3 Combinatorics of Weighted Strings

[THIS CHAPTER IS "FEATURE COMPLETE" AND READY FOR PROOFREADING!]

"If numbers aren't beautiful, I don't know what is." (Paul Erdős)

WE NOW TURN TO SOME PROBLEMS that keep reappearing, in various flavors, not only throughout this book but also throughout the computational mass spectrometry literature. These problems, and also our strategies for solving them, lie at the core of many MS applications. In fact, we have already stumbled upon some of the problems in the previous section. All problems circle around the question of decomposing masses: We are given a weighted alphabet, such as the alphabet of amino acid residues; and we want to know if and how peak masses that we see in a mass spectrum can be explained.

For simplicity, we assume throughout this chapter that all masses are integer: For example, we can round amino acid masses to the closest integer. Alternatively, we multiply all masses by a large constant c such as $c = 1000$ before rounding, to reduce the impact of rounding errors. See Sec. 10.1 below for a thorough investigation on how to decompose real numbers, and how this can be applied in metabolomics.

3.1 Formal problem definitions

We are given an alphabet $\Sigma = \{a_1, \ldots, a_k\}$, for example the alphabet of amino acids, or an alphabet of elements. Throughout this chapter, we denote the cardinality of our alphabet by $k = |\Sigma|$. We are also given a mass function $\mu : \Sigma \to \mathbb{N}$. Recall that the mass of a string $s = s_1 \ldots s_n$ over Σ is defined as $\mu(s) := \sum_{i=1}^{n} \mu(s_i)$. In the following, we usually assume that all characters have pairwise different mass, even though this is not required for some of the algorithms. Still and all, there are very few applications where different characters of the same mass are reasonable: Instead, the method of choice usually is to treat all characters of identical mass as one, and to sort out this impreciseness at a later stage, see Exercise 3.6. Recall that for the amino acid alphabet, leucine and isoleucine have identical mass and will be regarded as one character.

One should immediately notice that the order of characters in the string has no effect on the mass: only the number of occurrences of each of the characters is important. To this end, we make the following definition: A *compomer* over Σ can be viewed either as a map $c : \Sigma \to \mathbb{N}$, or as a vector $(c_1, \ldots, c_k) \in \mathbb{N}^k$.[1] Defining compomers as vectors is easier to grasp, whereas defining them as maps is mathematically more elegant: We do not have to order the alphabet, and the definition also works for infinite alphabets. In the following, we will use these two definitions interchangeably. If we view compomers as vectors $c = (c_1, \ldots, c_k)$, the order of characters in the

[1] Compomers have been proposed numerous times throughout the literature, and many different names have been proposed such as *compositions* [15], *Parikh-vectors* [203], *multiplicity vectors* [6], or *abelian patterns*.

alphabet is relevant, and we assume this order to be arbitrarily fixed. Given a string $s = s_1 \dots s_n$, the function $\text{comp} : \Sigma^* \to \mathbb{N}^k$ maps s to its compomer by counting characters,

$$\text{comp}(s) = (c_1, \dots, c_k) \quad \text{with} \quad c_j = \#\{i : s_i = a_i\}. \tag{3.1}$$

The *length* of compomer c is $|c| := \sum_{j=1}^{k} c_j$, and the *mass* of c is $\mu(c) := \sum_{j=1}^{k} c_j \cdot \mu(a_j)$. The definition of length of a compomer, becomes obvious by the following lemma:

Lemma 3.1. *Given a string $s \in \Sigma^*$ and a compomer $c := \text{comp}(s)$. Then $|c| = |s|$ and $\mu(c) = \mu(s)$.*

We will often denote a compomer c as $(a_1)_{c_1} \dots (a_k)_{c_k}$, omitting those characters a_i with $c_i = 0$. This increases readability and is particularly favorable for large alphabets, such as amino acids. This presentation is obviously inspired by molecular formulas from chemistry, such as $C_{12}H_{22}O_{11}$ for sucrose. In fact, we will come back to this link in Sec. 10.2. This shows that compomers also exist "without strings", see Chapter 9. Sometimes, compomers with negative entries make sense: For example, the difference between two compomers can be useful in certain applications.

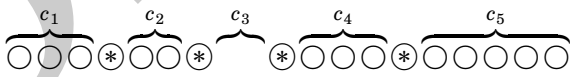
Example 3.1. Let $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$ be our alphabet with masses $\mu(\mathsf{a}) = 2$, $\mu(\mathsf{b}) = 3$, $\mu(\mathsf{c}) = 7$, and $\mu(\mathsf{d}) = 10$. We will make use of this weighted alphabet throughout this chapter. For a string $s = \mathsf{baacbcaca}$ we have $c := \text{comp}(s) = (4, 2, 3, 0)$ in vector notation or, equivalently, $c = \mathsf{a}_4\mathsf{b}_2\mathsf{c}_3$. Now, $|c| = 9$ and $\mu(c) = 4 \cdot 2 + 2 \cdot 3 + 3 \cdot 7 = 35$.

Now, we turn to an obvious question: It is well-known that there are k^n strings of length n over an alphabet of size k. Now, how many compomers of a given length exist?

Lemma 3.2. *The number of compomers of length n over an alphabet of size k is $\binom{n+k-1}{k-1}$.*

Notably, this question has been answered wrongly several times in the mass spectrometry literature as "exponentially many" [1]. But for large alphabets, even though the number is polynomial in n, it is still increasing rapidly: For a fixed alphabet of size k, we have $\binom{n+k-1}{k-1} \in \Theta(n^{k-1})$ many compomers. For the amino acid alphabet of size 19, this implies that the number of compomers is increasing with a polynomial of degree 18.

Lemma 3.2. Every compomer of length n can be mapped bijectively onto $n + k - 1$ points, where $k - 1$ points are selected by asterisks. The compomer $(c_1, \dots, c_k)$ is mapped to c_1 points, asterisk, c_2 points, asterisk, …, asterisk, c_k points. For example, $(c_1, \dots, c_5) = (3, 2, 0, 3, 5)$ is mapped to:



How many possibilities exist to choose (cross out) $k - 1$ points out of $n + k - 1$ points? It is well known that there exist $\binom{n+k-1}{k-1}$ such possibilities. □

There are four problems that we will address in the following: Given a mass integer mass $M \geq 0$, what is the number of compomers and strings with this mass? Is there at least one such compomer or string? If yes, can we provide a witness or proof, that is, a compomer c with $\mu(c) = M$? And finally, can we enumerate all compomers of mass M? Searching for compomers or strings over an alphabet Σ with mass M, we also say that we *decompose* mass M, and the compomers or strings of mass M will be called *decompositions*.

In combinatorics, determining the number of things is called "counting", whereas "enumerating" refers to constructing all objects with a particular property.[2] Usually, counting can be achieved faster than enumerating. In turn, counting (how many?) is at least as hard as the decision problem (at least one?).

In the remainder of the chapter, we name the characters of the alphabet with their integer masses: Instead of the alphabet $\Sigma = \{a, b, c, d\}$ from Example 3.1, we will consider the alphabet $\Sigma = \{2, 3, 7, 10\}$. This will make it easier to follow the formalism. As we assume that all characters have pairwise distinct masses, this is not a restriction. Unless explicitly stated otherwise, we assume that *all masses are positive*. Finally, let us assume that masses in the alphabet $\Sigma = \{a_1, \ldots, a_k\}$ are ordered, so in particular, a_1 is the smallest mass and a_k is the largest mass.

Finally, let us consider the problem of negative integer masses. If all masses are negative, you can take the (additive) inverse of all masses, and you are back at our previous problem. If at least one mass is positive and one mass is negative, there is an infinite number of solutions, see Exercises 3.19, so counting and enumerating does not make sense. But sometimes you are not interested in enumerating all solutions but only some optimum one following, say, the parsimony principle; for example, we might be interested in the minimum number of Post-Translational Modifications that makes some protein fit with some peak mass. Finding such solutions is possible using dynamic programming, and we will come back to this in Exercise 8.5.

3.2 Counting compomers and strings

We have seen in Lemma 3.2 how to compute the number of compomers of given length. In mass spectrometry, the usually more interesting question is: How many compomers exist with mass M? We will present an exact solution based on dynamic programming (see Sec. 17.3) that is actually very simple — a related problem is "scientific folklore" in computer science and combinatorics, see Exercise 3.1. We solve the problem by two-dimensional dynamic programming. Let C be a two-dimensional table, where $C[i, m]$ is the number of compomers c over the alphabet $\{a_1, \ldots, a_i\}$ with mass $\mu(c) = m$, for $i = 0, \ldots, k$ and $m = 0, \ldots, M$. For $i < k$, this means that we do not take into considerations the complete alphabet but only a sub-alphabet. In the extreme case $i = 0$, this corresponds to an empty alphabet and, obviously, the only mass that we can decompose over this alphabet is $m = 0$, and there is exactly one decomposition (the empty compomer) for this mass. Hence, we initialize our table by $C[0, 0] = 1$ and $C[0, m] = 0$ for $m = 1, \ldots, M$.

Let us assume that we have previously computed all entries $C[i', m']$ with $i' \leq i$ and $m' \leq m$, where $i' < i$ or $m' < m$ or both holds. To compute the number of compomers with mass m over the alphabet $\{a_1, \ldots, a_i\}$ we sort these compomers into two buckets: One bucket contains those compomers where $a_i = 0$ holds, the other bucket contains those with $a_i \geq 1$. Clearly, the two buckets are disjoint, so we can add up these two numbers to reach the desired value. But we already know these values: The number of compomers that *do not* use letter a_i is exactly $C[i - 1, m]$, the number of compomers for mass m over the alphabet $\{a_1, \ldots, a_{i-1}\}$. And for the compomers that use letter a_i at least once, we can remove a single letter a_i and count compomers

[2] Depending on the scientific area and the year of publication, you sometimes find "to enumerate" as a synonym for "to count"; prominent examples include Garey and Johnson [87], or "graph enumeration" that deals with counting certain graphs. To reduce confusion, we will stick to this sharp differentiation throughout this book.

of mass $m - a_i$ that use a_i at least zero times: This number is stored in $C[i, m - a_i]$. Obviously, the later number is only meaningful in case $m \geq a_i$. In total, we reach the recurrence:

$$C[i,m] = \begin{cases} C[i-1,m] + C[i,m-a_i] & \text{if } m \geq a_i \\ C[i-1,m] & \text{else} \end{cases} \tag{3.2}$$

At the end of our computation, $C[k,M]$ holds the desired number. We formalized the above argumentation in the proof of Lemma 3.3.

Example 3.2. Consider the weighted alphabet $\Sigma = \{2,3,7,10\}$ from Example 3.1. How many compomers exist with mass $M = 13$? Using (3.2) we compute the following table:

i	a_i	$M=0$	1	2	3	4	5	6	7	8	9	10	11	12	13
0	-	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	1	0	1	0	1	0	1	0	1	0	1	0	1	0
2	3	1	0	1	1	1	1	2	1	2	2	2	2	3	2
3	7	1	0	1	1	1	1	2	2	2	3	3	3	4	4
4	10	1	0	1	1	1	1	2	2	2	3	4	3	5	**5**

For example, $C[2,12] = C[1,12] + C[2,9] = 1 + 2 = 3$. So, the number of compomers is $C[4,13] = 5$. Note that the above table tells us the number of compomers for *any* $m \leq 13$.

An algorithm that applies recurrence (3.2) has running time $O(kM)$, and requires $O(kM)$ space to store table C. In complexity theory, this is called a *pseudo-polynomial* running time: The running time depends linearly (polynomially) on the integer M which is part of the input. Regarding memory consumption, slight improvements are possible: For the computation of entries $C[i,\cdot]$, only entries of type $C[i-1,\cdot]$ and $C[i,\cdot]$ are needed. To this end, we can calculate the table row-by-row, and "forget" each row after computation of the subsequent row has been finished. Then, memory requirements are reduced to $O(M)$. Doing so, we can no longer ask for the number of decompositions for some sub-alphabet $\{a_1,\ldots,a_i\}$ for $i < k$, but this is of minor concern here. An implementation of this idea is given in Alg. 3.1. If we are only interested in the number of compomers for mass M but not for any smaller masses, we can compute the table column-by-column, and reduce memory requirements to $O(k \max_i a_i)$. Not that the later does not depend on M, which is very favorable in applications. On the other hand, we have to forget the number of decompositions for almost all $m < M$, which is unattractive in applications. We reach:

Lemma 3.3. *For a alphabet $\Sigma = \{a_1,\ldots,a_k\}$ of integer masses, the number of compomers with mass m, for each $m = 0,\ldots,M$, can be computed in $O(kM)$ time and with $O(M)$ space using Alg. 3.1.*

The formal proof of this lemma can be found in the next section.

The related question for strings is, how many strings over Σ have mass M? This question is very similar to answer, and requires only one-dimensional dynamic programming: Let $C'[m]$ denote the the number of strings over Σ that have mass exactly m. Then, each such string can be divided into a string that is one character shorter, plus one character. Now, we can sort the strings into k many buckets, depending on the last character, and see that all of these buckets

```
1: function ComputeNumberCompomers(weighted alphabet Σ, mass M)
2:     arrays C[0...M], C'[0...M] of integers
3:     integer i
4:     if k is even then
5:         C[0] ← 1; C[m] ← 0 for m = 1,...,M; i ← 1
6:     else
7:         C[m] ← 1 if m is divisible by a_1, and C[m] ← 0 otherwise; i ← 2
8:     end if
9:     while i < k do
10:        for m = 0,...,a_i − 1 do
11:            C'[m] ← C[m]
12:        end for
13:        for m = a_i,...,M do
14:            C'[m] ← C[m] + C'[m − a_i]
15:        end for
16:        i ← i + 1
17:        for m = 0,...,a_i − 1 do
18:            C[m] ← C'[m]
19:        end for
20:        for m = a_i,...,M do
21:            C[m] ← C'[m] + C[m − a_i]
22:        end for
23:        i ← i + 1
24:    end while
25:    return array C
26: end function
```

Algorithm 3.1: Computing the number of compomers over an alphabet $\Sigma = \{a_1, \ldots, a_k\}$ of integer masses, up to some maximum mass M.

are disjoint. We initialize $C'[m] = 0$ as there is exactly one string (the empty string) of mass zero. We easily reach the recurrence:

$$C'[m] = \sum_{i_1}^{k} C'[m - a_i] \tag{3.3}$$

where we assume that $C'[m] = 0$ holds for all $m < 0$. Put differently: Iterate over all $i = 1, \ldots, k$ and for those that satisfy $m \geq a_i$, add $C'[m - a_i]$ to the total number of strings.

