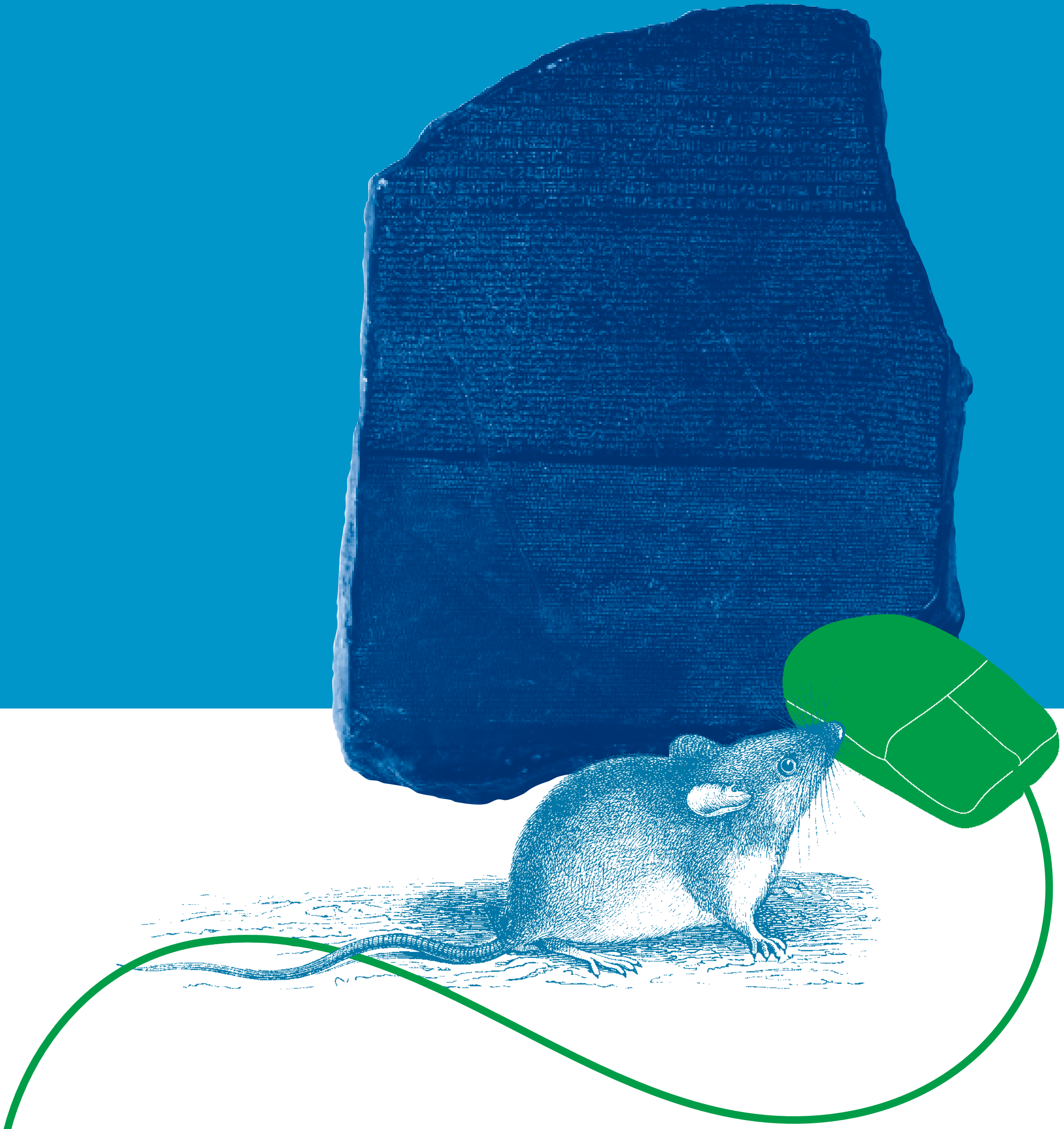
AN INTRODUCTION TO BIOINFORMATICS ALGORITHMS

NEIL C. JONES AND PAVEL A. PEVZNER



8 *Graph Algorithms*

Many bioinformatics algorithms may be formulated in the language of *graph theory*. The use of the word "graph" here is different than in many physical science contexts: we do not mean a chart of data in a Cartesian coordinate system. In order to work with graphs, we will need to define a few concepts that may not appear at first to be particularly well motivated by biological examples, but after introducing some of the mathematical theory we will show how powerful they can be in such bioinformatics applications as DNA sequencing and protein identification.

8.1 Graphs

Figure 8.1 (a) shows two white and two black knights on a 3×3 chessboard. Can they move, using the usual chess knight's moves,[1] to occupy the positions shown in figure 8.1 (b)? Needless to say, two knights cannot occupy the same square while they are moving.

Figure 8.2b represents the chessboard as a set of nine points. Two points are connected by a line if moving from one point to another is a valid knight move. Figure 8.2c shows an equivalent representation of the resulting diagram that reveals that knights move around a "cycle" formed by points 1, 6, 7, 2, 9, 4, 3, and 8. Every knight's move on the chessboard corresponds to moving to a neighboring point in the diagram, in either a clockwise or counterclockwise direction. Therefore the white-white-black-black knight arrangement cannot be transformed into the alternating white-black-white-black arrangement.

1. In the game of chess, knights (the "horses") can move two steps in any of four directions (left, right, up, and down) followed by one step in a perpendicular direction, as shown in figure 8.1 (c).

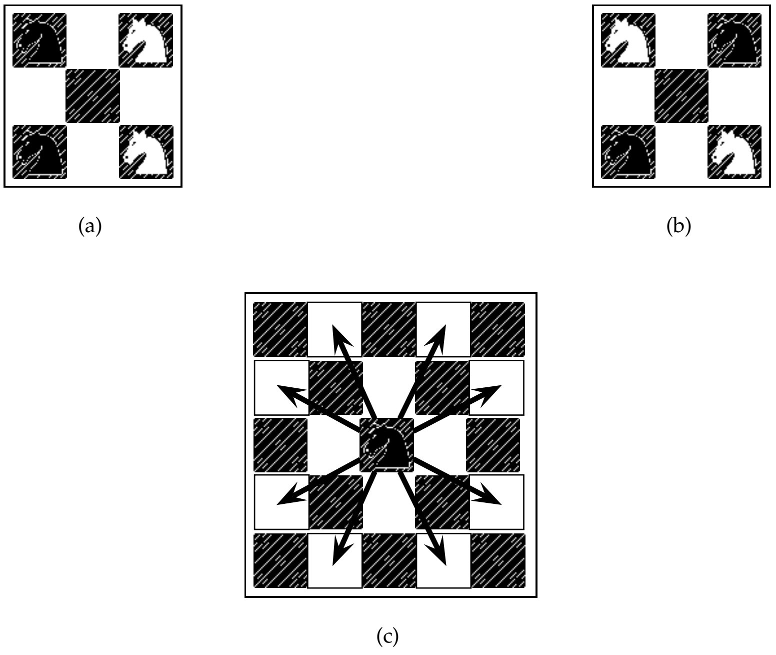


Figure 8.1 Two configurations of four knights on a chessboard. Can you use valid knight moves to turn the configuration in (a) into the configuration in (b)? Valid knight moves are shown in (c).

