Ananke Garani-Krishnan
Anna Ritz
Bio 131

Cyclopeptide Sequencing Project

Repl link: https://repl.it/@anankek/Final-Project

Cyclopeptides, such as antibiotic tyrocidines, are small, approximately ten amino acid long proteins that are *not* synthesized by ribosomes via the central dogma. Instead, they are synthesized by large proteins called NRP (non-ribosomal peptide) synthase, which simply attaches amino acids together without using the genetic code. These tiny cyclopeptides are used by bacteria to communicate and kill other bacteria, and have human applications as antibiotics and tumor or immunosuppressant drugs (Jones et. al 2004).

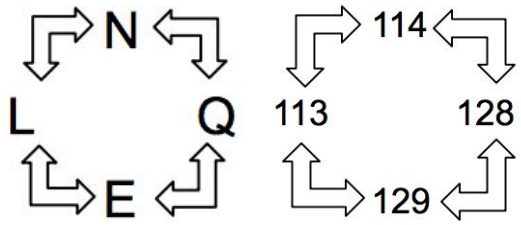
The problem with sequencing cyclopeptides is that they are not encoded in the genome, and cannot be sequenced using past algorithms. Instead, we use a mess spectroscopy technique that shatters the cyclopeptide into smaller fragments and returns a cyclospectrum showing the weight of each of the fragments. Each amino acid has a specific, known integer mass (see Table 1) it is possible to reconstruct the full peptide with these masses (Jones et. al 2004).

G	A	S	P	V	T	C	I	L	N	D	K	Q	E	M	H	F	R	Y	W
57	71	87	97	99	101	10	113	113	114	115	128	128	129	131	137	147	156	167	186

Table 1. Integer masses of amino acids in Daltons (Da) from *Bioinformatic Algorithms.*

The computational problem to sequence cyclopeptides is formulated using a theoretical cyclospectrum that contains all possible fragments from the cyclopeptide, as well as an empty fragment (mass: 0) and the entire peptide (mass: total peptide mass). There is a way to use a leaderboard sequencing technique to sequence cyclopeptides from messy spectrums, but I did not get there. Given a small cyclopeptide, it is possible to generate the theoretical spectrum and then use the theoretical spectrum to regenerate the peptide, although it is more difficult to go backwards than forwards. This project implemented problems 4C and 4D in the *Bioinformatics*

Algorithms textbook, and used given cyclospectrums to sequence cyclopeptides, including antibiotic Tyrocidine B1. For the purposes of this report, the example cyclopeptide NQEL will be used as an example.



NQEL	NQE	QEL	LNQ	ELN	NQ	QE	EL	LN	N	Q	E	L	" "
484	371	370	355	356	242	257	242	227	114	128	129	113	0

Figure 1. Cyclopeptide NQEL, represented with both 1-letter amino acid abbreviations (top left) and integer masses (top right). The theoretical fragments of the cyclospectrum, and their respective weights, are shown in the table (bottom).

The algorithm to generate a theoretical spectrum from a peptide is relatively straightforward. The function theoreticalSpectrum inputs a string, *peptide*, and outputs a dictionary, *cyclospectrum*, where the keys are the fragments and the values are the masses. This function loops through all possible orders of the string peptide (e.g. NQEL, LNQE, ELNQ, QELN) and calculates the linear spectra for each peptide, which accounts for the cyclic nature of the overall peptide. One potential problem is that this algorithm does not account for fragment sequences that appear more than once, but this has not been a problem with the peptides I sequenced so far. The pseudocode for this function is shown below, where text highlighted in yellow are helper functions (see Appendix for helper functions pseudocode).

```
theoreticalSpectrum (peptide)
    cyclospectrum ← an empty dictionary
    cyclospectrum[peptide] ← getPeptideWeight(peptide)
    for each amino acid i in the length of the original string peptide
        peptide ← first n letters of (last letter of peptide + peptide), where n is
        length of original peptide
```

```
            for each letter in peptide
                for each letter in peptide
                    add entry to cyclospectrum (key: peptide fragment, value:
                    getPeptideWeight(peptide fragment)
    return cyclospectrum
```