Example 3.3. Consider again the weighted alphabet $\Sigma = \{2, 3, 7, 10\}$. How many strings exist with mass $M = 13$? Using (3.3) we compute:

m	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$C'[m]$	1	0	1	1	1	2	2	4	4	7	10	12	20	**25**

Hence, the number of strings is $C'[13] = 25$. Even for this small example, we observe the different rate of growth for compomers vs. strings, namely, polynomial vs. exponential: Regarding $m = 5$ there exist only one compomer $\mathtt{a}_1\mathtt{b}_1$, but two strings ab and ba.

Note that the problem of counting weighted strings, has a striking similarity with Fibonacci-numbers: In fact, for $\Sigma = \{1,2\}$ we reach the definition of these numbers, $F(n) = F(n-1) + F(n-2)$.[3] For an arbitrary alphabet, (3.3) defines a linear recurrence relation with finite history and constant coefficients [101], or a *linear recursive sequence* for short. In theory, we can find a closed-form solution for any linear recursive sequence. Then, the exact number $C[m]$ can then be computed in constant time, just like the n^{th} Fibonacci number $F(n)$ can be computed as

$$F(n) = \frac{1}{\sqrt{5}}\left(\varphi^n - (1-\varphi)^n\right) = \left\lfloor \tfrac{1}{\sqrt{5}}\varphi^n + \tfrac{1}{2} \right\rfloor$$

where $\varphi = \frac{1+\sqrt{5}}{2}$ is the golden ratio, and $\lfloor\cdot\rfloor$ denotes the floor function for rounding down. The problem is that there exists no simple way to find this closed form for an arbitrary alphabet Σ.

3.3 Formal proof of the counting lemma

We now give a formal proof of Lemma 3.3. Other lemmata and claims in this chapter can be proven similarly, so we will go through this formal exercise only once. Reader with no formal background in mathematics or computer science might want to trust me on the subject matter, and skip this section altogether.

I present this proof as an example of the requirements mentioned in the preface of this book; namely, specification of the input, generalizability of the method, correctness of the algorithm, and running time of the algorithm. We have clearly stated the input of the method (an alphabet of integer masses and a maximum mass M, all non-negative), and we have not stated any restrictions that the algorithm might work only for certain inputs, but choose to fail on other. Now, we will *prove* that the algorithm works correctly for all input, and we will also prove its running time.

First, we show that recurrence (3.2) computes all entries $C[i,m]$ according to the *definition* of the $C[i,m]$: By this definition, $C[i,m]$ is the number of compomers of mass m, over the alphabet $\{a_1,\ldots,a_l\}$. We do so by induction on i and m. As our induction start, we observe that only mass $m = 0$ can be decomposed over the empty alphabet, having a unique decomposition, so the $C[0,\cdot]$ are correctly initialized. Assume that $i \geq 1$. Let $\mathscr{C}$ be the set of compomers over the alphabet $\{a_1,\ldots,a_i\}$ with $\mu(c) = m$ for all $c \in \mathscr{C}$; then, $|\mathscr{C}| = C[i,m]$ must hold. Partition $\mathscr{C}$ into two sets $\mathscr{C}_1, \mathscr{C}_2$ with $\mathscr{C}_1 \cup \mathscr{C}_2 = \mathscr{C}$ and $\mathscr{C}_1 \cap \mathscr{C}_2 = \emptyset$:

$$\mathscr{C}_1 := \{c : c = (c_1,\ldots,c_i) \in \mathscr{C}, c_i = 0\}$$
$$\mathscr{C}_2 := \{c : c = (c_1,\ldots,c_i) \in \mathscr{C}, c_i \geq 1\}$$

Let $\mathscr{C}_1'$ be the set of compomers over the alphabet $\{a_1,\ldots,a_{i-1}\}$ of mass m; by induction, $\left|\mathscr{C}_1'\right| = C[i-1,m]$ must hold. We can easily define a bijection between the sets $\mathscr{C}_1$ and $\mathscr{C}_1'$, either removing the trailing zero, or appending it. Hence, $|\mathscr{C}_1| = \left|\mathscr{C}_1'\right| = C[i-1,m]$.

For $m < a_i$ we obviously have $\mathscr{C}_2 = \emptyset$ and, hence, $\mathscr{C} = \mathscr{C}_1$, so our claim follows. Assume $m \geq a_i$: Then, let $\mathscr{C}_2'$ be the set of compomers over the alphabet $\{a_1,\ldots,a_i\}$ of mass $m - a_i$; by induction, $\left|\mathscr{C}_2'\right| = C[i,m-a_i]$ must hold. We define a bijection $\varphi : \mathscr{C}_2 \to \mathscr{C}_2'$ by $\varphi(c_1,\ldots,c_{i-1},c_i) := (c_1,\ldots,c_{i-1},c_i-1)$, removing one character a_i with mass a_i from the compomer. Hence, $|\mathscr{C}_2| = \left|\mathscr{C}_2'\right| = C[i,m-a_i]$. As $\mathscr{C}$ is the disjoint union of $\mathscr{C}_1, \mathscr{C}_2$ we reach

$$C[i,m] = |\mathscr{C}| = |\mathscr{C}_1 \cup \mathscr{C}_2| = |\mathscr{C}_1| + |\mathscr{C}_2| = C[i-1,m] + C[i,m-a_i]$$

[3] Note that the sequence is shifted, though, as we set $C[0] = 1$.

as claimed.

Next, we make sure that Alg. 3.1 does in fact compute recurrence (3.2). But this is rather easy to see: During the course of the algorithm, i is increased from 1 or 2 to M with increment 1. After line 8 of the algorithm, array $C[\cdot]$ equals $C[i,\cdot]$ for either $i = 1$ or $i = 2$. We claim that at the start of each WHILE-loop, we have $C[m] = C[i,m]$ for all $m = 0,\dots,M$, where $C[i,m]$ is computed by (3.2). To this end, after the execution of the first FOR-loop (at line 16) we know that $C'[m] = C[i,m]$ must hold; similarly, after the execution of the second FOR-loop we again have $C[m] = C[i,m]$, as claimed.

Finally, we consider running time and memory of the algorithm: Space is clearly $O(M)$ for storing arrays C, C'. But equally clearly, running time is $O(kM)$ as initialization requires $M+1$ assignments. Afterwards, we have $\lceil k/2 \rceil$ outer loops; in each loop, we do $2M+2$ assignments and $O(M)$ summations. □

3.4 Finding witnesses and the decision problem

We now turn to the slightly simpler question: Is there a compomer with mass M over the alphabet $\Sigma = \{a_1,\dots,a_k\}$? Note that we can answer this question by using our algorithms from the previous section, checking whether "$C[k,M] \geq 1$". We will now give a related solution but here, we only need a one-dimensional binary table A. We define $A[m] = 1$ if and only if there is at least one compomer of mass m over the alphabet $\{a_1,\dots,a_k\}$. We initialize $A[0] = 1$, and use the recurrence

$$A[m] = \begin{cases} 1 & \text{if there is some } i \text{ with } A[m-a_i] = 1 \text{ for } m \geq a_i, \\ 0 & \text{else.} \end{cases} \tag{3.4}$$

Note the similarity with (3.3): Actually, asking whether there exists some compomer of mass M, or if there exists some string of mass M, is equivalent. Computation of table A again requires $O(kM)$ time and $O(M)$ space. In case we do not want to store table A for "future use", memory consumption can be reduced to $O(\max_i a_i)$. Again, our algorithm has pseudo-polynomial running time, linear in M. In contract, the size of the input is only $\log_2 M$ as this is the number of bits required to encode the number M in memory. Unfortunately, deciding if there is a compomer or a string of mass M is NP-hard [153]: No exact algorithm with running time polynomial in $\log M$ can exist, unless P = NP. The pseudo-polynomial algorithm introduced above is no contradiction to this hardness result: In fact, the problem is weakly NP-hard, but not strongly [87].

How can we produce a witness, that is, find some compomer c with $\mu(c) = M$? Here and in the following, let $e_i = (0,\dots,0,1,0,\dots,0)$ denote the i^{th} unit vector that has all-zero entries, except for the i^{th} entry, which equals one. Finding a witness is very simple, using table A: Assume that $A[M] = 1$. Start with $c \leftarrow 0$ and $m \leftarrow M$. Find some i such that $m \geq a_i$ and $A[m-a_i] = 1$. Set $c \leftarrow c + e_i$ and $m \leftarrow m - a_i$, and repeat until $m = 0$. Output c. Similarly, we can build a witness string s with $\mu(s) = M$. One can easily see that this algorithm is correct, and has running time $O(k\frac{M}{a_1})$.

Example 3.4. Consider again the weighted alphabet $\Sigma = \{2,3,7,10\}$. We want to compute a witness c for mass $M = 13$. Here is table A, modified from Example 3.3:

m	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$A[m]$	1	0	1	1	1	1	1	1	1	1	1	1	1	1

```
1: procedure FindAllRec(integer i ≤ k, mass m, compomer c)
2:     if i = 0 then
3:         Output c and return
4:     end if
5:     if B[i − 1, m] = 1 then
6:         FindAllRec(i − 1, m, c)
7:     end if
8:     if m ≥ a_i and B[i, m − a_i] = 1 then
9:         FindAllRec(i, m − a_i, c + e_i)
10:    end if
11: end procedure
```

Algorithm 3.2: Recursive algorithm for enumerating all compomers of a given mass m. To decompose mass M, this algorithm is initially called as FindAllRec$(k, M, 0)$.

We start with $m \leftarrow 13$ and $c = (0,0,0,0)$. For $i = 1$ we find $A[m-a_1] = A[13-2] = 1$, so we set $m \leftarrow m - a_1 = 11$ and $c \leftarrow (0,0,0,1)$. We repeat this four more times and reach $m = 3$ and $c = (0,0,0,5)$. Now, $A[m-a_1] = 0$, but for $i = 2$ we have again $A[m-a_2] = A[3-3] = 1$. We set $m \leftarrow m - a_2 = 0$ and $c \leftarrow (0,0,0,5) + (0,0,1,0) = (0,0,1,5)$, and we are done. As desired, $\mu(c) = 5 \cdot 2 + 1 \cdot 3 = 13$.

3.5 Enumerating strings and compomers

Finally, we consider the question most interesting for the majority of MS applications: Given a mass M, find all strings s with $\mu(s) = M$, and find all compomers c with $\mu(c) = M$. First, we consider creating all strings of mass M.

Now, we consider the problem of enumerating all compomers c with $\mu(c) = M$. This problem can be solved by backtracking through the table C: For that, we consider the two "buckets" of recurrence (3.2), and follow both cases to actually compute the individual compomers. As for the decision problem, a binary table is sufficient for this task, as only requests of the form "$C[i,m] > 0$?" have to be answered. Unfortunately, we cannot use table A to enumerate all compomers, as the information stored there is not sufficient. To this end, we define a third, two-dimensional binary table B. We define $B[i,m] = 1$ if and only if there is at least one compomer of mass m over the sub-alphabet $\{a_1, \ldots, a_i\}$. Now, we initialize $B[0,0] = 1$ and $B[0,m] = 0$ for $m = 1, \ldots, M$, and use the recurrence

$$B[i,m] = \begin{cases} 1 & \text{if } B[i-1,m] = 1 \text{ or } B[i,m-a_i] = 1 \text{ for } m \geq a_i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

Clearly, $B[i,m] = 1$ holds if and only if $C[i,m] > 0$.

See Alg. 3.2 for the pseudo-code of the enumeration algorithm. Clearly, we can replace the statement "$B[i,m] = 1$" by "$C[i,m] > 0$". The algorithm is written as a recursion for the sake of simplicity, but we can also do this task iteratively, see Alg. 3.5 on page 55. These algorithms can be easily modified to take into account upper and lower bounds for each character, see Exercise 3.18.

How much time is needed to compute all compomers? Comparable to the algorithm for computing a single witness, the algorithm FindAllRec requires $O(k\frac{M}{a_1})$ time *per decomposition*. So, the running time is linear in the size of the output, which is quite obvious: the larger the

output, the longer the running time. But even if there is only a few compomers with mass M, the running time also depends linearly on M, which is somewhat unfavorable. In the next section, we will get to know a different approach that does not have this unfavorable property.