Diagrams with collections of points connected by lines are examples of *graphs*. The points are called *vertices* and lines are called *edges*. A simple graph shown in figure 8.3, consists of five vertices and six edges. We denote a graph by $G = G(V, E)$ and describe it by its set of vertices V and set of edges E (every edge can be written as a pair of vertices). The graph in figure 8.3 is described by the vertex set $V = \{a, b, c, d, e\}$ and the edge set $E = \{(a, b), (a, c), (b, c), (b, d), (c, d), (c, e)\}$.

The way the graph is actually drawn is irrelevant; two graphs with the same vertex and edge sets are equivalent, even if the particular pictures that represent the graph appear different (see figure 8.3). The only important feature of a graph is which vertices are connected and which are not.

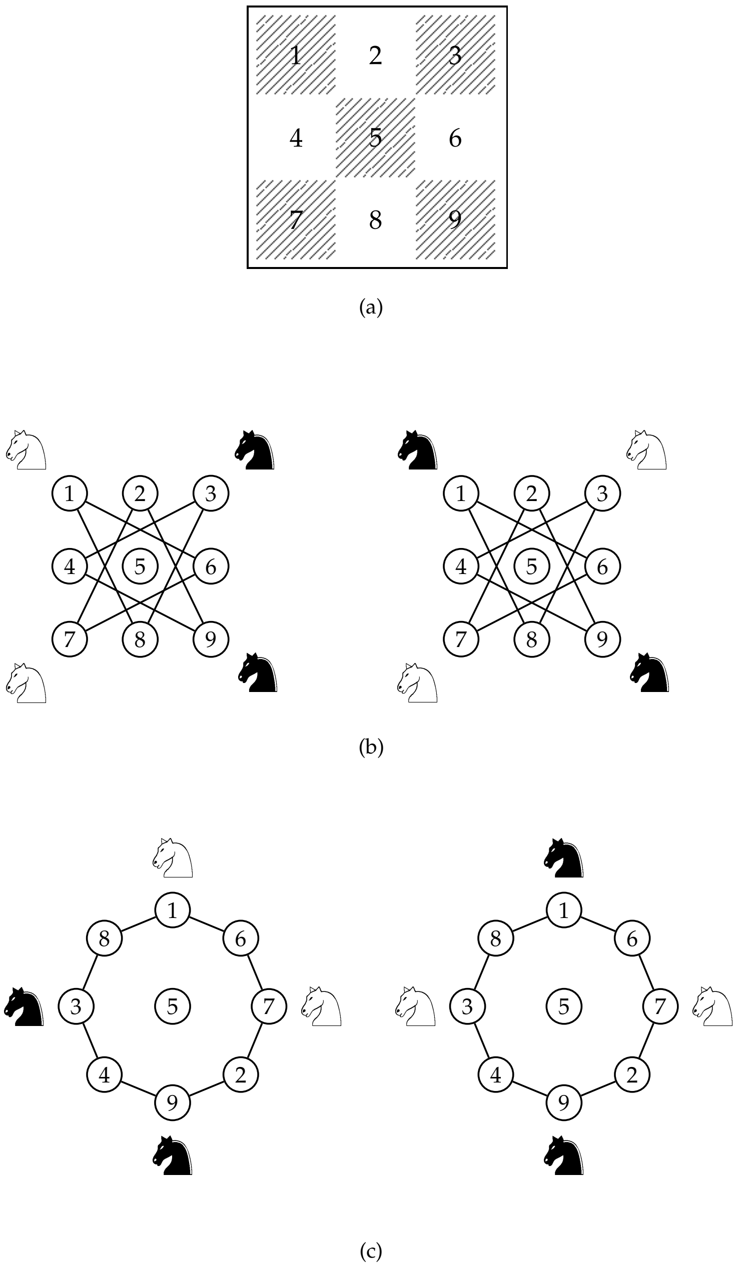


Figure 8.2 A graph representation of a chessboard. A knight sitting on some square can reach any of the squares attached to that square by an edge.

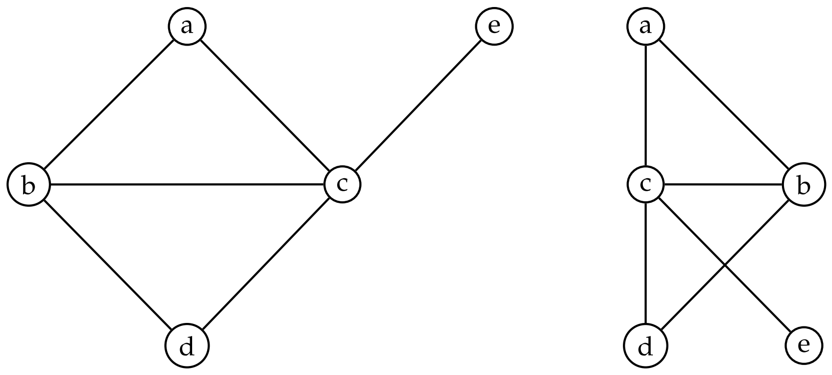
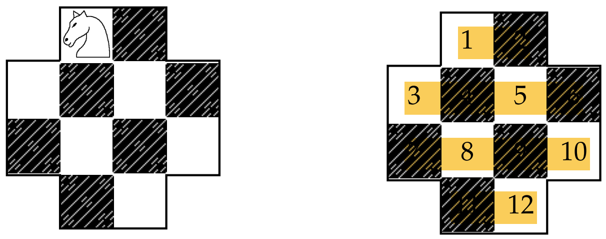


Figure 8.3 Two equivalent representations of a simple graph with five vertices and six edges.

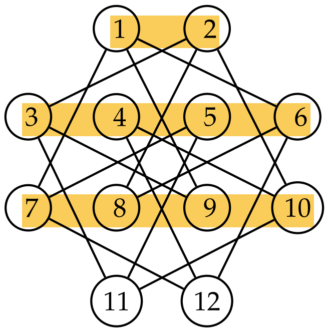
Figure 8.4 represents another chessboard obtained from a 4×4 chessboard by removing the four corner squares. Can a knight travel around this board, pass through each square exactly once, and return to the same square it started on? Figure 8.4 (b) shows a rather complex graph with twelve vertices and sixteen edges revealing all possible knight moves. However, rearranging the vertices (fig. 8.4c) reveals the cycle that describes the correct sequence of moves.

The number of edges incident to a given vertex v is called the *degree* of the vertex and is denoted $d(v)$. For example, vertex 2 in figure 8.4 (c) has degree 3 while vertex 4 has degree 2. The sum of degrees of all 12 vertices is, in this case, 32 (8 vertices of degree 3 and 4 vertices of degree 2), twice the number of edges in the graph. This is not a coincidence: for every graph G with vertex set V and edge set E, $\sum_{v \in V} d(v) = 2 \cdot |E|$. Indeed, an edge connecting vertices v and w is counted in the sum $\sum_{v \in V} d(v)$ twice: first in the term $d(v)$ and again in the term $d(w)$. The equality $\sum_{v \in V} d(v) = 2 \cdot |E|$ explains why you cannot connect fifteen phones such that each is connected to exactly seven others, and why a country with exactly three roads out of every city cannot have precisely 1000 roads.

