backtracking

From the filled out matrix we want to construct the proper base pairing. That is, we want to reconstruct the list of paired complementary bases (as tuples). To this end we need to use the matrix, starting at the full sequence, i.e. $N(1,n)$ and retrace the steps of the algorithm. The tricky part is the step where a new base pairing is constructed from two separate pairings.

Starting at $N(1,n)$, we will need to check for every $N(i,j)$:

1. $j - i + 1 \le 2$: In this case we are done, as no base pairing is possible for sequences of length 2 or less.
2. $N(i,j) = N(i+1,j-1) + \delta(i,j)$: Two cases need to be distinguished:
   (a) $\delta(i,j) = 0$: There is no base pairing between i and j, i.e. do nothing.
   (b) $\delta(i,j) = 1$. There is a base pairing between i and j, i.e. add (i,j) to the list of base pairings.

   In any case continue with $N(i+1,j-1)$
3. Look for a k in the range i to $j-1$ for which $N(i,j) = N(i,k) + N(k+1,j)$. In this case, we need to retrieve the list of paired complementary bases for $N(i,k)$ and $N(k+1,j)$ and join them.

Exercises

Ex 1. *(8 pts) Nussinov folding algorithm*
Implement the Nussinov folding algorithm, constructing the dynamic programming matrix $N(i,j)$. Your function should take an RNA sequence as argument and return the completed dynaminc programming matrix.

Ex 2. *(8 pts) Nussinov algorithm: Backtracking*
Given the RNA sequence and the completed dynamic programming matrix $N(i,j)$ as input write a function that returns the list of all matched base pairs.

Ex 3. *(2 pts) Display matching base pairs*
To display the resulting base pairing use parentheses to produce an output as shown above. Use the completed code of the previous exercises to display the optimal base pairing for the sequence:

```
AUCGGAGCAUUUUUUGCUCCGACGCAGCCUCAUGCUUUUUU
```

How many base pairs are found?

Ex 4. *(2 pts) Modifications of the algorithm*
In the algorithm as described, the minimum hairpin loop consists of 1 base, e.g `GUC` where `G` and `C` are paired and the loop is formed by the single base `U`. Modify the algorithm so that a parameter h, indicating the minimum allowed loop length, can be given. The above algorithm corresponds to $h = 1$. In addition to $h > 1$ make sure your algorithm also works for $h = 0$.

If you like to challenge yourself:

5. *(Optional 3 pts) Recursion forbidden!*
   Nowadays, most popular programming languages support recursion. That was not always the case. Most notably the language FORTRAN did not support recursion. Solve the backtracking problem of exercise 2 without recursion.

6. *(Optional 3 pts) How many ways?*
   The base pairing with the highest number of base pairs might not be unique. For the sequence from exercise 4, determine how many different possibilities to assign the maximum number of base pairs exist in the default implementation, i.e. with $h = 1$.

Note: The Nussinov folding algorithm is a little too simple to yield good results. Instead algorithms like the Zuker algorithm are used that are based on energy minimization considerations to identify the optimal folding structure of RNA. However, at their heart, these algorithms employ dynamic programming strategies very similar to the one of the Nussinov algorithm