3.6 The Money Changing problem and the Round Robin algorithm

In Sec. 3.4, we asked if there is at least one compomer c with mass $\mu(c) = M$. This problem was first posed in 1884 and is known as the MONEY CHANGING PROBLEM. To understand what this has to do with money changing, assume that we live in a country where only coins with values $a_1,\ldots,a_k$ such that $a_1 < a_2 < \cdots < a_k$ are available. We want to know what change can be given with these coins. This problem is trivial if a coin with value $a_1 = 1$ exists. Let $g := \gcd(a_1,\ldots,a_k)$ be the *greatest common divisor* of numbers $a_1,\ldots,a_k$: So, g is a divisor of each a_i for all $i = 1,\ldots,k$, and g is the largest such integer. In case $g > 1$ then it is easy to see that we can only make change for numbers $0, 1g, 2g, 3g, \ldots$. In the following, we will usually assume $\gcd(a_1,\ldots,a_k) = 1$. In turns out that the results of this section can also be applied in case $\gcd(a_1,\ldots,a_k) > 1$, see the end of the section for the simple details.[4]

Let $\Sigma = \{a_1,\ldots,a_k\}$ be a fixed weighted alphabet with $\gcd(a_1,\ldots,a_k) = 1$. We want to decide whether some mass M is decomposable or not over Σ. For an integer M, we write $r = M \mod a_1$ for the *residue* of M modulo a_1: This is the unique number $r \in \{0,\ldots,a_1-1\}$ such that $M = qa_1 + r$ for some integer q. For $M \geq 0$ we easily see that $q = \lfloor M/a_1 \rfloor$. We say that M belongs to *residue class* r (modulo a_1).

A simple observation is as follows: If M is decomposable, then $M+a_1, M+2a_1, M+3a_1, \ldots$ are also decomposable. (It holds that $M + a_i$ is decomposable for any $i = 1,\ldots,k$, but we only need the statement for $i = 1$.) This implies that there is a smallest such mass that is decomposable. For each residue classes $r = 0,\ldots,a_1-1$, let $N[r]$ be the smallest mass that is decomposable satisfying $N[r] \mod a_1 = r$. So, $N[0\ldots a_1-1]$ is a one-dimensional array satisfying

$$N[r] = \min\{n : r = n \mod a_1, \text{ and } n \text{ is decomposable over } \{a_1,\ldots,a_k\}\}$$

for $r = 0,\ldots,a_1-1$. Here, $N[r] = +\infty$ if no such number exists, and the minimum is empty. Clearly, $\mu(c) = N[r]$ for some compomer $c = (c_1,\ldots,c_k)$ implies $c_1 = 0$ because otherwise, $N[r]-a_1$ has a decomposition, too. The table $N[0\ldots a_1-1]$ form the *residue table* of the instance.

Example 3.5. For the remainder of this section, we will consider the weighted alphabet $\Sigma = \{5,8,9,12\}$. The residue table $N[0\ldots a_1-1]$ of this alphabet is:

r	0	1	2	3	4
$N[r]$	0	16	12	8	9

It is straightforward to check that this truly is the residue table of the above instance: Clearly, mass 0 belongs to residue class $r = 0$ (modulo $a_1 = 5$) and obviously, there is no smaller non-negative integer. Next, 16 belongs to residue class $r = 1$ and can be decomposed as $16 = 8+8$, whereas we cannot decompose $16-5 = 11$. We can continue in this fashion up to residue class $r = 4$, where $9 = 9$ can be decomposed but $9-5 = 4$ cannot.

[4] Many countries got rid of small coins, but since supermarkets love prices such as \$0.99, the amount you have to pay has to be rounded.

Assume that we know the residue table $N[0\ldots a_1-1]$ of a weighted alphabet: This allows us to answer the question "is mass M decomposable?" in *constant* time. We simply calculate $r \leftarrow M \bmod a_1$; then, M is decomposable if and only if $M \geq N[r]$. For example, assume that we want to know if 17 can be decomposed over the weighted alphabet from Example 3.5. We calculate $17 \mod 5 = 2$ and check $17 \geq N[2] = 12$, so the answer is "yes". On the other hand, we cannot decompose 11, as $11 \mod 5 = 1$, and $11 < N[1] = 16$.

Before we concentrate on computing the residue table, we take a short detour: Recall that $\gcd(a_1,\ldots,a_k) = 1$. Then there exists a number $g := g(a_1,\ldots,a_k)$, called the *Frobenius number*, such that g cannot be decomposed, but *all* masses $M > g$ can be decomposed. It might be somewhat surprising that for an arbitrary weighted alphabet, all sufficiently large masses can be decomposed. See Exercise 3.14 for the proof of the special case $k = 2$. Given the residue table $N[0\ldots a_1-1]$ of an instance, there is a simple formula to compute the Frobenius number g, as well as the number ω of *omitted* values that cannot be decomposed over $\Sigma = \{a_1,\ldots,a_k\}$:

$$g = \max_{r=0,\ldots,a_1-1}\{N[r]\} - a_1 \quad \text{and} \quad \omega = \sum_{r=0}^{a_1-1}\left\lfloor \frac{N[r]}{a_1} \right\rfloor = \frac{1}{a_1}\sum_{r=0}^{a_1-1} N[r] - \frac{a_1-1}{2}. \tag{3.6}$$

These two formulas may also appear somewhat surprising but in fact, it is rather straightforward to show that these identities hold; see Exercise 3.15. For the weighted alphabet from Example 3.5, we can read from the residue table that $g = 16-5 = 11$, and there are

$$\omega = \frac{16+12+8+9}{5} - \frac{4}{2} = 9-2 = 7$$

masses without a decomposition; namely, these are masses $1,2,3,4,6,7,11$.

Now, the interesting question is: Given a weighted alphabet $\{a_1,\ldots,a_k\}$, how can we efficiently calculate its residue table $N[0\ldots a_1-1]$? This can be achieved by the *Round Robin* algorithm: We compute the values of N iteratively for the sub-problems "Find $N[0\ldots a_1-1]$ for the instance $\{a_1,\ldots,a_i\}$", for $i = 1,\ldots,k$. For $i = 1$ we can only decompose masses of the form $M = ia_1$ whereas all other masses cannot be decomposed. Hence, we start with $N[0] = 0$ and $N[r] = \infty$ for $r = 1,\ldots,a_1-1$. When constructing the residue table for the next step, the current values $N[0\ldots a_1-1]$ are updated. Suppose we know the correct values $N'[r]$ for the sub-problem $\{a_1,\ldots,a_{k-1}\}$, and we want to calculate those of the original problem $\{a_1,\ldots,a_k\}$. We first concentrate on the simple case that $\gcd(a_1,a_k) = 1$. We initialize $N[r] \leftarrow N'[r]$ for all $r = 0,\ldots,a_1-1$, and $n \leftarrow N[0] = 0$. In every step of the algorithm, set $n \leftarrow n + a_k$ and $r \leftarrow n \bmod a_1$. Let $n \leftarrow \min\{n, N[r]\}$ and $N[r] \leftarrow n$. We repeat this loop until n equals 0. In case all $a_2,\ldots,a_k$ are coprime to a_1 — that is, $\gcd(a_1,a_i) = 1$ holds for all $i = 2,\ldots,k$ — then this short algorithm is already sufficient to find the correct values $N[r]$.

Example 3.6. Consider the weighted alphabet $\Sigma = \{5,8,9,12\}$ from Example 3.5. In Figure 3.1, each column can be viewed as representing one iteration of the Round Robin algorithm. For example, focus on the column $a_3 = 9$. We start with $n = 0$. In the first step, we have $n \leftarrow 9$ and $r = 4$. Since $n < N[4] = 24$ we update $N[4] \leftarrow 9$. Second, we have $n \leftarrow 9+9 = 18$ and $r = 3$. In view of $n > N[3] = 8$ we set $n \leftarrow 8$. Third, we have $n \leftarrow 8+9 = 17$ and $r = 2$. Since $n < N[2] = 32$ we update $N[2] \leftarrow 17$. Fourth, we have $n \leftarrow 17+9 = 26$ and $r = 1$. In view of $n > N[1] = 16$ we set $n \leftarrow 16$. Finally, we return to $r = 0$ via $n \leftarrow 16+9 = 25$.

It is straightforward how to generalize the algorithm for $d := \gcd(a_1,a_i) > 1$: In this case, we do the updating independently for every residue $p = 0,\ldots,d-1$: Only those $N[r]$ for $r \in$

r	$a_1 = 5$	$a_2 = 8$	$a_3 = 9$	$a_4 = 12$
0	0	0	0	0
1	∞	16	16	16
2	∞	32	17	12
3	∞	8	8	8
4	∞	24	9	9

Figure 3.1: Extended residue table $N[0\ldots4, 0\ldots4]$ of the weighted alphabet $\Sigma = \{5,8,9,12\}$ from Example 3.5, as well as iterations of the Round Robin algorithm.

```
1:  procedure RoundRobin(weighted alphabet Σ)
2:      initialize N[0] ← 0 and N[r] ← ∞ for r = 1,...,a_1 − 1
3:      for i ← 2,...,k do
4:          d ← gcd(a_1, a_i)
5:          for p ← 0,...,d − 1 do
6:              find n = min{N[q] : p = q mod d, 0 ≤ q ≤ a_1 − 1}
7:              if n < ∞ then
8:                  for j ← 1,...,a_1/d − 1 do                ▷ repeat a_1/d − 1 times
9:                      n ← n + a_i
10:                     r = n mod a_1
11:                     n ← min{n, N[r]}
12:                     N[r] ← n
13:                 end for
14:             end if
15:         end for
16:     end for
17: end procedure
```

Algorithm 3.3: Constructing the residue table $N[0\ldots a_1 - 1]$ of a weighted alphabet $\Sigma = \{a_1,\ldots,a_k\}$.

$\{0,\ldots,a_1 - 1\}$ are updated that satisfy $r = p \mod d$. To guarantee that the Round Robin loop completes updating after a_1/d steps, we have to start the loop from a minimal $N[r]$ with $r = p \mod d$. For $p = 0$ we know that $N[0] = 0$ is the unique minimum, while for $p \neq 0$ we search for the minimum first. See Alg. 3.3 for the pseudo-code of the algorithm. The inner loop (lines 8–13) will be executed only if the minimum $\min\{N[q]\}$ is finite; otherwise, the elements of the residue class cannot be decomposed over $a_1,\ldots,a_i$ because of $\gcd(a_1,\ldots,a_i) > 1$.

It is quite easy to see that the Round Robin algorithm computes the residue table of a weighted alphabet $\Sigma = \{a_1,\ldots,a_k\}$ in $\Theta(k\,a_1)$ time; besides $O(a_1)$ memory for storing the current residue table, we need only constant extra memory.

Example 3.7. Consider the weighted alphabet $\Sigma = \{6,7,8\}$. Now, $\gcd(a_1,a_3) = 2$ so for $i = 3$, we have to compute the residue table $N[0\ldots a_1 - 1]$ in two independent round robin runs. The residue tables computed by the iterations of the Round Robin algorithm are:

r	$a_1 = 6$	$a_2 = 7$	$a_3 = 8$
0	0	0	0
1	∞	7	7
2	∞	14	8
3	∞	21	15
4	∞	28	16
5	∞	35	23

For the last column, we first consider $p = 1$: then, $n = \min\{0, 14, 28\} = 0$. We repeat two times: In the first step, we set $n \leftarrow 0 + 8 = 8$ and $r = 2$. Since $n < N[2] = 14$ we update $N[2] \leftarrow 8$. Second, we have $n \leftarrow 8 + 8 = 16$ and $r = 4$. In view of $n < N[4] = 28$ we set $N[4] \leftarrow 28$. The next step would bring us back to $r = 0$. Next, we consider $p = 1$: here, $n = \min\{7, 21, 35\} = 7$. We again repeat two times: In the first step, we set $n \leftarrow 7 + 8 = 15$ and $r = 3$. Since $n < N[3] = 21$ we update $N[3] \leftarrow 15$. Second, we have $n \leftarrow 15 + 8 = 23$ and $r = 5$. In view of $n < N[5] = 35$ we set $N[5] \leftarrow 23$. The next step would bring us back to $r = 1$, and we are done.

For enumerating *all* decompositions of mass M, the information contained in the residue table is unfortunately insufficient. But to our delight, we have implicitly come up with a data structure that allows us to tackle the enumeration problem: Namely, for each $r = 0, \ldots, a_1 - 1$ and each $i = 1, \ldots, k$, we search for the smallest number $N[i, r]$ such that $r = N[i, r] \mod a_1$, and $N[i, r]$ is decomposable over $\{a_1, \ldots, a_i\}$. Formally, we define the *extended residue table* $N[0 \ldots k, 0 \ldots a_1 - 1]$ to be a two-dimensional table such that

$$N[i, r] = \min\{n : r = n \mod a_1, \text{ and } n \text{ is decomposable over } \{a_1, \ldots, a_i\}\}$$

where $N[i, r] = +\infty$ if no such number exists, and the minimum is empty. Clearly, space for storing the extended residue table $O(ka_1)$. Here, the nice feature is that there is no "largest mass" that we have to decide upon during preprocessing.

The nice feature of the Round Robin algorithm is that we have already computed the extended residue table of the instance: We simply have to store each iteration of the algorithm as a "column" of the matrix, when iterating $i = 0, \ldots, k$. See Fig. 3.1 for the extended residue table of Example 3.5.

We can easily use the extended residue table to enumerate all decompositions: In fact, we simply have to replace the query "$B[i, m] = 1$" in Alg. 3.2 by the equivalent query "$m \geq N[i, r]$ for $r = m \mod a_1$". See Alg. 3.2 for the result. In fact, we easily transform the iterative variant of that algorithm, namely Alg. 3.5, into an iterative variant using the extended residue table, see Exercise 3.10.

So, we have improved upon the memory consumption of our approach, as well as the running time during preprocessing. Also, the new algorithm has the desirable property that we do not have to decide upon a largest mass that we want to decompose during preprocessing. Instead, for a fixed weighted alphabet, we compute its unique extended residue table; this allows us to compute decompositions for *any* mass m at a later stage. It turns out that the resulting algorithm is also much faster in practice, at least for certain applications: This is due to the reduced memory consumption, which allows us to store the extended residue table in the processor cache, instead of having to store array B in main memory, see Sec. 10.1.