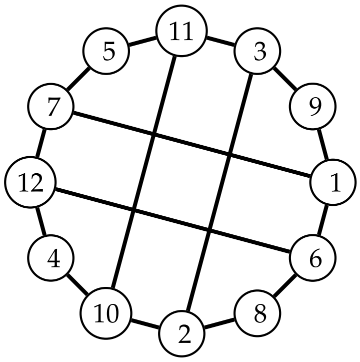
Many bioinformatics problems make use of *directed* graphs, in which every edge is directed from one vertex to another, as shown by the arrows in figure 8.5. Every vertex v in a directed graph is characterized by $indegree(v)$ (the number of incoming edges) and $outdegree(v)$ (the number of outgoing



(a)



(b)



(c)

Figure 8.4 The knight's tour through the twelve squares in part (a) can be seen by constructing a graph (b) and rearranging its vertices in a clever way (c).

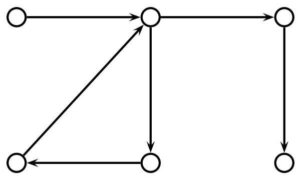


Figure 8.5 A directed graph.

edges). For every directed graph $G(V, E)$,

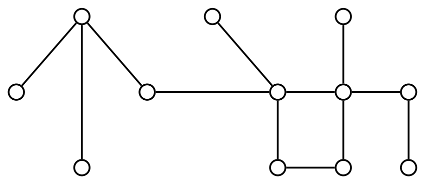
$$\sum_{v \in V} indegree(v) = \sum_{v \in V} outdegree(v),$$

since every edge is counted once on the right-hand side of the equation and once on the left-hand side.

A graph is called *connected* if all pairs of vertices can be connected by a *path,* which is a continuous sequence of edges, where each successive edge begins where the previous one left off. Paths that start and end at the same vertex are referred to as *cycles*. For example, the paths (3-2-10-11-3) and (3-2-8-6-12-7-5-11-3) in figure 8.4 (c) are cycles.

Graphs that are not connected are *disconnected* (fig. 8.6). Disconnected graphs can be partitioned into *connected components.* One can think of a graph as a map showing cities (vertices) and the freeways (edges) that connect them. Not all cities are connected by freeways: for example, you cannot drive from Miami to Honolulu. These two cities belong to two different connected components of the graph. A graph is called *complete* if there is an edge between every two vertices.

Graph theory was born in the eighteenth century when Leonhard Euler solved the famous Königsberg Bridge problem. Königsberg is located on the banks of the Pregel River, with a small island in the middle. The various parts of the city are connected by bridges (fig. 8.7) and Euler was interested in whether he could arrange a tour of the city in such a way that the tour visits each bridge exactly once. For Königsberg this turned out to be impossible, but Euler basically invented an algorithm to solve this problem for any city.



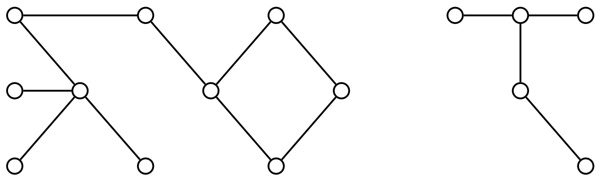


Figure 8.6 A connected (a) and a disconnected (b) graph.

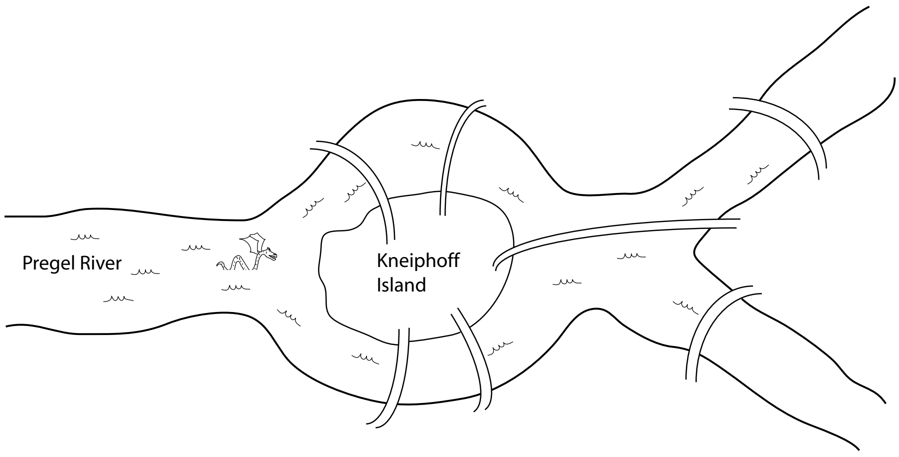


Figure 8.7 Bridges of Königsberg.

Bridge Obsession Problem:
Find a tour through a city (located on n islands connected by m bridges) that starts on one of the islands, visits every bridge exactly once, and returns to the originating island.

Input: A map of the city with n islands and m bridges.

Output: A tour through the city that visits every bridge exactly once and returns to the starting island.

Figure 8.8 shows a city map with ten islands and sixteen bridges, as well as the transformation of the map into a graph with ten vertices and sixteen edges (every island corresponds to a vertex and every bridge corresponds to an edge). After this transformation, the Bridge Obsession problem turns into the Eulerian Cycle problem that was solved by Euler and later found thousands of applications in different areas of science and engineering:

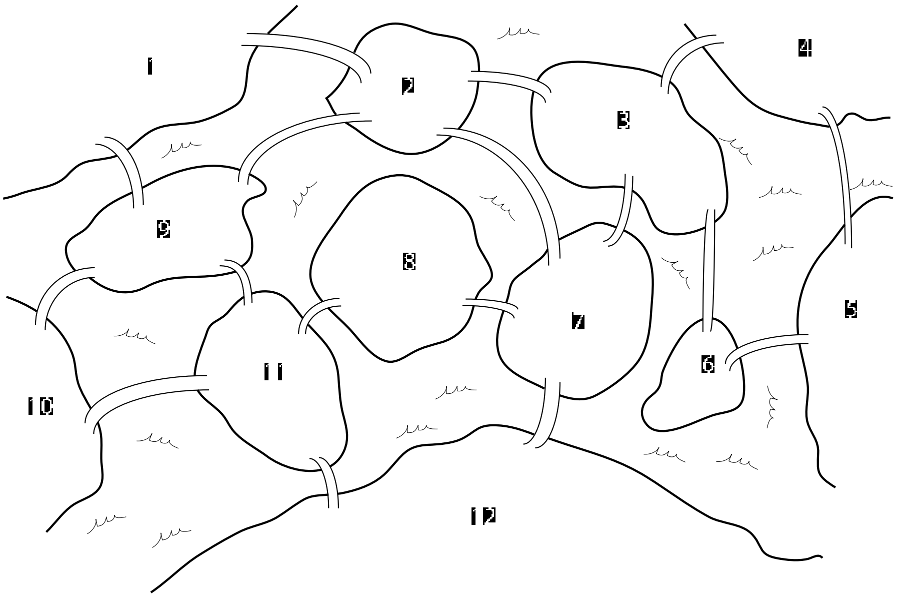
Eulerian Cycle Problem:
Find a cycle in a graph that visits every edge exactly once.