The cyclopeptideSequencing function takes a list *cyclospectrum* and outputs a list *peptide*, which is all possible peptides that fit the given spectrum. *peptide* should contain one string for each of the linear variations of peptide, as well as each of them backwards. For the peptide NQEL, the list would contain: [NQEL, LNQE, ELNQ, QELN, LEQN, EQNL, QNLE, NLEQ]. These strings all represent the same cyclic peptide (Figure 1). To generate this list, the algorithm cyclopeptideSequencing uses a branch and bound strategy, which starts with the individual amino acids in peptide, generates all possible peptide sequences with an "expand" function, and then trims these generated sequences based on whether they are consistent with the provided cyclospectrum or not. The pseudocode for this function is shown below.

```
cyclopeptideSequencing (cyclospectrum)
    peptides ← an empty list
    final_peptide ← an empty list
    for fragment fragment in cyclospectrum
        if fragment = an amino acid monoisotopic mass
            add fragment to peptides
    for peptide peptide in peptides
        while peptides is not empty
            peptides → expand(peptide)
            if getPeptideWeight(peptide) = maximum mass in cyclospectrum
                if linearSpectrum(peptide) is in cyclospectrum
                    add peptide to final_peptide
                    remove peptide from peptides
            if linearSpectrum(peptide) is in cyclospectrum
                remove peptide from peptides
    for peptide peptide in final_peptide
        if theoreticalSpectrum(peptide) = cyclospectrum
            add peptide to peptides
    return peptides
```

My algorithm, both for the cyclopeptideSequencing and theoreticalSpectrum functions, used the 1-letter amino acid abbreviations rather than the integer masses. This posed a problem for amino acids that have identical integer masses, such as I/L and K/Q. My algorithm cannot distinguish between the two and returns the peptides NKEL, NQEI, and NKEI in addition to the correct peptide NQEL. One way to work around this is to return the sequence in terms of integer masses (Figure 1), which would be 114-128-129-113 for all four variations. Although I did not implement this in my cyclopeptideSequencing function, I used the function peptideByMass to represent a cyclopeptide with integer masses. This function takes a string, *cyclopeptide*, and returns a string *numerical_peptide* where the amino acids are integer masses.

```
peptideByMass(cyclopeptide):
    numerical_peptide ← an empty string
    for amino acid amino in cyclopeptide
        add getPeptideWeight(amino) to numerical_peptide
    return numerical_peptide
```

It is possible to implement peptideByMass within the cyclopeptideSequencing function, and return two list representations of the sequenced cyclopeptide (one with 1-letter abbreviations and one with integer masses). However, in my final code, I ended up calling peptideByMass for only one of the sequenced cyclopeptide strings, as all the returned cyclopeptides should have the same integer mass sequence.

Finally, by implementing these algorithms on the cyclospectrum of Tyrocidine B1 (VKLFPWFNQY), I got the integer mass sequence 163-128-114-147-186-97-147-113-128-99, which is one of the many linear representations of this cyclic spectrum.

Appendix (helper functions pseudocode)

getPeptideWeight(*peptide*) [input: string *peptide,* output: integer weight of *peptide*]

```
getPeptideWeight(peptide)
    weight ← 0
    for amino acid amino in peptide:
        add monoisotopic mass of amino to weight
    return weight
```

expand(*peptides*) [input: string *peptides,* output: a list of strings *pep_list*]

```
expand (peptides):
    aminos ← ['A','C','D','E','F','G','H','I','K', 'L','M','N','P','Q','R','S','T','V','W','Y']
    for peptide peptide in peptides:
        for amino acid amino in aminos:
            add each amino in aminos to each peptide in peptides
    return peptides
```

theoreticalLinearSpectrum(*peptide*) [input: a string *peptide*, output: a dictionary of the linear peptide spectrum]

```
theoreticalLinearSpectrum(peptide):
    spectrum ← an empty dictionary
    i ←  length of peptide
    while i > 0
        for amino acid amino in peptide:
            add entry to spectrum (key: peptide fragment, value = weight of
            peptide fragment)
```

References

Jones, N. C., & Pevzner, P. (2004). *An Introduction to Bioinformatics Algorithms*. MIT Press.

I'm okay with this project being shared :)