One thing that we have not improved upon, is the running time per decomposition. But there is a modification of the algorithm so that we can guarantee that every decomposition is computed in $O(ka_1)$ time: To this end, we do not recurse in an arbitrary order but instead, treats

```
1: procedure FINDALLERT(integer i ≤ k, mass m, compomer c)
2:     if i = 0 then
3:         Output c and return
4:     end if
5:     r ← m mod a_1
6:     if m ≥ N[i − 1, r] then
7:         FINDALLERT(i − 1, m, c)
8:     end if
9:     r ← (m − a_i) mod a_1
10:    if m ≥ a_i and m − a_i ≥ N[i, r] then
11:        FINDALLERT(i, m − a_i, c + e_i)
12:    end if
13: end procedure
```

Algorithm 3.4: Recursive algorithm for enumerating all compomers of a given mass m, based on the Extended Residue Table $N[0\ldots k, 0\ldots a_1-1]$. To decompose mass M, this algorithm is initially called as FINDALLERT$(k, M, 0)$.

all recursions for each residue class in one batch. Whereas the resulting algorithm allows us to prove an improved worst-case running time, the overhead required for processing the residue classes individually, is usually too high in applications. The algorithm can be found in Fig. 4 of [24], we defer further details.

Finally, a few words about the case $g := \gcd(a_1,\ldots,a_k) > 1$ that we have ignored so far. To cover this case is rather simple: Replace masses $a_1,\ldots,a_k$ by new masses $a_1/g,\ldots,a_k/g$, and construct the (extended) residue table for this weighted alphabet. If you want to decompose a mass M (or decide if it is decomposable), first check if $M \mod g = 0$ holds: Otherwise, M has no decomposition over $a_1,\ldots,a_k$. Next, decompose the mass M/d over the alphabet $a_1/g,\ldots,a_k/g$; all decompositions that you compute, are also decompositions of M over $a_1,\ldots,a_k$.

3.7 Approximating the number of compomers

Before we start this section, a word of warning is in place. The term "f approximates g" for two functions $f, g : \mathbb{N} \to \mathbb{R}$ can be used with many different meanings: In computer science, this means that we can calculate some number with a *guaranteed* relative error, so $f(n) \le (1+\varepsilon)g(n)$ or $f(n) \ge (1-\varepsilon)g(n)$ for *all* $n \in \mathbb{N}$. Here, $\varepsilon > 0$ can be a constant (sometimes, even an arbitrary constant) or a function depending on n. In mathematics, this sometimes means that f and g are *asymptotically equivalent* or *asymptotically equal*, so $\lim_{n\to\infty} f(n)/g(n) = 1$, what is denoted $f \sim g$. Hence, f "behaves like" g and as n goes towards infinity, the relative error goes to zero. Be aware that in both cases, we only consider relative errors: The absolute difference may be huge and might even go to infinity as $n \to \infty$, see Exercise 3.20. Be also warned that $f \sim g$ does not tell us how fast the error goes to zero, and that the approximation might be arbitrarily bad for the first N numbers where, again, N might be arbitrarily large, such as $N = 10^{1000}$. Finally, colloquial speaking, "f approximates g" means that "f is somewhat close to g". For this last

case, we will say that *f estimates g*, so that it cannot be confused with the two formal uses of this word.[5]

Let $\gamma(M)$ denote the number of compomers with mass exactly M, over some fixed alphabet Σ. Often, we do not have to compute $\gamma(M)$ exactly but rather, want to compute a reasonable estimate. Luckily, there is a simple formula that can help us to estimate the number of decompositions in constant time. The following result is due to Issai Schur:

Theorem 1. *If* $\gcd(a_1,\ldots,a_k) = 1$ *then*

$$\gamma(M) \sim \frac{1}{(k-1)!a_1a_2\cdots a_k} M^{k-1} \quad \textit{for } M \to \infty. \tag{3.7}$$

Actually, we can infer from this theorem that every sufficiently large number M is decomposable over Σ. Unfortunately, convergence is rather slow. A better approximation was given by Beck, Gessel, and Komatsu [12]:

$$\gamma(M) \approx b_{k-1}M^{k-1} + b_{k-2}M^{k-2} + b_{k-3}M^{k-3}\ldots \tag{3.8}$$

where

$$\begin{aligned} b_{k-1} &:= \frac{1}{a_1\cdots a_k}\cdot\frac{1}{(k-1)!} \\ b_{k-2} &:= \frac{1}{a_1\cdots a_k}\cdot\frac{1}{2(k-2)!}\cdot\sum_{i=1}^{k} a_i \\ b_{k-3} &:= \frac{1}{a_1\cdots a_k}\cdot\frac{1}{4(k-3)!}\cdot\left(\frac{1}{3}\sum_{i=1}^{k} a_i^2 + \sum_{i<j} a_ia_j\right) \end{aligned} \tag{3.9}$$

In fact, the authors show how to compute all the coefficients of the polynomial, we omit the details. See Sec. 10.1 on how this can be used to estimate the number of compomers over the amino acid, and the number of molecular formulas over some alphabet of elements.

We mentioned that the above estimates do not give any guarantees, such as: The approximation will in all cases be at most twice as large as the number of decompositions. Only when M goes to infinity, Theorem 1 guarantee that the relative error will drop to zero. As we will see in Sec. 10.1, Eq. (3.8) cannot be used to give a reasonable estimate of the number of amino acid decompositions. By contrast, even the simpler approximation (3.7) results in reliable estimates of the number of molecular formulas, if we ignore the fluctuation of this number due to the combinatorial nature of the problem.

We will now turn to approximations that give us a guarantee on how large the relative error is. Let $\Gamma(M)$ be the number of decompositions with mass $m \le M$, so $\Gamma(M) = \sum_{m=0}^{M}\gamma(m)$. Dyer [62] gave a *Polynomial Time Approximation Scheme (PTAS)* for this number: We choose an arbitrary relative error $\varepsilon > 0$, then the algorithm computes an estimate $\bar{\Gamma}(M)$ such that

$$\Gamma(M) \le \bar{\Gamma}(M) \le (1+\varepsilon)\Gamma(M)$$

in time $O(k^5 + \varepsilon^{-2}k^4)$. So, we can approximate $\Gamma(M)$ with arbitrary precision, where we trade running time for precision. Note that M itself is no longer part of the running time. Unfortunately, this will not lead to an approximation for the number of compomers with mass *exactly* M. In fact, one can easily see that no PTAS can exist for this number: The reason is that

[5] There are other meanings to this phrase, even in mathematics and computer science: For example, mathematical approximation theory is all about *absolute* errors.

if we can approximate $\gamma(M)$ with performance ratio $\varepsilon = \frac{1}{2}$ in polynomial time, then we can decide in polynomial time whether $\gamma(M) = 0$ holds. We noted above that this is not possible unless P = NP.

3.8 Historical notes and further reading

Our presentation in this chapter roughly follows the paper of Böcker and Lipták [24], see there for additional details and missing proofs.

The idea of using compomers for the analysis of mass spectrometry data, dates back at least to the 1980's: Back in 1984, Sakurai *et al*. [202] used compomers over the amino acid alphabet for *de novo* sequencing of peptides. In their approach, they took an MS/MS spectrum of an unknown peptide with parent mass M, generated all compomers c with $\mu(c) = M$, then generated all strings s with $\text{comp}(s) = c$ and, finally, simulated a reference spectrum for each such string s to compare it against the measured spectrum. Obviously, this approach suffered heavily from the huge number of compomers over an alphabet with 19 characters.

The MONEY CHANGING problem and, in particular, the problem of computing Frobenius numbers has been around in Mathematics for quite some time: In 1884, Sylvester asked for the Frobenius number of $k = 2$ coins a_1, a_2, and Curran Sharp showed that $g(a_1, a_2) = a_1 a_2 - a_1 - a_2$ [221]. For three coins a_1, a_2, a_3, Greenberg [100] and Davison [51] independently discovered simple algorithms with fast running times. Kannan [125] established algorithms that for *any fixed* k, compute the Frobenius number in time polynomial in $\log a_k$. Unfortunately, the running time has a double exponential dependency on k, and cannot be applied for $k \geq 5$. Reading the Frobenius number from the residue table was suggested by Brauer and Shockley [31].

In 2007, Einstein, Lichtblau, Strzebonski, and Wagon [67] presented an elaborate method that can solve instance with $k = 4$ and $a_k \leq 10^{100}$ in under one second, and instances with $k = 7$ and $a_k \approx 10^{1000}$ in a matter of minutes. Other methods might be faster if k is large whereas a_1 is relatively small [13]. Computing the Frobenius number is NP-hard [192], so we cannot hope to find algorithms polynomial in k and $\log a_k$ simultaneously unless P = NP. Many results regarding the MONEY CHANGING problem and Frobenius numbers are based on generating functions, see [235] for an introduction. There has been considerable work on bounds for Frobenius numbers, see Ramírez-Alfonsín [191] for a survey.

The solution of the CHANGE MAKING problem (see Exercise 3.1) was proposed by Gilmore and Gomory [91] in 1965, but it is probably easier to come up with a solution yourself than to find it in their paper.

The MONEY CHANGING problem is also closely related to unbounded integer knapsacks [162]: There, one replaces the condition $\sum_j c_j a_j = M$ by $\sum_j c_j a_j \leq M$. In fact, the approximation result of Dyer [62] mentioned in Sec. 3.7 is for unbounded integer knapsacks. Although these problems look similar, algorithms for solving unbounded integer knapsacks such as the algorithm of Martello and Toth [161], cannot be used for the MONEY CHANGING problem.

Alg. 3.5 is the iterative version of the FINDALL algorithm. Be aware that, although it is more complicated than the recursive Alg. 3.2, it is presumably much faster in application. The fasted variant is hard-coding $|\Sigma|$ many WHILE-Loops. In practice, both approaches will show a comparable running time, as compiler optimizations such as loop unrolling cannot be performed here, see Exercise 3.17.

3.9 Exercises

3.1 Assume you are given an infinite supply of coins with values $\Sigma = \{2,3,7,10\}$ dollars. How can you make change for 18 dollars with as few coins as possible? Provide a general solution to the problem. This problem is known in computer science as the CHANGE MAKING problem, and can be solved with a recurrence similar to (3.2).

3.2 Compute the residue table and the Frobenius number for the weighted alphabet $\Sigma = \{3,6,20\}$. How can you "make change" for 41 "dollars"? This particular problem is also known as CHICKEN MCNUGGETS problem — explain why.

```
1: procedure FindAllIt(mass m)
2:     compomer c = (c_1, ..., c_k) ← 0
3:     integer i ← k
4:     while i ≤ k do
5:         if B[i, m] = 0 then                          ▷ is this decomposable at all?
6:             while i ≤ k and B[i, m] = 0 do           ▷ no, go to next one
7:                 m ← m + c_i a_i
8:                 c_i ← 0
9:                 i ← i + 1
10:            end while                                ▷ now, B[i, m] = 1 holds
11:            if i ≤ k then
12:                m ← m − a_i
13:                c_i ← c_i + 1
14:            end if
15:        else
16:            while i > 1 and B[i − 1, m] = 1 do       ▷ initially, we do not add any coins
17:                i ← i − 1
18:            end while                                ▷ now, B[i, m] = 1 but B[i − 1, m] = 0
19:            if i = 1 then
20:                c_1 ← m/a_1
21:                Output c = (c_1, ..., c_k)
22:                i ← 2
23:            end if
24:            if i ≤ k then
25:                m ← m − a_i
26:                c_i ← c_i + 1
27:            end if
28:        end if
29:    end while
30: end procedure
```

Algorithm 3.5: Iterative algorithm for enumerating all compomers of a given mass m. To decompose mass M, this algorithm is initially called as FINDALLIT(M). **[TODO: THIS HAS TO BE CHECKED!]**

3.3 You can compute Frobenius numbers using the search engine Wolfram Alpha at `http://www.wolframalpha.com/`. What is the Frobenius number of the alphabet $\{12312312, 4567456745, 678678678, 4567894567\}$?

3.4 Show by examples that the greedy algorithm cannot optimally solve the MONEY CHANGING and the CHANGE MAKING problem.

3.5 How many strings can be made using all characters of the string ALGORITHMUS exactly once? As an example, there are three strings for the input string ABA, namely AAB, ABA, and BAA. How many strings can be made from ABRACADABRA? Try to find a formula for this number.

3.6 Let Σ be a weighted alphabet with integer masses $\mu : \Sigma \to \mathbb{N}_{>0}$, where not all masses are necessarily different. Build an algorithm that decomposes some mass M over this alphabet, using any of the FINDALL algorithms as a subroutine.

3.7 Let $\Sigma = \{a,b,c,d\}$ be a weighted alphabet with masses $\mu(a) = 3$, $\mu(b) = 6$, $\mu(c) = 8$, and $\mu(d) = 9$. Compute all compomers using the recursive algorithm FINDALLREC (Alg. 3.2). List all calls of the the algorithm, in the order in which they are executed. Here, we use the "old fashioned" version of weighted alphabets, to make it easier to write up the compomers.

3.8 Let Σ be the weighted alphabet from the previous exercise. Compute all compomers using the iterative algorithm FINDALLIT (Alg. 3.5). List values of variables i, m, and c for each entry into the WHILE-loop (line 5), in the order in which the algorithm is executed.

3.9 Let $\Sigma := \{6, 7, 17, 22\}$ be a weighted alphabet. Compute the ERT table using the Round Robin algorithm, and use the ERT table to compute all decompositions of mass 35. Compute the Frobenius number and the number of omitted values of this instance.

3.10 Modify Alg. 3.5 so that it uses the Extended Residue Table instead of array B, similar to Algorithms 3.2 and 3.4.

3.11 Assume that we have computed $C'[m]$ for all $m = 0, \ldots, M$ as the number of strings over an alphabet Σ. How many strings of parent mass M have a prefix of mass m, and how many have a suffix of mass m? Finally, how many strings of parent mass M have a prefix or suffix (or both) of mass $m \leq M/2$? Hint: The solution to all three questions is very simple and, in particular, you do not need a new recurrence.