Input: A graph G.

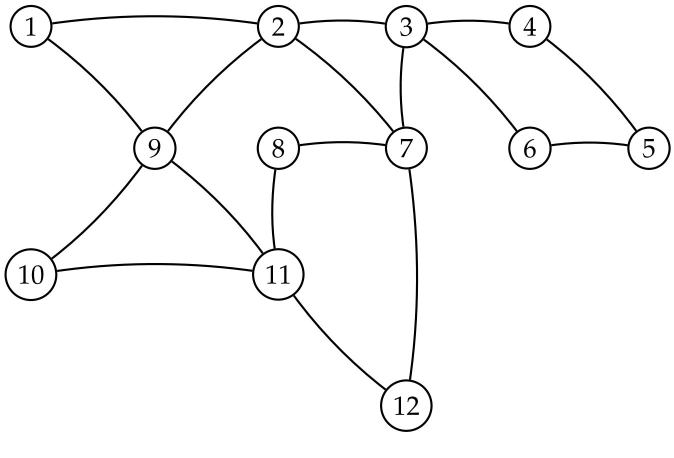
Output: A cycle in G that visits every edge exactly once.

After the Königsberg Bridge problem was solved, graph theory was forgotten for a century before it was rediscovered by Arthur Cayley who studied the chemical structures of (noncyclic) saturated hydrocarbons C_nH_{2n+2} (fig. 8.9). Structures of this type of hydrocarbon are examples of *trees*, which are simply connected graphs with no cycles. It is not hard to show that every tree has at least one vertex with degree 1, or *leaf*.[2] This observation immediately implies that every tree on n vertices has $n-1$ edges, regardless of the structure of the tree. Indeed, since every tree has a leaf, we can remove it and its attached edge, resulting in another tree. So far we have removed one edge and one vertex. In this smaller tree there exists a leaf that we, again, remove. So far, we have removed two vertices and two edges. We keep this up until we are left with a graph with a single vertex and no edges. Since we have removed $n-1$ vertices and $n-1$ edges, the number of edges in every tree is $n-1$ (fig. 8.10).

2. Actually, every tree has at least two leaves, except for the trivial single-vertex tree.



(a)



(b)

Figure 8.8 A more complicated version of Königsberg (a). To solve the Bridge Obsession problem, Euler transformed the map of Königsberg into a graph (b) and found an *Eulerian cycle*. The path that runs through vertices 1-2-3-4-5-6-3-7-2-9-11-8-7-12-11-10-9-1 is an Eulerian cycle.

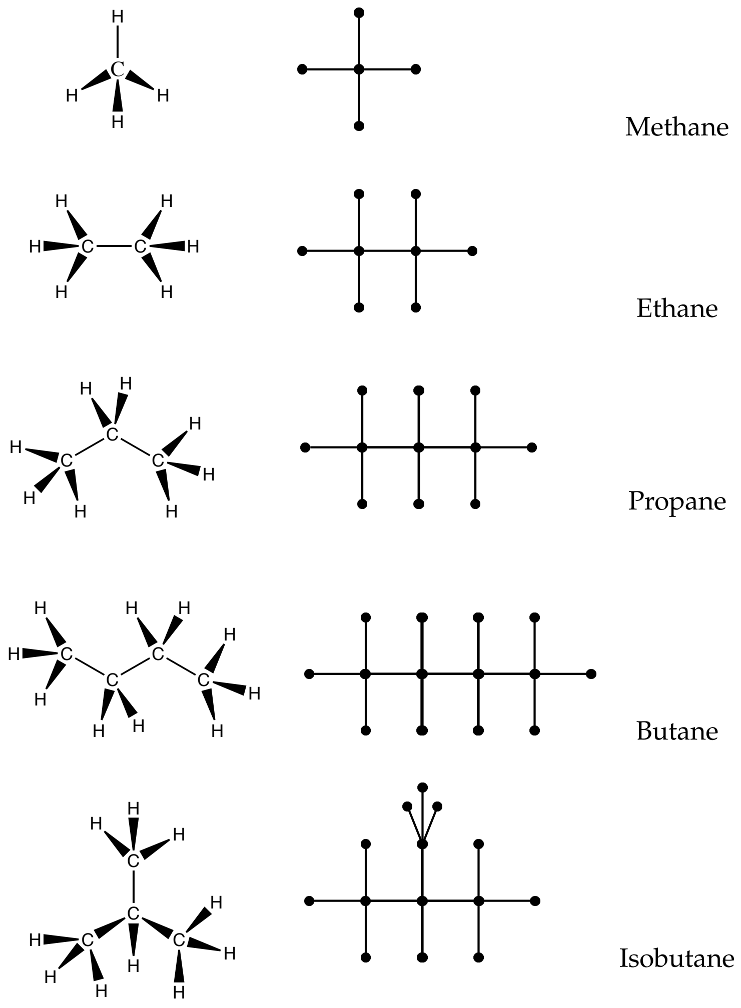


Figure 8.9 Hydrocarbons (the saturated, nonaromatic variety) as chemists see them (left), and their graph representation (right). Two different molecules with the same number of the same types of atoms are called *structural isomers*.

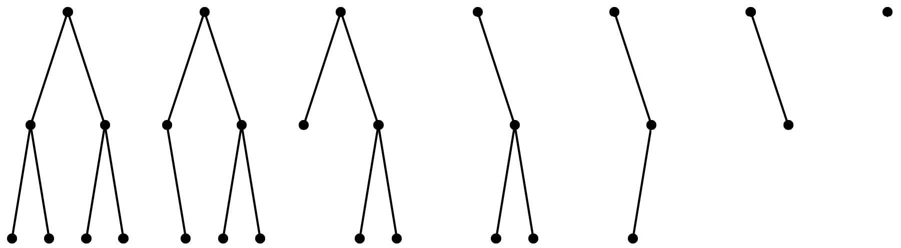


Figure 8.10 Proving that a tree with n vertices has $n - 1$ edges.

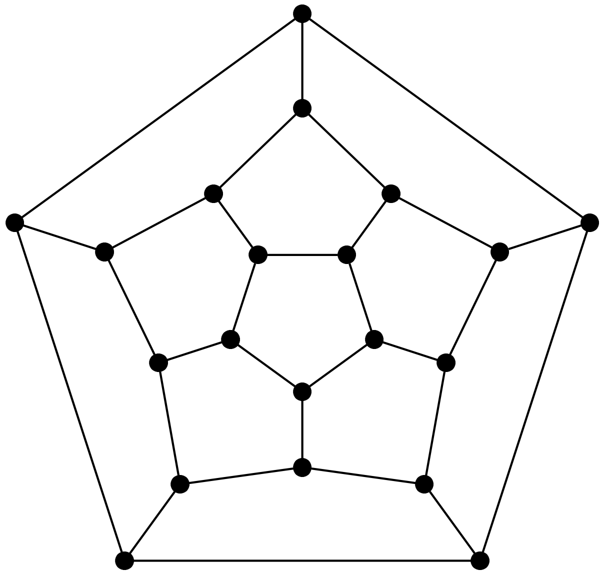


Figure 8.11 Can you travel from any one of the vertices in this graph, visit every other vertex exactly once, and end up at the original vertex?