3.12 Using integer masses, find the prefix and suffix of a peptide with smallest mass so that, with the "true" *de novo* sequencing mass modification ± 0 and $+18$, both have identical mass, violating Assumption 4 from Chapter 2. Argue why the string resulting from appending prefix and suffix, is truly the string of smallest mass violating the assumption.

3.13* Using integer masses, count the number of peptide strings of mass M that have a prefix of mass m and a suffix of mass $m+18$, for any $m \in \{0, \ldots, M\}$. If you have previously computed $C'[0 \ldots M]$, you can do so in $O(M^2)$ time. To come up with useful numbers, you should treat character I and L as one; similarly, characters K and Q. Plot the relative number of such strings against M, for $M = 0, \ldots, 3500$.

3.14* Proof that for a weighted alphabet $\{a_1, a_2\}$ that the number $g = g(a_1, a_2) = a_1 a_2 - a_1 - a_2$ cannot be decomposed, but *all* $M > g$ can be decomposed.

3.15 Proof the correctness of Eq. (3.6).

3.16 Write a program to compute arrays B and C for the amino acid alphabet. As integer weights, use those from Table 9.1 times 100, rounded to the closest integer. Compute $B[0\ldots20000]$ and $C[0\ldots20000]$.

3.17 Implement the recursive and iterative algorithms for enumerating compomers, Algorithms 3.2 and 3.5, as well as 19 nested WHILE-loops for the amino acid alphabet. Using each algorithm, enumerate all compomers for integer masses $m = 0,\ldots,20000$, using the array B from the previous exercise. Compare running times. Warning: Do not print out compomers, as this will by far exceed the time required to compute them.

3.18 It is easy to modify all presented algorithms for enumerating compomers, when upper and lower bounds for each character in Σ are given. Show how this can be done. Note that for lower bounds, you do not need *any* changes to the actual algorithms.

3.19 Assume that Σ is an alphabet of integer masses, such that one character has positive mass, and one character has negative mass. Proof: If some mass M has at least one decomposition, then it has an infinite number of decompositions.

3.20 Let $f(n) := 2^n + n + 1000$ and $g(n) := 2^n$. Show that $f \sim g$. Compute the absolute and relative error for $n = 1,\ldots,20$.

Bibliography

[1] A. Aaant. I need a title, quick. **[TODO: REPLACE WITH A REAL CITATION]**, 2101.

[2] G. Alves, A. Y. Ogurtsov and Y.-K. Yu. RAId_DbS: peptide identification using database searches with realistic statistics. *Biol. Direct.*, 2:25, 2007.

[3] S. Andreotti, G. W. Klau and K. Reinert. Antilope – a lagrangian relaxation approach to the *de novo* peptide sequencing problem. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 2011. To appear, doi:10.1109/TCBB.2011.59.

[4] R. Apweiler, H. Hermjakob and N. Sharon. On the frequency of protein glycosylation, as deduced from analysis of the SWISS-PROT database. *Biochim. Biophys. Acta*, 1473(1): 4–8, 1999.

[5] G. Audi, A. Wapstra and C. Thibault. The AME2003 atomic mass evaluation (ii): Tables, graphs, and references. *Nucl. Phys. A*, 729:129–336, 2003.

[6] J.-M. Autebert, J. Berstel and L. Boasson. Context-free languages and pushdown automata. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 1, pages 111–174. Springer, 1997.

[7] V. Bafna and N. Edwards. SCOPE: A probabilistic model for scoring tandem mass spectra against a peptide database. *Bioinformatics*, 17:S13–S21, 2001.

[8] D. A. Barkauskas and D. M. Rocke. A general-purpose baseline estimation algorithm for spectroscopic data. *Anal. Chim. Acta*, 657(2):191–197, 2010.

[9] C. Bartels. Fast algorithm for peptide sequencing by mass spectrometry. *Biomed. Environ. Mass Spectrom.*, 19:363–368, 1990.

[10] J. M. S. Bartlett and D. Stirling. A short history of the polymerase chain reaction. *Methods Mol. Biol.*, 226:3–6, 2003.

[11] C. Bauer, R. Cramer and J. Schuchhardt. Evaluation of peak-picking algorithms for protein mass spectrometry. *Methods Mol. Biol.*, 696:341–352, 2011.

[12] M. Beck, I. M. Gessel and T. Komatsu. The polynomial part of a restricted partition function related to the Frobenius problem. *Electron. J. Comb.*, 8(1):N7, 2001.

[13] D. E. Beihoffer, J. Hendry, A. Nijenhuis and S. Wagon. Faster algorithms for Frobenius numbers. *Electron. J. Comb.*, 12:R27, 2005.

[14] C. Benecke, T. Grüner, A. Kerber, R. Laue and T. Wieland. MOLecular Structure GENeration with MOLGEN, new features and future developments. *Anal. Chim. Acta*, 314:141–147, 1995.

[15] G. Benson. Composition alignment. In *Proc. of Workshop on Algorithms in Bioinformatics (WABI 2003)*, volume 2812 of *Lect. Notes Comput. Sc.*, pages 447–461. Springer, 2003.

[16] M. W. Bern and D. Goldberg. EigenMS: De novo analysis of peptide tandem mass spectra by spectral graph partitioning. In *Proc. of Research in Computational Molecular Biology (RECOMB 2005)*, volume 3500 of *Lect. Notes Comput. Sc.*, pages 357–372. Springer, 2005.

[17] M. W. Bern and D. Goldberg. De novo analysis of peptide tandem mass spectra by spectral graph partitioning. *J. Comput. Biol.*, 13(2):364–378, 2006.

[18] A. Bertsch, A. Leinenbach, A. Pervukhin, M. Lubeck, R. Hartmer, C. Baessmann, Y. A. Elnakady, R. Müller, S. Böcker, C. G. Huber, and O. Kohlbacher. De novo peptide sequencing by tandem MS using complementary CID and electron transfer dissociation. *Electrophoresis*, 30(21):3736–3747, 2009.

[19] K. Biemann, C. Cone and B. R. Webster. Computer-aided interpretation of high-resolution mass spectra. II. Amino acid sequence of peptides. *J. Am. Chem. Soc.*, 88(11):2597–2598, 1966.

[20] K. Biemann, C. Cone, B. R. Webster and G. P. Arsenault. Determination of the amino acid sequence in oligopeptides by computer interpretation of their high-resolution mass spectra. *J. Am. Chem. Soc.*, 88(23):5598–5606, 1966.

[21] A. Björklund, T. Husfeldt, P. Kaski and M. Koivisto. Fourier meets Möbius: fast subset convolution. In *Proc. of ACM Symposium on Theory of Computing (STOC 2007)*, pages 67–74. ACM Press New York, 2007.

[22] N. Blow. Glycobiology: A spoonful of sugar. *Nature*, 457(7229):617–620, 2009.

[23] S. Böcker. Sequencing from compomers: Using mass spectrometry for DNA de-novo sequencing of 200+ nt. *J. Comput. Biol.*, 11(6):1110–1134, 2004.

[24] S. Böcker and Zs. Lipták. A fast and simple algorithm for the Money Changing Problem. *Algorithmica*, 48(4):413–432, 2007.

[25] S. Böcker and V. Mäkinen. Combinatorial approaches for mass spectra recalibration. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 5(1):91–100, 2008.

[26] S. Böcker and F. Rasche. Towards de novo identification of metabolites by analyzing tandem mass spectra. *Bioinformatics*, 24:I49–I55, 2008. Proc. of *European Conference on Computational Biology* (ECCB 2008).

[27] S. Böcker, M. Letzel, Zs. Lipták and A. Pervukhin. Decomposing metabolomic isotope patterns. In *Proc. of Workshop on Algorithms in Bioinformatics (WABI 2006)*, volume 4175 of *Lect. Notes Comput. Sc.*, pages 12–23. Springer, 2006.

[28] S. Böcker, B. Kehr and F. Rasche. Determination of glycan structure from tandem mass spectra. In *Proc. of Computing and Combinatorics Conference (COCOON 2009)*, volume 5609 of *Lect. Notes Comput. Sc.*, pages 258–267. Springer, 2009.

[29] S. Böcker, M. Letzel, Zs. Lipták and A. Pervukhin. SIRIUS: Decomposing isotope patterns for metabolite identification. *Bioinformatics*, 25(2):218–224, 2009.

[30] S. Böcker, F. Rasche and T. Steijger. Annotating fragmentation patterns. In *Proc. of Workshop on Algorithms in Bioinformatics (WABI 2009)*, volume 5724 of *Lect. Notes Comput. Sc.*, pages 13–24. Springer, 2009.

[31] A. Brauer and J. E. Shockley. On a problem of Frobenius. *J. Reine Angew. Math.*, 211: 215–220, 1962.

[32] R. Breitling, A. R. Pitt and M. P. Barrett. Precision mapping of the metabolome. *Trends Biotechnol.*, 24(12):543–548, 2006.

[33] K. Q. Brown. *Geometric transforms for fast geometric algorithms*. Report cmucs-80-101, Dept. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, USA, 1980.

[34] S. Cappadona, P. Nanni, M. Benevento, F. Levander, P. Versura, A. Roda, S. Cerutti, and L. Pattini. Improved label-free LC-MS analysis by wavelet-based noise rejection. *J Biomed Biotechnol*, 2010:131505, 2010.

[35] A. Ceroni, K. Maass, H. Geyer, R. Geyer, A. Dell and S. M. Haslam. GlycoWorkbench: a tool for the computer-assisted annotation of mass spectra of glycans. *J. Proteome Res.*, 7 (4):1650–1659, 2008.

[36] D. C. Chamrad, G. Körting, K. Stühler, H. E. Meyer, J. Klose and M. Blüggel. Evaluation of algorithms for protein identification from sequence databases using mass spectrometry data. *Proteomics*, 4:619–628, 2004.

[37] S. Chattopadhyay and P. Das. The K-dense corridor problems. *Pattern Recogn. Lett.*, 11 (7):463–469, 1990.

[38] E. Check. Proteomics and cancer: Running before we can walk? *Nature*, 429:496–497, 2004.

[39] T. Chen, M.-Y. Kao, M. Tepel, J. Rush and G. M. Church. A dynamic programming approach to de novo peptide sequencing via tandem mass spectrometry. *J. Comput. Biol.*, 8(3):325–337, 2001. Preliminary version in *Proc. of Symposium on Discrete Algorithms (SODA 2000)*, Association for Computing Machinery, 2000, 389–398.

[40] W. L. Chen. Chemoinformatics: past, present, and future. *J. Chem. Inf. Model.*, 46(6): 2230–2255, 2006.

[41] F. Y. Chin, C. A. Wang and F. L. Wang. Maximum stabbing line in 2D plane. In *Proc. of Conf. on Computing and Combinatorics (COCOON 1999)*, volume 1627 of *Lect. Notes Comput. Sc.*, pages 379–388. Springer, 1999.

[42] H. H. Chou, H. Takematsu, S. Diaz, J. Iber, E. Nickerson, K. L. Wright, E. A. Muchmore, D. L. Nelson, S. T. Warren, and A. Varki. A mutation in human CMP-sialic acid hydroxylase occurred after the Homo-Pan divergence. *Proc. Natl. Acad. Sci. U. S. A.*, 95(20):11751–11756, 1998.

[43] Y. Chu and T. Liu. On the shortest arborescence of a directed graph. *Sci. Sinica*, 14: 1396–1400, 1965.

[44] K. R. Clauser, P. Baker and A. L. Burlingame. Role of accurate mass measurement (+/− 10 ppm) in protein identification strategies employing MS or MS/MS and database searching. *Anal. Chem.*, 71(14):2871–2882, 1999.

[45] C. A. Cooper, E. Gasteiger and N. H. Packer. GlycoMod – a software tool for determining glycosylation compositions from mass spectrometric data. *Proteomics*, 1(2):340–349, 2001.

[46] C. A. Cooper, H. J. Joshi, M. J. Harrison, M. R. Wilkins and N. H. Packer. GlycoSuiteDB: a curated relational database of glycoprotein glycan structures and their biological sources. 2003 update. *Nucleic Acids Res.*, 31(1):511–513, 2003.

[47] R. Craig and R. C. Beavis. Tandem: matching proteins with tandem mass spectra. *Bioinformatics*, 20(9):1466–1467, 2004.

[48] V. Dančík, T. A. Addona, K. R. Clauser, J. E. Vath and P. A. Pevzner. De novo peptide sequencing via tandem mass spectrometry: A graph-theoretical approach. *J. Comput. Biol.*, 6(3/4):327–342, 1999. Preliminary version in *Proc. of Research in Computational Molecular Biology (RECOMB 1999)*, 135–144.

[49] C. Dass. *Principles and practice of biological mass spectrometry*. John Wiley and Sons, 2001.

[50] R. Datta and M. W. Bern. Spectrum fusion: using multiple mass spectra for de novo peptide sequencing. *J. Comput. Biol.*, 16(8):1169–1182, 2009.

[51] J. L. Davison. On the linear diophantine problem of Frobenius. *J. Number Theory*, 48(3): 353–363, 1994.

[52] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, second edition, 2000.

[53] E. de Hoffmann and V. Stroobant. *Mass Spectrometry: Principles and Applications*. Wiley-Interscience, third edition, 2007.

[54] J. R. de Laeter, J. K. Böhlke, P. D. Biévre, H. Hidaka, H. S. Peiser, K. J. R. Rosman and P. D. P. Taylor. Atomic weights of the elements. Review 2000 (IUPAC technical report). *Pure Appl. Chem.*, 75(6):683–800, 2003.

[55] E. W. Deutsch, H. Lam and R. Aebersold. Data analysis and bioinformatics tools for tandem mass spectrometry in proteomics. *Physiological Genomics*, 33:18–25, 2008.