Shortly after Cayley's work on tree enumeration, Sir William Hamilton invented a game corresponding to a graph whose twenty vertices were labeled with the names of twenty famous cities (fig. 8.11). The goal is to visit all twenty cities in such a way that every city is visited exactly once before returning back to the city where the tour started. As the story goes, Hamilton sold the game for 25 pounds to a London game dealer and it failed miserably. Despite the commercial failure of a great idea, the more general problem of finding "Hamiltonian cycles" in arbitrary graphs is of critical importance to many scientific and engineering disciplines. The problem of finding Hamiltonian cycles looks deceivingly simple and somehow similar to the Eulerian Cycle problem. However, it turns out to be $\mathcal{NP}$-complete while the Eulerian Cycle problem can be solved in linear time.[3]

Hamiltonian Cycle Problem:
Find a cycle in a graph that visits every vertex exactly once.

Input: A graph G.

Output: A cycle in G that visits every vertex exactly once (if such a cycle exists).

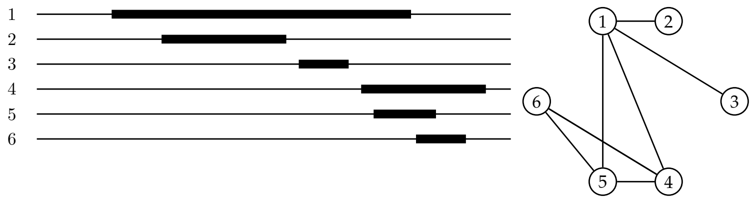
Graphs, like many freeway maps, often give some sort of weight to every edge, as in the Manhattan Tourist problem in chapter 6. The weight of an edge may reflect, depending on the context, different attributes. For example, the length of a freeway segment connecting two cities, the number of tourist attractions along a city block, and the alignment score between two amino acids are all natural weighting schemes. Weighted graphs are often formally represented as an ordered triple, $G = (V, E, w)$, where V is the set of vertices in the graph, E is the set of edges, and w is a weight function defined for every edge e in E (i.e., $w(e)$ is a number reflecting the weight of edge e). Given a weighted graph, one may be interested in finding some shortest path between two vertices (e.g., a shortest path between San Diego and New York). Though this problem may sound difficult if you were given a complicated road map, it turns out that there exist fast algorithms to answer this question.

Shortest Path Problem:
Given a weighted graph and two vertices, find the shortest distance between them.

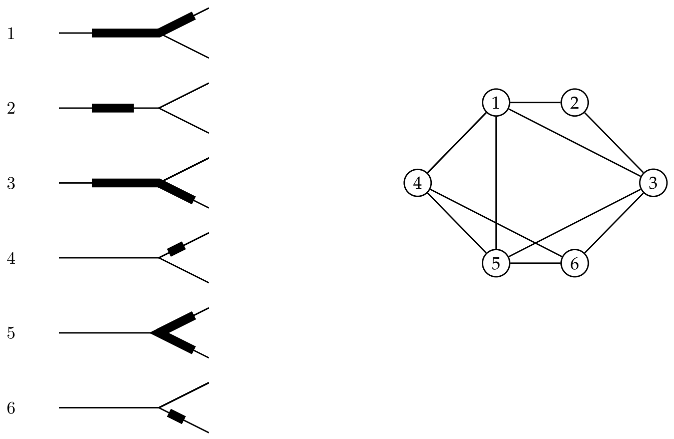
Input: A weighted graph, $G = (V, E, w)$, and two distinguished vertices s and t.

Output: The shortest path between s and t in graph G.

3. The Hamiltonian Cycle problem is equivalent in complexity to the Traveling Salesman problem mentioned in chapter 2 and is therefore $\mathcal{NP}$-complete.



(a)



(b)

Figure 8.12 Two different hypothetical structures of a gene. (a) Linear structures exhibit very different interval graphs than the graphs exhibited by (b) branched structures. It is impossible to construct a linear sequence of overlapping intervals that gives rise to the graph in (b).

We emphasize that this problem is different—and somewhat more complicated—than the Longest Path in a Directed Acyclic Graph (DAG) problem that we considered in chapter 6.

Graphs are powerful tools in many applied studies and their role goes well beyond analysis of maps of freeways. For example, at first glance, the following "basketball" problem has nothing to do with graph theory:

> Fifteen teams play a basketball tournament in which each team plays with each other team. Prove that the teams can always be numbered from 1 to 15 in such a way that team 1 defeated team 2, team 2 defeated team 3, ... , team 14 defeated team 15.

A careful analysis reduces this problem to finding a Hamiltonian path in a directed graph on fifteen vertices.

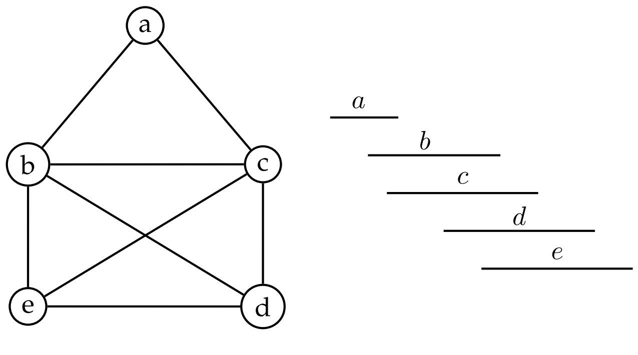
8.2 Graphs and Genetics

Conceived by Euler, Cayley, and Hamilton, graph theory flourished in the twentieth century to become a critical component of discrete mathematics. In the 1950s, Seymour Benzer applied graph theory to show that genes are linear.

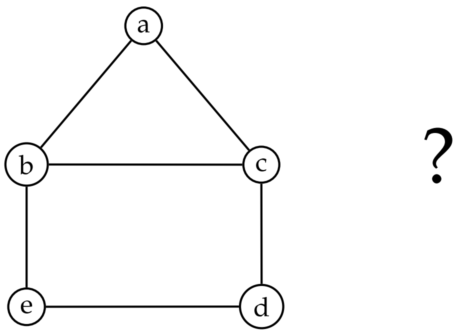
At that time, it was known that genes behaved as functional units of DNA, much like pearls on a necklace, but the chemical structure and organization of the genes was not clear. Were genes broken into still smaller components? If so, how were they organized? Prior to Watson and Crick's elucidation of the DNA double helix, it seemed a reasonable hypothesis that the DNA content of genes was branched, as in figure 8.12, or even looped, rather than linear. These two organizations have very different topological implications, which Benzer exploited in an ingenious experiment.

Benzer studied a large number of bacteriophage[4] T4 mutants, which happened to have a continuous interval deleted from an important gene. In their "normal" state, nonmutant T4 phages will kill a bacterium. The mutant T4 phages that were missing a segment of their genome could not kill the bacterium. Different mutants had different intervals deleted from the gene, and Benzer had to determine which interval had been deleted in each mutant. Though Benzer did not know exactly where in the gene the mutant's deletion was, he had a way to test whether two deletions (i.e., their intervals) overlapped, relying on how two phages with two different deletions behave

4. A bacteriophage is a virus that attacks bacteria.



(a)



(b)

Figure 8.13 An interval graph (a) and a graph that is not an interval graph (b).

inside a bacterial host cell. When two mutant phages with two different deletions infect a bacterium, two outcomes are possible depending on whether the deleted intervals overlap or not. If they do not overlap, then two phages combined have all the genetic material of one normal phage. However, if two deleted intervals overlap, some genetic material is absent in both phages. The Benzer experiment was based on the observation that in the former case (all genetic material of a normal phage is present), two mutants are able to kill the bacterium; and in the latter case (where some genetic material is removed in both phages) the bacterium survives.

Benzer infected bacteria with each *pair* of mutant strains from his T4 phage and simply noted which pairs killed the bacterial host. Pairs which were lethal to the host were mutants whose deleted intervals did not overlap. Benzer constructed a graph[5] where each strain was a vertex and two vertices were connected when a double infection of a bacterial host was nonlethal. He reasoned that if genes were linear [fig. 8.12 (a)], he would probably see one type of graph, but if the genes were branched [fig. 8.12 (b)] he would see another.

Given a set of n intervals on a line segment, the *interval graph* is defined as a graph with n vertices that correspond to the intervals. There is an edge between vertices v and w if and only if the intervals v and w overlap. Interval graphs have several important properties that make them easy to recognize— for example, the graph in figure 8.13 (a) is an interval graph, whereas the "house" graph in figure 8.13 (b) is not. Benzer's problem was equivalent to deciding whether the graph obtained from his bacteriophage experiment represented an interval graph. Had the experiment resulted in a graph like the "house" graph, then the genes could not have been organized as linear structures. As it turned out, the graph was indeed an interval graph, indicating that genes were composed of linearly organized functional units.

8.3 DNA Sequencing

Imagine several copies of a magazine cut into millions of pieces. Each copy is cut in a different way, so a piece from one copy may overlap pieces from another. Assuming that some large number of pieces are just sort of lost, and the remaining pieces are splashed with ink, can you recover the original text? This, essentially, is the problem of fragment assembly in DNA sequencing. Classic DNA sequencing technology allows one to read short 500- to 700-nucleotide sequences per experiment, each fragment corresponding to one of the many magazine pieces. Assembling the entire genome from these short fragments is like reassembling the magazine from the millions of tiny slips of paper.[6] Both problems are complicated by unavoidable experimental errors—ink splashes on the magazine, and mistakes in reading nucleotides. Furthermore, the data are frequently incomplete—some magazine pieces get lost, while some DNA fragments never make it into the sequencing machine. Nevertheless, efforts to determine the DNA sequence of organisms have been remarkably successful, even in the face of these difficulties.

5. It is not clear that he actually knew anything about graph theory at the time, but graph theorists eventually noticed his work.
6. We emphasize that biologists reading these 500- to 700-nucleotide sequences have no idea where they are located within the entire DNA string.

Two DNA sequencing methods were invented independently and simultaneously in Cambridge, England by Fred Sanger and in Cambridge, Massachusetts by Walter Gilbert. Sanger's method takes advantage of how cells make copies of DNA. Cells copy a strand of DNA nucleotide by nucleotide in a reaction that adds one base at a time. Sanger realized that he could make copies of DNA fragments of different lengths if he starved the reaction of one of the four bases: a cell can only copy its DNA while it has all of the bases in supply. For a sequence ACGTAAGCTA, starving at T would produce a mixture of the fragments ACG and ACGTAAGC. By running one starvation experiment for each of A, T, G, and C and then separating the resulting DNA fragments by length, one can read the DNA sequence.[7] Each of four starvation experiments produces a *ladder* of fragments of varying lengths called the *Sanger ladder*.[8] This approach culminated in the sequencing of a 5386-nucleotide virus in 1977 and a Nobel Prize shortly thereafter. Since then the amount of DNA sequence data has been increasing exponentially, particularly after the launch of the Human Genome Project in 1989. By 2001, it had produced the roughly 3 billion-nucleotide sequence of the human genome.

Within the past twenty years, DNA sequencing technology has been developed to the point where modern sequencing machines can sequence 500- to 700-nucleotide DNA fragments, called *sequencing reads*. These reads then have to be assembled into a continuous genome, which turns out to be a very hard problem. Even though the DNA reading process has become quite automated, these machines are not microscope-like devices that simply scan 500 nucleotides as if they were a sentence in a book. The DNA sequencing machines measure the lengths of DNA fragments in the Sanger ladder, but even this task is difficult; we cannot measure a *single* DNA fragment, but must measure billions of identical fragments.

Shotgun sequencing starts with a large sample of genomic DNA. The sample is *sonicated*, a process which randomly partitions each piece of DNA in the sample into *inserts*; the inserts that are smaller than 500 nucleotides are removed from further consideration. Before the inserts can be read, each one must be multiplied billions of times so that it is possible to read the ladders produced by Sanger's technique. To amplify the inserts, a sample is cloned into a *vector*, and this vector used to infect a bacterial host. As the bacterium reproduces, it creates a colony that contains billions of copies of the vector

7. Later Sanger found chemicals—so-called dideoxynucleotides—that could be inserted in place of A, T, G, or C, and cause a growing DNA chain to end.
8. The Sanger ladder for T shows the lengths of all sub-fragments ending at T and therefore reveals the set of positions where T occurs.

and its associated insert. As a result, the cloning process results in the production of a large sample of one particular insert that can then be sequenced by the Sanger method. Usually, only the first 500 to 700 nucleotides of the insert can be interpreted from this experiment. DNA sequencing is therefore a two stage process including both experimental (reading 500–700 nucleotide sequences form different inserts) and computational (assembling these reads into a single long sequence) components.

8.4 Shortest Superstring Problem

Since every string, or *read*, that we sequence came from the much longer genomic string, we are interested in a *superstring* of the reads—that is, we want a long string that "explains" all the reads we generated. However, there are many possible superstrings to choose from—for example, we could concatenate all the reads together to get a (not very helpful) superstring. We choose to be most interested in the shortest one, which turns out to be a reasonable first approximation to the unknown genomic DNA sequence. With this in mind, the simplest approximation of DNA sequencing corresponds to the following problem.

Shortest Superstring Problem:
Given a set of strings, find a shortest string that contains all of them.

Input: Strings $s_1, s_2, \ldots, s_n$.

Output: A string s that contains all strings $s_1, s_2, \ldots, s_n$ as substrings, such that the length of s is as small as possible.

Figure 8.14 presents two superstrings for the set of all eight three-letter strings in a 0–1 alphabet. The first (trivial) superstring is obtained by the concatenation of all eight strings, while the second one is a shortest superstring.