[56] P. A. DiMaggio and C. A. Floudas. De novo peptide identification via tandem mass spectrometry and integer linear optimization. *Anal. Chem.*, 79(4):1433–1446, 2007.

[57] B. Domon and R. Aebersold. Mass spectrometry and protein analysis. *Science*, 312:212–217, 2006.

[58] B. Domon and C. E. Costello. A systematic nomenclature for carbohydrate fragmentations in FAB-MS/MS spectra of glycoconjugates. *Glycoconjugate J.*, 5:397–409, 1988.

[59] R. Dondi, G. Fertin and S. Vialette. Complexity issues in vertex-colored graph pattern matching. *J. Discrete Algorithms*, 2010. In press, doi:10.1016/j.jda.2010.09.002.

[60] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[61] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1972.

[62] M. Dyer. Approximate counting by dynamic programming. In *Proc. of Symposium on Theory of Computing (STOC 2003)*, pages 693–699, 2003.

[63] S. R. Eddy. "antedisciplinary" science. *PLoS Comput. Biol.*, 1(1):e6, 2005.

[64] P. Edman. Method for determination of the amino acid sequence in peptides. *Acta Chem. Scand.*, 4:283–293, 1950.

[65] J. Edmonds. Optimum branchings. *J. Res. Nat. Bur. Stand.*, 71B:233–240, 1967.

[66] M. Ehrich, S. Böcker and D. van den Boom. Multiplexed discovery of sequence polymorphisms using base-specific cleavage and MALDI-TOF MS. *Nucleic Acids Res.*, 33(4):e38, 2005.

[67] D. Einstein, D. Lichtblau, A. Strzebonski and S. Wagon. Frobenius numbers by lattice point enumeration. *INTEGERS*, 7(1):#A15, 2007.

[68] J. E. Elias and S. P. Gygi. Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. *Nat. Methods*, 4(3):207–214, 2007.

[69] J. E. Elias, F. D. Gibbons, O. D. King, F. P. Roth and S. P. Gygi. Intensity-based protein identification by machine learning from a library of tandem mass spectra. *Nat. Biotechnol.*, 22(2):214–219, 2004.

[70] J. K. Eng, A. L. McCormack and J. R. Yates III. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *J. Am. Soc. Mass Spectr.*, 5:976–989, 1994.

[71] M. Ethier, J. A. Saba, M. Spearman, O. Krokhin, M. Butler, W. Ens, K. G. Standing, and H. Perreault. Application of the StrOligo algorithm for the automated structure assignment of complex N-linked glycans from glycoproteins using tandem mass spectrometry. *Rapid Commun. Mass Spectrom.*, 17(24):2713–2720, 2003.

[72] M. Fellows, G. Fertin, D. Hermelin and S. Vialette. Sharp tractability borderlines for finding connected motifs in vertex-colored graphs. In *Proc. of International Colloquium on Automata, Languages and Programming (ICALP 2007)*, volume 4596 of *Lect. Notes Comput. Sc.*, pages 340–351. Springer, 2007.

[73] J. Fenn, M. Mann, C. Meng, S. Wong and C. Whitehouse. Electrospray ionisation for mass spectrometry of large biomolecules. *Science*, 246:64–71, 1989.

[74] D. Fenyö and R. C. Beavis. A method for assessing the statistical significance of mass spectrometry-based protein identifications using general scoring schemes. *Anal. Chem.*, 75(4):768–774, 2003.

[75] J. Fernández-de-Cossío, L. J. Gonzalez and V. Besada. A computer program to aid the sequencing of peptides in collision-activated decomposition experiments. *Comput. Appl. Biosci.*, 11(4):427–434, 1995.

[76] J. Fernández-de-Cossío, J. Gonzalez, T. Takao, Y. Shimonishi, G. Padron and V. Besada. A software program for the rapid sequence analysis of unknown peptides involving modifications, based on MS/MS data. In *ASMS Conf. on Mass Spectrometry and Allied Topics, Slot 074*, 1997.

[77] J. Fernández-de-Cossío, L. J. Gonzalez, Y. Satomi, L. Betancourt, Y. Ramos, V. Huerta, A. Amaro, V. Besada, G. Padron, N. Minamino, and T. Takao. Isotopica: a tool for the calculation and viewing of complex isotopic envelopes. *Nucleic Acids Res.*, 32(Web Server issue):W674–W678, 2004.

[78] A. R. Fernie, R. N. Trethewey, A. J. Krotzky and L. Willmitzer. Metabolite profiling: from diagnostics to systems biology. *Nat. Rev. Mol. Cell Biol.*, 5(9):763–769, 2004.

[79] H. I. Field, D. Fenyö and R. C. Beavis. RADARS, a bioinformatics solution that automates proteome mass spectral analysis, optimises protein identification, and archives data in a relational database. *Proteomics*, 2(1):36–47, 2002.

[80] B. Fischer, V. Roth, F. Roos, J. Grossmann, S. Baginsky, P. Widmayer, W. Gruissem, and J. M. Buhmann. NovoHMM: a hidden Markov model for de novo peptide sequencing. *Anal. Chem.*, 77(22):7265–7273, 2005.

[81] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009. Freely available from `http://algo.inria.fr/flajolet/Publications/book.pdf`.

[82] A. Frank and P. Pevzner. PepNovo: de novo peptide sequencing via probabilistic network modeling. *Anal. Chem.*, 15:964–973, 2005.

[83] A. M. Frank, M. M. Savitski, M. N. Nielsen, R. A. Zubarev and P. A. Pevzner. De novo peptide sequencing and identification with precision mass spectrometry. *J. Proteome Res.*, 6(1):114–123, 2007.

[84] A. Fürst, J.-T. Clerc and E. Pretsch. A computer program for the computation of the molecular formula. *Chemom. Intell. Lab. Syst.*, 5:329–334, 1989.

[85] V. A. Fusaro, D. R. Mani, J. P. Mesirov and S. A. Carr. Prediction of high-responding peptides for targeted protein assays by mass spectrometry. *Nat. Biotechnol.*, 27(2):190–198, 2009.

[86] H. Gabow, Z. Galil, T. Spencer and R. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6:109–122, 1986.

[87] M. R. Garey and D. S. Johnson. *Computers and Intractability (A Guide to Theory of NP-Completeness)*. Freeman, New York, 1979.

[88] J. Gasteiger, W. Hanebeck and K.-P. Schulz. Prediction of mass spectra from structural information. *J. Chem. Inf. Comput. Sci.*, 32(4):264–271, 1992.

[89] S. P. Gaucher, J. Morrow and J. A. Leary. STAT: a saccharide topology analysis tool used in combination with tandem mass spectrometry. *Anal. Chem.*, 72(11):2331–2336, 2000.

[90] L. Y. Geer, S. P. Markey, J. A. Kowalak, L. Wagner, M. Xu, D. M. Maynard, X. Yang, W. Shi, and S. H. Bryant. Open mass spectrometry search algorithm. *J. Proteome Res.*, 3:958–964, 2004.

[91] P. Gilmore and R. Gomory. Multi-stage cutting stock problems of two and more dimensions. *Oper. Res.*, 13(1):94–120, 1965.

[92] D. Goldberg, M. Sutton-Smith, J. Paulson and A. Dell. Automatic annotation of matrix-assisted laser desorption/ionization N-glycan spectra. *Proteomics*, 5(4):865–875, 2005.

[93] D. Goldberg, M. W. Bern, B. Li and C. B. Lebrilla. Automatic determination of O-glycan structure from fragmentation spectra. *J. Proteome Res.*, 5(6):1429–1434, 2006.

[94] D. Goldberg, M. W. Bern, S. Parry, M. Sutton-Smith, M. Panico, H. R. Morris and A. Dell. Automated N-glycopeptide identification using a combination of single- and tandem-MS. *J. Proteome Res.*, 6(10):3995–4005, 2007.

[95] D. Goldberg, M. W. Bern, S. J. North, S. M. Haslam and A. Dell. Glycan family analysis for deducing N-glycan topology from single MS. *Bioinformatics*, 25(3):365–371, 2009.

[96] A. H. Grange, M. C. Zumwalt and G. W. Sovocool. Determination of ion and neutral loss compositions and deconvolution of product ion mass spectra using an orthogonal acceleration time-of-flight mass spectrometer and an ion correlation program. *Rapid Commun. Mass Spectrom.*, 20(2):89–102, 2006.

[97] N. A. Gray. Applications of artificial intelligence for organic chemistry: Analysis of C-13 spectra. *Artificial Intelligence*, 22(1):1–21, 1984.

[98] N. A. B. Gray, R. E. Carhart, A. Lavanchy, D. H. Smith, T. Varkony, B. G. Buchanan, W. C. White, and L. Creary. Computerized mass spectrum prediction and ranking. *Anal. Chem.*, 52(7):1095–1102, 1980.

[99] N. A. B. Gray, A. Buchs, D. H. Smith and C. Djerassi. Computer assisted structural interpretation of mass spectral data. *Helv. Chim. Acta*, 64(2):458–470, 1981.

[100] H. Greenberg. Solution to a linear diophantine equation for nonnegative integers. *J. Algorithms*, 9(3):343–353, 1988.

[101] D. H. Greene and D. E. Knuth. *Mathematics for the Analysis of Algorithms*, volume 1 of *Progress in Computer Science and Applied Logic (PCS)*. Birkhäuser Boston, 1990.

[102] J. Gross. *Mass Spectrometry: A textbook*. Springer, Berlin, 2004.

[103] K. Grützmann, S. Böcker and S. Schuster. Combinatorics of aliphatic amino acids. *Naturwissenschaften*, 98(1):79–86, 2011.

[104] M. Guilhaus. Principles and instrumentation in time-of-flight mass spectrometry. *J. Mass Spectrom.*, 30:1519–1532, 1995.

[105] S. Guillemot and F. Sikora. Finding and counting vertex-colored subtrees. In *Proc. of Symposium on Mathematical Foundations of Computer Science (MFCS 2010)*, volume 6281 of *Lect. Notes Comput. Sc.*, pages 405–416. Springer, 2010.

[106] C. Hamm, W. Wilson and D. Harvan. Peptide sequencing program. *Comput. Appl. Biosci.*, 2:115–118, 1986.

[107] F. Harary, R. W. Robinson and A. J. Schwenk. Twenty-step algorithm for determining the asymptotic number of trees of various species. *J. Austral. Math. Soc.*, 20(Series A): 483–503, 1975.

[108] M. Havilio, Y. Haddad and Z. Smilansky. Intensity-based statistical scorer for tandem mass spectrometry. *Anal. Chem.*, 75:435–444, 2003.

[109] M. Heinonen, A. Rantanen, T. Mielikäinen, J. Kokkonen, J. Kiuru, R. A. Ketola and J. Rousu. FiD: a software for ab initio structural identification of product ions from tandem mass spectrometric data. *Rapid Commun. Mass Spectrom.*, 22(19):3043–3052, 2008.

[110] D. W. Hill, T. M. Kertesz, D. Fontaine, R. Friedman and D. F. Grant. Mass spectral metabonomics beyond elemental formula: Chemical database querying by matching experimental with computational fragmentation spectra. *Anal. Chem.*, 80(14):5574–5582, 2008.

[111] W. M. Hines, A. M. Falick, A. L. Burlingame and B. W. Gibson. Pattern-based algorithm for peptide sequencing from tandem high energy collision-induced dissociation mass spectra. *J. Am. Soc. Mass Spectrom.*, 3(4):326 – 336, 1992.

[112] C. A. R. Hoare. FIND (algorithm 65). *Communications of the ACM*, 4:321–322, 1961.

[113] D. H. Horn, R. A. Zubarev and F. W. McLafferty. Automated reduction and interpretation of high resolution electrospray mass spectra of large molecules. *J. Am. Soc. Mass Spectr.*, 11:320–332, 2000.

[114] C. S. Hsu. Diophantine approach to isotopic abundance calculations. *Anal. Chem.*, 56(8): 1356–1361, 1984.

[115] Q. Hu, R. J. Noll, H. Li, A. Makarov, M. Hardman and R. G. Cooks. The Orbitrap: a new mass spectrometer. *J. Mass Spectrom.*, 40(4):430–443, 2005.

[116] R. Hussong and A. Hildebrandt. Signal processing in proteomics. *Methods Mol. Biol.*, 604: 145–161, 2010.

[117] N. Jaitly, M. E. Monroe, V. A. Petyuk, T. R. W. Clauss, J. N. Adkins and R. D. Smith. Robust algorithm for alignment of liquid chromatography-mass spectrometry analyses in an accurate mass and time tag data analysis pipeline. *Anal. Chem.*, 78(21):7397–7409, 2006.

[118] N. Jeffries. Algorithms for alignment of mass spectrometry proteomic data. *Bioinformatics*, 21(14):3066–3073, 2005.

[119] R. S. Johnson and J. A. Taylor. Searching sequence databases via de novo peptide sequencing by tandem mass spectrometry. *Methods Mol. Biol.*, 146:41–61, 2000.

[120] R. S. Johnson and J. A. Taylor. Searching sequence databases via de novo peptide sequencing by tandem mass spectrometry. *Mol. Biotechnol.*, 22(3):301–315, 2002.

[121] P. Jones, R. G. Côté, L. Martens, A. F. Quinn, C. F. Taylor, W. Derache, H. Hermjakob, and R. Apweiler. PRIDE: a public repository of protein and peptide identifications for the proteomics community. *Nucleic Acids Res.*, 34(Database-Issue):659–663, 2006.

[122] H. J. Joshi, M. J. Harrison, B. L. Schulz, C. A. Cooper, N. H. Packer and N. G. Karlsson. Development of a mass fingerprinting tool for automated interpretation of oligosaccharide fragmentation data. *Proteomics*, 4(6):1650–1664, 2004.