Define $overlap(s_i, s_j)$ to be the length of the longest prefix of s_j that matches a suffix of s_i. The Shortest Superstring problem can be cast as a Traveling Salesman problem in a complete directed graph with n vertices corresponding to strings $s_1, \ldots, s_n$ and edges of length $-overlap(s_i, s_j)$ (fig. 8.15). This reduction, of course, does not lead to an efficient algorithm since the TSP is $\mathcal{NP}$-complete. Moreover it is known that the Shortest Superstring problem is itself $\mathcal{NP}$-complete, so that a polynomial algorithm for this problem is un-

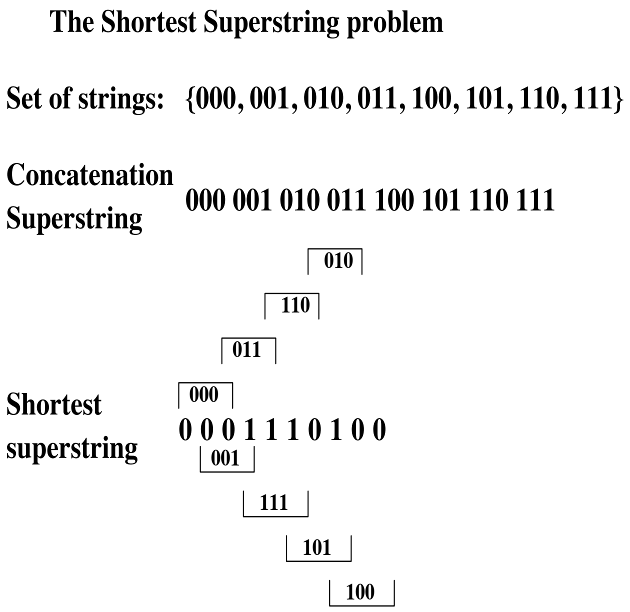

The Shortest Superstring problem
Set of strings: {000, 001, 010, 011, 100, 101, 110, 111}
Concatenation Superstring
000 001 010 011 100 101 110 111
010
110
011
000
Shortest superstring
0 0 0 1 1 1 0 1 0 0
001
111
101
100

Figure 8.14 Superstrings for the set of eight three-letter strings in a 0–1 alphabet. Concatenating all eight strings results in a 24-letter superstring, while the shortest superstring contains only 10 letters. The shortest superstring in this case represents a solution of the Clever Thief problem—it is the minimum string of tests a thief has to conduct to try all possible k-letter passwords for a combination lock.

likely. The early DNA sequencing algorithms used a simple greedy strategy: repeatedly merge a pair of strings with maximum overlap until only one string remains. It has been conjectured, but not yet proved, that this greedy algorithm has performance guarantee 2.

8.5 DNA Arrays as an Alternative Sequencing Technique

When the Human Genome Project started, DNA sequencing was a routine but time-consuming and hard-to-automate procedure. In 1988 four groups of biologists independently and simultaneously suggested a different sequencing technique called *Sequencing by Hybridization*. SBH involves building a miniature *DNA array*, also known as a *DNA chip*, that contains thousands of short DNA fragments called *probes*. Each of these short fragments reveals

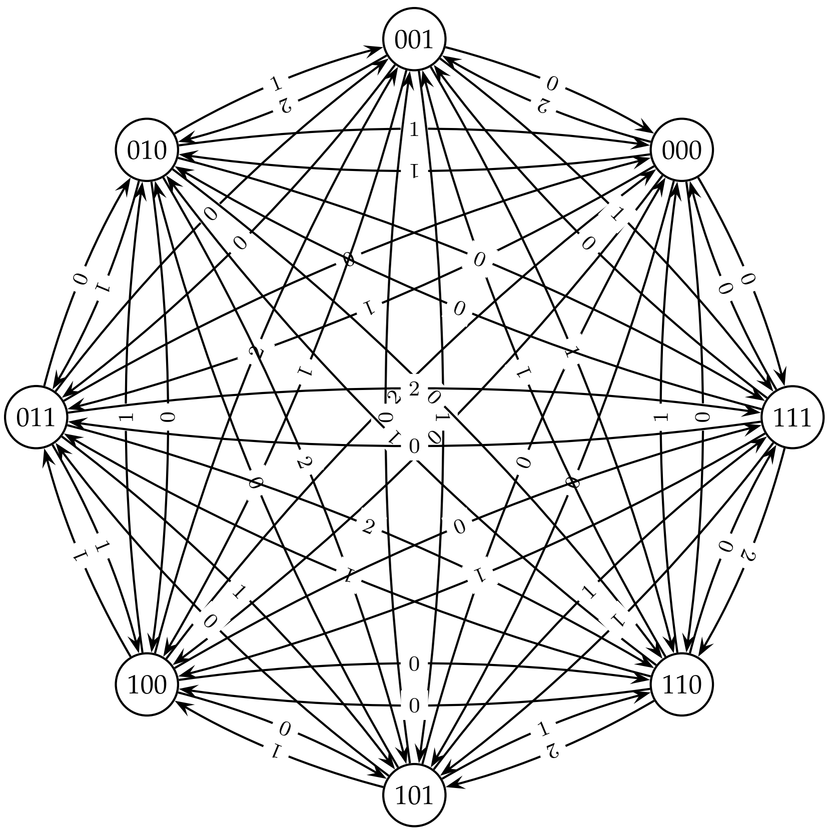


Figure 8.15 The overlap graph for the eight strings in figure 8.14.

whether or not a known—but short—sequence occurs in the unknown DNA sequence; all these pieces of information together should reveal the identity of the target DNA sequence.

Given a short probe (an 8- to 30-nucleotide single-stranded synthetic DNA fragment) and a single-stranded target DNA fragment, the target will *hybridize* with the probe if the probe is a substring of the target's Watson-Crick *complement*. When the probe and the target are mixed together, they form a weak chemical bond and stick together. For example, a probe ACCGTGGA will hybridize to a target CCCTGGCACCTA since it is complementary to the substring TGGCACCT of the target.

In 1988 almost nobody believed that the idea of using DNA probes to sequence long genomes would work, because both the biochemical problem of synthesizing thousands of short DNA fragments and the combinatorial problem of sequence reconstruction appeared too complicated. Shortly after the first paper describing DNA arrays was published, the journal *Science* wrote that given the amount of work required to synthesize a DNA array, "[using DNA arrays for sequencing] would simply be substituting one horrendous task for another." A major breakthrough in DNA array technology was made by Steve Fodor and colleagues in 1991. Their approach to array manufacturing relies on *light-directed polymer synthesis*,[9] which has many similarities to computer chip manufacturing (fig. 8.16). Using this technique, building an array with all 4^l probes of length l requires just $4 \cdot l$ separate reactions, rather than the presumed 4^l reactions. With this method, a California-based biotechnology company, Affymetrix, built the first 64-kb DNA array in 1994. Today, building 1-Mb or larger arrays is routine, and the use of DNA arrays has become one of the most widespread new biotechnologies.

SBH relies on the hybridization of the target DNA fragment against a very large array of short probes. In this manner, probes can be used to test the unknown target DNA to determine its *l-mer composition*.[10] The *universal* DNA array contains all 4^l probes of length l and is applied as follows (fig. 8.17):

- Attach all possible probes of length l (l=8 in the first SBH papers) to a flat surface, each probe at a distinct and known location. This set of probes is called the *DNA array*.