[123] L. Käll, J. D. Canterbury, J. Weston, W. S. Noble and M. J. MacCoss. Semi-supervised learning for peptide identification from shotgun proteomics datasets. *Nat. Methods*, 4(11): 923–925, 2007.

[124] M. Kanehisa, S. Goto, M. Hattori, K. F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa. From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Res.*, 34:D354–D357, 2006.

[125] R. Kannan. Lattice translates of a polytope and the Frobenius problem. *Combinatorica*, 12:161–177, 1991.

[126] E. A. Kapp, F. Schütz, L. M. Connolly, J. A. Chakel, J. E. Meza, C. A. Miller, D. Fenyo, J. K. Eng, J. N. Adkins, G. S. Omenn, and R. J. Simpson. An evaluation, comparison, and accurate benchmarking of several publicly available MS/MS search algorithms: Sensitivity and specificity analysis. *Proteomics*, 5:3475–3490, 2005.

[127] M. Karas and F. Hillenkamp. Laser desorption ionization of proteins with molecular masses exceeding 10,000 Daltons. *Anal. Chem.*, 60:2299–2301, 1988.

[128] A. Keller, A. I. Nesvizhskii, E. Kolker and R. Aebersold. Empirical statistical model to estimate the accuracy of peptide identifications made by MS/MS and database search. *Anal. Chem.*, 74(20):5383–5392, 2002.

[129] A. Keller, J. Eng, N. Zhang, X.-J. Li and R. Aebersold. A uniform proteomics MS/MS analysis platform utilizing open XML file formats. *Mol. Syst. Biol.*, 1:2005.0017, 2005.

[130] E. Kendrick. A mass scale based on CH2 = 14.0000 for high resolution mass spectrometry of organic compounds. *Anal. Chem.*, 35(13):2146–2154, 1963.

[131] A. Kerber, R. Laue and D. Moser. Ein Strukturgenerator für molekulare Graphen. *Anal. Chim. Acta*, 235:221 – 228, 1990.

[132] A. Kerber, R. Laue, M. Meringer and C. Rücker. Molecules in silico: The generation of structural formulae and its applications. *J. Comput. Chem. Japan*, 3(3):85–96, 2004.

[133] S. Kim, N. Gupta and P. A. Pevzner. Spectral probabilities and generating functions of tandem mass spectra: a strike against decoy databases. *J. Proteome Res.*, 7(8):3354–3363, 2008.

[134] S. Kim, N. Bandeira and P. A. Pevzner. Spectral profiles, a novel representation of tandem mass spectra and their applications for de novo peptide sequencing and identification. *Mol. Cell. Proteomics*, 8(6):1391–1400, 2009.

[135] S. Kim, N. Gupta, N. Bandeira and P. A. Pevzner. Spectral dictionaries: Integrating de novo peptide sequencing with database search of tandem mass spectra. *Mol. Cell. Proteomics*, 8(1):53–69, 2009.

[136] T. Kind and O. Fiehn. Metabolomic database annotations via query of elemental compositions: Mass accuracy is insufficient even at less than 1 ppm. *BMC Bioinformatics*, 7(1): 234, 2006.

[137] T. Kind and O. Fiehn. Seven golden rules for heuristic filtering of molecular formulas obtained by accurate mass spectrometry. *BMC Bioinformatics*, 8:105, 2007.

[138] H. Kubinyi. Calculation of isotope distributions in mass spectrometry: A trivial solution for a non-trivial problem. *Anal. Chim. Acta*, 247:107–119, 1991.

[139] K.-S. Kwok, R. Venkataraghavan and F. W. McLafferty. Computer-aided interpretation of mass spectra. III. Self-training interpretive and retrieval system. *J. Am. Chem. Soc.*, 95 (13):4185–4194, 1973.

[140] V. Lacroix, C. G. Fernandes, and M.-F. Sagot. Motif search in graphs: Application to metabolic networks. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 3(4):360–368, 2006.

[141] A. J. Lapadula, P. J. Hatcher, A. J. Hanneman, D. J. Ashline, H. Zhang and V. N. Reinhold. Congruent strategies for carbohydrate sequencing. 3. OSCAR: an algorithm for assigning oligosaccharide topology from MS^n data. *Anal. Chem.*, 77(19):6271–6279, 2005.

[142] R. L. Last, A. D. Jones and Y. Shachar-Hill. Towards the plant metabolome and beyond. *Nat. Rev. Mol. Cell Biol.*, 8:167–174, 2007.

[143] A. Lavanchy, T. Varkony, D. H. Smith, N. A. B. Gray, W. C. White, R. E. Carhart, B. G. Buchanan, and C. Djerassi. Rule-based mass spectrum prediction and ranking: Applications to structure elucidation of novel marine sterols. *Org. Mass Spectrom.*, 15(7): 355–366, 1980.

[144] J. Lederberg. Topological mapping of organic molecules. *Proc. Natl. Acad. Sci. U. S. A.*, 53 (1):134–139, 1965.

[145] J. Lederberg. How DENDRAL was conceived and born. In *ACM Conference on the History of Medical Informatics, History of Medical Informatics archive*, pages 5–19, 1987. Available from `http://doi.acm.org/10.1145/41526.41528`.

[146] T. A. Lee. *A Beginner's Guide to Mass Spectral Interpretation*. Wiley, 1998.

[147] M. Lefmann, C. Honisch, S. Boecker, N. Storm, F. von Wintzingerode, C. Schloetelburg, A. Moter, D. van den Boom, and U. B. Goebel. A novel mass spectrometry based tool for genotypic identification of mycobacteria. *J. Clin. Microbiol.*, 42(1):339–346, 2004.

[148] G. Li and F. Ruskey. The advantages of forward thinking in generating rooted and free trees. In *Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA 1999)*, pages 939–940, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[149] G. Liu, J. Zhang, B. Larsen, C. Stark, A. Breitkreutz, Z.-Y. Lin, B.-J. Breitkreutz, Y. Ding, K. Colwill, A. Pasculescu, T. Pawson, J. L. Wrana, A. I. Nesvizhskii, B. Raught, M. Tyers, and A.-C. Gingras. ProHits: integrated software for mass spectrometry-based interaction proteomics. *Nat. Biotechnol.*, 28(10):1015–1017, 2010.

[150] K. K. Lohmann and C.-W. von der Lieth. GlycoFragment and GlycoSearchMS: web tools to support the interpretation of mass spectra of complex carbohydrates. *Nucleic Acids Res.*, 32(Web Server issue):W261–W266, 2004.

[151] B. Lu and T. Chen. A suffix tree approach to the interpretation of tandem mass spectra: Applications to peptides of non-specific digestion and post-translational modifications. *Bioinformatics*, 19(Suppl 2):ii113–ii121, 2003. Proc. of *European Conference on Computational Biology* (ECCB 2003).

[152] A. Luedemann, K. Strassburg, A. Erban and J. Kopka. TagFinder for the quantitative analysis of gas chromatography–mass spectrometry (GC-MS)-based metabolite profiling experiments. *Bioinformatics*, 24(5):732–737, 2008.

[153] G. S. Lueker. Two NP-complete problems in nonnegative integer programming. Technical Report TR-178, Department of Electrical Engineering, Princeton University, 1975.

[154] Y.-R. Luo. *Handbook of Bond Dissociation Energies in Organic Compounds*. CRC Press, Boca Raton, 2003.

[155] B. Ma and G. Lajoie. Improving the de novo sequencing accuracy by combining two independent scoring functions in peaks software. Poster at the ASMS Conference on Mass Spectrometry and Allied Topics, 2005.

[156] B. Ma, K. Zhang, C. Hendrie, C. Liang, M. Li, A. Doherty-Kirby and G. Lajoie. PEAKS: powerful software for peptide de novo sequencing by tandem mass spectrometry. *Rapid Commun. Mass Spectrom.*, 17(20):2337–2342, 2003.

[157] B. Ma, K. Zhang and C. Liang. An effective algorithm for peptide de novo sequencing from MS/MS spectra. *J. Comput. Syst. Sci.*, 70:418–430, 2005.

[158] K. Maass, R. Ranzinger, H. Geyer, C.-W. von der Lieth and R. Geyer. "Glyco-peakfinder" – de novo composition analysis of glycoconjugates. *Proteomics*, 7(24):4435–4444, 2007.

[159] P. Mallick, M. Schirle, S. S. Chen, M. R. Flory, H. Lee, D. Martin, J. Ranish, B. Raught, R. Schmitt, T. Werner, B. Kuster, and R. Aebersold. Computational prediction of proteotypic peptides for quantitative proteomics. *Nat. Biotechnol.*, 25(1):125–131, 2007.

[160] M. Mann and M. Wilm. Error-tolerant identification of peptides in sequence databases by peptide sequence tags. *Anal. Chem.*, 66(24):4390–4399, 1994.

[161] S. Martello and P. Toth. An exact algorithm for large unbounded knapsack problems. *Oper. Res. Lett.*, 9(1):15–20, 1990.

[162] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Chichester, 1990.

[163] R. Matthiesen, J. Bunkenborg, A. Stensballe, O. N. Jensen, K. G. Welinder and G. Bauw. Database-independent, database-dependent, and extended interpretation of peptide mass spectra in VEMS V2.0. *Proteomics*, 4(9):2583–2593, 2004.

[164] R. Matthiesen, M. B. Trelle, P. Hojrup, J. Bunkenborg and O. N. Jensen. VEMS 3.0: algorithms and computational tools for tandem mass spectrometry based identification of post-translational modifications in proteins. *J. Proteome Res.*, 4(6):2338–2347, 2005.

[165] L. McHugh and J. W. Arthur. Computational methods for protein identification from mass spectrometry data. *PLoS Comput. Biol.*, 4(2):e12, 2008.

[166] P. E. Miller and M. B. Denton. The quadrupole mass filter: Basic operating concepts. *J. Chem. Educ.*, 63:617–622, 1986.

[167] L. Mo, D. Dutta, Y. Wan and T. Chen. MSNovo: a dynamic programming algorithm for de novo peptide sequencing via tandem mass spectrometry. *Anal. Chem.*, 79(13):4870–4878, 2007.

[168] E. Mostacci, C. Truntzer, H. Cardot and P. Ducoroy. Multivariate denoising methods combining wavelets and principal component analysis for mass spectrometry data. *Proteomics*, 10(14):2564–2572, 2010.

[169] I. K. Mun and F. W. Mclafferty. Computer methods of molecular structure elucidation from unknown mass spectra. In *Supercomputers in Chemistry*, ACS Symposium Series, chapter 9, pages 117–124. American Chemical Society, 1981.

[170] S. Na, J. Jeong, H. Park, K.-J. Lee and E. Paek. Unrestrictive identification of multiple post-translational modifications from tandem mass spectrometry using an error-tolerant algorithm based on an extended sequence tag approach. *Mol. Cell. Proteomics*, 7(12): 2452–2463, 2008.

[171] S. Neumann and S. Böcker. Computational mass spectrometry for metabolomics – a review. *Anal. Bioanal. Chem.*, 398(7):2779–2788, 2010.

[172] N. Nguyen, H. Huang, S. Oraintara and A. Vo. Mass spectrometry data processing using zero-crossing lines in multi-scale of Gaussian derivative wavelet. *Bioinformatics*, 26(18): i659–i665, 2010.

[173] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[174] J. A. November. *Digitizing life: the introduction of computers to biology and medicine*. PhD thesis, Princeton University, Princeton, USA, 2006.

[175] H. Oberacher, M. Pavlic, K. Libiseller, B. Schubert, M. Sulyok, R. Schuhmacher, E. Csaszar, and H. C. Köfeler. On the inter-instrument and inter-laboratory transferability of a tandem mass spectral reference library: 1. results of an austrian multicenter study. *J. Mass Spectrom.*, 44(4):485–493, 2009.

[176] H. Oberacher, M. Pavlic, K. Libiseller, B. Schubert, M. Sulyok, R. Schuhmacher, E. Csaszar, and H. C. Köfeler. On the inter-instrument and the inter-laboratory transferability of a tandem mass spectral reference library: 2. optimization and characterization of the search algorithm. *J. Mass Spectrom.*, 44(4):494–502, 2009.

[177] S. Orchard, L. Montechi-Palazzi, E. W. Deutsch, P.-A. Binz, A. R. Jones, N. Paton, A. Pizarro, D. M. Creasy, J. Wojcik, and H. Hermjakob. Five years of progress in the standardization of proteomics data: 4th annual spring workshop of the HUPO-proteomics standards initiative. *Proteomics*, 7:3436–3440, 2007.

[178] R. Otter. The number of trees. *The Annals of Mathematics*, 49(3):583–599, 1948.

[179] K. G. Owens. Application of correlation analysis techniques to mass spectral data. *Appl. Spectrosc. Rev.*, 27(1):1–49, 1992.

[180] N. H. Packer, C.-W. von der Lieth, K. F. Aoki-Kinoshita, C. B. Lebrilla, J. C. Paulson, R. Raman, P. Rudd, R. Sasisekharan, N. Taniguchi, and W. S. York. Frontiers in glycomics: bioinformatics and biomarkers in disease. An NIH white paper prepared from discussions by the focus groups at a workshop on the NIH campus, Bethesda MD (September 11-13, 2006). *Proteomics*, 8(1):8–20, 2008.

[181] G. Palmisano, D. Antonacci and M. R. Larsen. Glycoproteomic profile in wine: a 'sweet' molecular renaissance. *J. Proteome Res.*, 9(12):6148–6159, 2010.

[182] D. J. Pappin, P. Hojrup and A. Bleasby. Rapid identification of proteins by peptide-mass fingerprinting. *Curr. Biol.*, 3(6):327–332, 1993.

[183] C. Y. Park, A. A. Klammer, L. Käll, M. J. MacCoss and W. S. Noble. Rapid and accurate peptide identification from tandem mass spectra. *J. Proteome Res.*, 7(7):3022–3027, 2008.

[184] W. E. Parkins. The uranium bomb, the calutron, and the space-charge problem. *Physics Today*, 58(5):45–51, 2005.

[185] V. Pellegrin. Molecular formulas of organic compounds: the nitrogen rule and degree of unsaturation. *J. Chem. Educ.*, 60(8):626–633, 1983.

[186] D. N. Perkins, D. J. Pappin, D. M. Creasy and J. S. Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, 20(18):3551–3567, 1999.

[187] R. H. Perry, R. G. Cooks and R. J. Noll. Orbitrap mass spectrometry: instrumentation, ion motion and applications. *Mass Spectrom. Rev.*, 27(6):661–699, 2008.

[188] G. Pólya. Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen. *Acta Mathematica*, 68(1):145–254, 1937.

[189] S. C. Pomerantz, J. A. Kowalak and J. A. McCloskey. Determination of oligonucleotide composition from mass spectromectrically measured molecular weight. *J. Am. Soc. Mass Spectrom.*, 4:204–209, 1993.

[190] R. Raman, S. Raguram, G. Venkataraman, J. C. Paulson and R. Sasisekharan. Glycomics: an integrated systems approach to structure-function relationships of glycans. *Nat. Methods*, 2(11):817–824, 2005.

[191] J. L. Ramírez-Alfonsín. *The Diophantine Frobenius Problem*. Oxford University Press, 2005.

[192] J. L. Ramírez-Alfonsín. Complexity of the Frobenius problem. *Combinatorica*, 16(1):143–147, 1996.

[193] F. Rasche, A. Svatoš, R. K. Maddula, C. Böttcher and S. Böcker. Computing fragmentation trees from tandem mass spectrometry data. *Anal. Chem.*, 83:1243–1251, 2011.

[194] I. Rauf, F. Rasche and S. Böcker. Computing maximum colorful subtrees in practice. Manuscript. **[TODO: REMOVE OR UPDATE]**, 2011.

[195] A. L. Rockwood and P. Haimi. Efficient calculation of accurate masses of isotopic peaks. *J. Am. Soc. Mass Spectrom.*, 17(3):415–419, 2006.

[196] A. L. Rockwood, M. M. Kushnir and G. J. Nelson. Dissociation of individual isotopic peaks: Predicting isotopic distributions of product ions in MS^n. *J. Am. Soc. Mass Spectr.*, 14:311–322, 2003.

[197] A. L. Rockwood, J. R. Van Orman and D. V. Dearden. Isotopic compositions and accurate masses of single isotopic peaks. *J. Am. Soc. Mass Spectr.*, 15:12–21, 2004.

[198] P. Roepstorff and J. Fohlman. Proposal for a common nomenclature for sequence ions in mass spectra of peptides. *Biomed. Mass Spectrom.*, 11(11):601, 1984.

[199] S. Rogers, R. A. Scheltema, M. Girolami and R. Breitling. Probabilistic assignment of formulas to mass peaks in metabolomics experiments. *Bioinformatics*, 25(4):512–518, 2009.

[200] R. G. Sadygov and J. R. Yates III. A hypergeometric probability model for protein identification and validation using tandem mass spectral data and protein sequence databases. *Anal. Chem.*, 75(15):3792–3798, 2003.

[201] R. G. Sadygov, D. Cociorva and J. R. Yates III. Large-scale database searching using tandem mass spectra: looking up the answer in the back of the book. *Nat. Methods*, 1(3): 195–202, 2004.

[202] T. Sakurai, T. Matsuo, H. Matsuda and I. Katakuse. PAAS 3: A computer program to determine probable sequence of peptides from mass spectrometric data. *Biomed. Mass Spectrom.*, 11(8):396–399, 1984.

[203] A. Salomaa. Counting (scattered) subwords. *B. Euro. Assoc. Theo. Comp. Sci.*, 81:165–179, 2003.

[204] F. Sanger, S. Nicklen and A. R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proc. Natl. Acad. Sci. U.S.A.*, 74(12):5463–5467, 1977.

[205] M. M. Savitski, M. L. Nielsen, F. Kjeldsen and R. A. Zubarev. Proteomics-grade de novo sequencing approach. *J. Proteome Res.*, 4:2348–2354, 2005.

[206] K. Scheubert, F. Hufsky, F. Rasche and S. Böcker. Computing fragmentation trees from metabolite multiple mass spectrometry data. In *Proc. of Research in Computational Molecular Biology (RECOMB 2011)*, 2011. To be presented.

[207] J. Seidler, N. Zinn, M. E. Boehm and W. D. Lehmann. De novo sequencing of peptides by MS/MS. *Proteomics*, 10(4):634–649, 2010.

[208] J. Senior. Partitions and their representative graphs. *Am. J. Math.*, 73(3):663–689, 1951.

[209] B. Shan, B. Ma, K. Zhang and G. Lajoie. Complexities and algorithms for glycan sequencing using tandem mass spectrometry. *J. Bioinformatics and Computational Biology*, 6(1): 77–91, 2008.

[210] Q. Sheng, Y. Mechref, Y. Li, M. V. Novotny and H. Tang. A computational approach to characterizing bond linkages of glycan isomers using matrix-assisted laser desorption/ionization tandem time-of-flight mass spectrometry. *Rapid Commun. Mass Spectrom.*, 22(22):3561–3569, 2008.

[211] I. V. Shilov, S. L. Seymour, A. A. Patel, A. Loboda, W. H. Tang, S. P. Keating, C. L. Hunter, L. M. Nuwaysir, and D. A. Schaeffer. The paragon algorithm, a next generation search engine that uses sequence temperature values and feature probabilities to identify peptides from tandem mass spectra. *Mol. Cell. Proteomics*, 6(9):1638–1655, 2007.

[212] H. Shin, M. P. Sampat, J. M. Koomen and M. K. Markey. Wavelet-based adaptive denoising and baseline correction for MALDI TOF MS. *OMICS*, 14(3):283–295, 2010.

[213] F. Sikora. An (almost complete) state of the art around the graph motif problem. Technical report, Université Paris-Est, France, 2010. Available from `http://www-igm.univ-mlv.fr/~fsikora/pub/GraphMotif-Resume.pdf`.

[214] R. M. Silverstein, F. X. Webster and D. Kiemle. *Spectrometric Identification of Organic Compounds*. Wiley, 7$^{\text{th}}$ edition, 2005.

[215] G. Siuzdak. *The Expanding Role of Mass Spectrometry in Biotechnology*. MCC Press, second edition, 2006.

[216] D. H. Smith, N. A. Gray, J. G. Nourse and C. W. Crandell. The DENDRAL project: recent advances in computer-assisted structure elucidation. *Anal. Chim. Acta*, 133(4):471 – 497, 1981.

[217] R. K. Snider. Efficient calculation of exact mass isotopic distributions. *J. Am. Soc. Mass Spectrom.*, 18(8):1511–1515, 2007.

[218] H. M. Sobell. Actinomycin and DNA transcription. *Proc. Natl. Acad. Sci. U. S. A.*, 82(16): 5328–5331, 1985.

[219] H. Steen and M. Mann. The ABC's (and XYZ's) of peptide sequencing. *Nature Rev.*, 5: 699–711, 2004.

[220] M. T. Sykes and J. R. Williamson. Envelope: interactive software for modeling and fitting complex isotope distributions. *BMC Bioinformatics*, 9:446, 2008.

[221] J. J. Sylvester and W. J. Curran Sharp. Problem 7382. *Educational Times*, 37:26, 1884.

[222] D. L. Tabb, M. J. MacCoss, C. C. Wu, S. D. Anderson and J. R. Yates. Similarity among tandem mass spectra from proteomic experiments: detection, significance, and utility. *Anal. Chem.*, 75(10):2470–2477, 2003.

[223] H. Tang, Y. Mechref and M. V. Novotny. Automated interpretation of MS/MS spectra of oligosaccharides. *Bioinformatics*, 21 Suppl 1:i431–i439, 2005. Proc. of *Intelligent Systems for Molecular Biology* (ISMB 2005).

[224] S. Tanner, H. Shu, A. Frank, L.-C. Wang, E. Zandi, M. Mumby, P. A. Pevzner, and V. Bafna. Inspect: Identification of posttranslationally modified peptides from tandem mass spectra. *Anal. Chem.*, 77:4626–4639, 2005.

[225] J. A. Taylor and R. S. Johnson. Implementation and uses of automated de novo peptide sequencing by tandem mass spectrometry. *Anal. Chem.*, 73(11):2594–2604, 2001.

[226] J. A. Taylor and R. S. Johnson. Sequence database searches via de novo peptide sequencing by tandem mass spectrometry. *Rapid Commun. Mass Spectrom.*, 11:1067–1075, 1997.

[227] J. van Lint and R. Wilson. *A Course in Combinatorics*. Cambridge University Press, 2001.

[228] A. Varki, R. D. Cummings, J. D. Esko, H. H. Freeze, P. Stanley, C. R. Bertozzi, G. W. Hart, and M. E. Etzler, editors. *Essentials of Glycobiology*. Cold Spring Harbor Laboratory Press, second edition, 2009. Freely available from `http://www.ncbi.nlm.nih.gov/books/NBK1908/`.

[229] R. Venkataraghavan, F. W. McLafferty and G. E. van Lear. Computer-aided interpretation of mass spectra. *Org. Mass Spectrom.*, 2(1):1–15, 1969.

[230] C.-W. von der Lieth, A. Bohne-Lang, K. K. Lohmann and M. Frank. Bioinformatics for glycomics: status, methods, requirements and perspectives. *Brief. Bioinform.*, 5(2):164–178, 2004.

[231] S. A. Waksman and H. B. Woodruff. Bacteriostatic and bacteriocidal substances produced by soil actinomycetes. *Proc. Soc. Exper. Biol.*, 45:609–614, 1940.

[232] M. S. Waterman and M. Vingron. Rapid and accurate estimates of statistical significance for sequence data base searches. *Proc. Natl. Acad. Sci. U. S. A.*, 91(11):4625–4628, 1994.

[233] J. T. Watson and O. D. Sparkman. *Introduction to Mass Spectrometry: Instrumentation, Applications, and Strategies for Data Interpretation*. Wiley, 2007.

[234] M. E. Wieser. Atomic weights of the elements 2005 (IUPAC technical report). *Pure Appl. Chem.*, 78(11):2051–2066, 2006.

[235] H. Wilf. *generatingfunctionology*. Academic Press, second edition, 1994. Freely available from `http://www.math.upenn.edu/~wilf/DownldGF.html`.

[236] S. Wolf, S. Schmidt, M. Müller-Hannemann and S. Neumann. In silico fragmentation for computer assisted identification of metabolite mass spectra. *BMC Bioinformatics*, 11:148, 2010.

[237] W. E. Wolski, M. Lalowski, P. Jungblut and K. Reinert. Calibration of mass spectrometric peptide mass fingerprint data without specific external or internal calibrants. *BMC Bioinformatics*, 6:203, 2005.

[238] J. W. Wong, G. Cagney and H. M. Cartwright. SpecAlign—processing and alignment of mass spectra datasets. *Bioinformatics*, 21(9):2088–2090, 2005.

[239] L.-C. Wu, H.-H. Chen, J.-T. Horng, C. Lin, N. E. Huang, Y.-C. Cheng and K.-F. Cheng. A novel preprocessing method using Hilbert Huang transform for MALDI-TOF and SELDI-TOF mass spectrometry data. *PLoS One*, 5(8):e12493, 2010.

[240] Y. Wu, Y. Mechref, I. Klouckova, M. V. Novotny and H. Tang. A computational approach for the identification of site-specific protein glycosylations through ion-trap mass spectrometry. In *Proc. of RECOMB 2006 satellite workshop on Systems biology and computational proteomics*, volume 4532 of *Lect. Notes Comput. Sc.*, pages 96–107. Springer, 2007.

[241] C. Xu and B. Ma. Complexity and scoring function of MS/MS peptide de novo sequencing. In *Proc. of Computational Systems Bioinformatics Conference (CSB 2006)*, volume 4 of *Series on Advances in Bioinformatics and Computational Biology*, pages 361–369. Imperial College Press, 2006.

[242] J. Yates, P. Griffin, L. Hood and J. Zhou. Computer aided interpretation of low energy MS/MS mass spectra of peptides. In J. Villafranca, editor, *Techniques in Protein Chemistry II*, pages 477–485. Academic Press, San Diego, 1991.

[243] J. A. Yergey. A general approach to calculating isotopic distributions for mass spectrometry. *Int. J. Mass Spectrom. Ion Phys.*, 52(2–3):337–349, 1983.

[244] J. Zaia. Mass spectrometry of oligosaccharides. *Mass Spectrom. Rev.*, 23(3):161–227, 2004.

[245] J. Zhang, E. Gonzalez, T. Hestilow, W. Haskins and Y. Huang. Review of peak detection algorithms in liquid-chromatography-mass spectrometry. *Curr. Genomics*, 10(6):388–401, 2009.

[246] J. Zhang, D. Xu, W. Gao, G. Lin and S. He. Isotope pattern vector based tandem mass spectral data calibration for improved peptide and protein identification. *Rapid Commun. Mass Spectrom.*, 23(21):3448–3456, 2009.

[247] N. Zhang, R. Aebersold and B. Schwikowski. ProbID: a probabilistic algorithm to identify peptides through sequence database searching using tandem mass spectral data. *Proteomics*, 2(10):1406–1412, 2002.

[248] W. Zhang and B. T. Chait. ProFound: an expert system for protein identification using mass spectrometric peptide mapping information. *Anal. Chem.*, 72(11):2482–2489, 2000.

[249] R. Zubarev and M. Mann. On the proper use of mass accuracy in proteomics. *Mol. Cell. Proteomics.*, 6(3):377–381, 2007.