- Apply a solution containing fluorescently labeled DNA fragment to the array.

- The DNA fragment hybridizes with those probes that are complementary to substrings of length l of the fragment.

- Using a spectroscopic detector, determine which probes hybridize to the DNA fragment to obtain the l-mer composition of the target DNA fragment.

- Apply the combinatorial algorithm described below to reconstruct the sequence of the target DNA fragment from the l-mer composition.

9. Light as in photons, not light as in "not heavy."
10. The l-mer composition of a string is simply the set of all l-mers present in the string. For example, the 8-mer composition of CCCTGGCACCTA is {CCCTGGCA, CCTGGCAC, CTGGCACC, TGGCACCT, GGCACCTA}.

Figure 8.16 A GeneChip, produced by Affymetrix. (Picture courtesy of Affymetrix, Inc.)

8.6 Sequencing by Hybridization

Given an unknown DNA sequence, an array provides information about all strings of length l that the sequence contains, but does not provide information about their positions in the sequence. For a string s of length n, the l-mer composition, or *spectrum*, of s, is the multiset of $n - l + 1$ l-mers in s and is written $Spectrum(s, l)$. If $l = 3$ and $s =$ TATGGTGC, then $Spectrum(s, l) = \{$TAT, ATG, TGG, GGT, GTG, TGC$\}$.[11] We can now formulate the problem of sequencing a target DNA fragment from its DNA array data.

11. The l-mers in this spectrum are listed in the order of their appearance in s creating the impression that we know which nucleotide occur at each position. We emphasize that the order of these l-mers in s is unknown and it is probably more appropriate to list them in lexicographic order, like ATG, GGT, GTG, TAT, TGC, TGG.

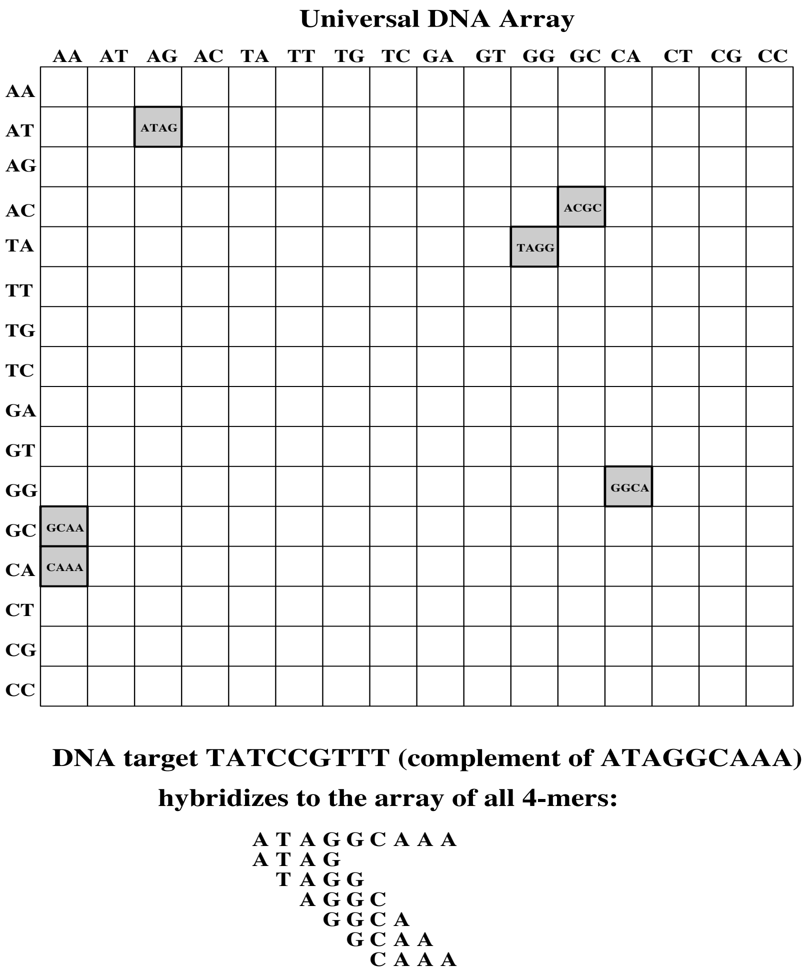

Universal DNA Array
AA AT AG AC TA TT TG TC GA GT GG GC CA CT CG CC
AA AT AG AC TA TT TG TC GA GT GG GC CA CT CG CC
ATAG
ACGC
TAGG
GGCA
GCAA
CAAA
DNA target TATCCGTTT (complement of ATAGGCAAA)
hybridizes to the array of all 4-mers:
A T A G G C A A A
A T A G
T A G G
A G G C
G G C A
G C A A
C A A A

Figure 8.17 Hybridization of TATCCGTTT with the universal DNA array consisting of all 4^4 4-mers.

Sequencing by Hybridization (SBH) Problem:
Reconstruct a string from its l-mer composition.

> **Input:** A set, $\mathcal{S}$, representing all l-mers from an (unknown) string s.
>
> **Output:** String s such that $Spectrum(s, l) = \mathcal{S}$.

Although conventional DNA sequencing and SBH are very different experimental approaches, you can see that the corresponding computational problems are quite similar. In fact, SBH is a particular case of the Shortest Superstring problem when the strings $s_1, \ldots, s_n$ represent the set of all substrings of s of fixed size. However, in contrast to the Shortest Superstring problem, there exists a simple linear-time algorithm for the SBH problem. Notice that it is not a contradiction that the Shortest Superstring problem is $\mathcal{NP}$-complete, yet we claim to have a linear-time algorithm for the SBH problem, since the Shortest Superstring problem is more general than the SBH problem.

Although DNA arrays were originally proposed as an alternative to conventional DNA sequencing, de novo sequencing with DNA arrays remains an unsolved problem in practice. The primary obstacle to applying DNA arrays for sequencing is the inaccuracy in interpreting hybridization data to distinguish between perfect matches [i.e., l-mers present in $Spectrum(s, l)$] and highly stable mismatches (i.e, l-mers not present in $Spectrum(s, l)$, but with sufficient chemical bonding potential to generate a strong hybridization signal). This is a particularly difficult problem for the short probes used in universal arrays. As a result, DNA arrays have become more popular in gene expression analysis[12] and studies of genetic variations—both of which are done with longer probes—than in *de novo* sequencing. In contrast to SBH where the target DNA sequence is unknown, these approaches assume that the DNA sequence is either known or "almost" known (i.e., known up to a small number of mutations). For example, to detect genetic variations one can design twenty- to thirty-nucleotide probes to reliably detect mutations, bypassing the still unsolved problem of distinguishing perfect matches from highly stable mismatches in the case of short probes. To detect mutations in

12. In gene expression analysis, a solution containing mRNA (rather than DNA) is applied to the array with the goal of figuring out whether a given gene is switched on or switched off. In this case, absence of a hybridization signal indicates that a gene is not being transcribed into an mRNA, and is therefore switched off.

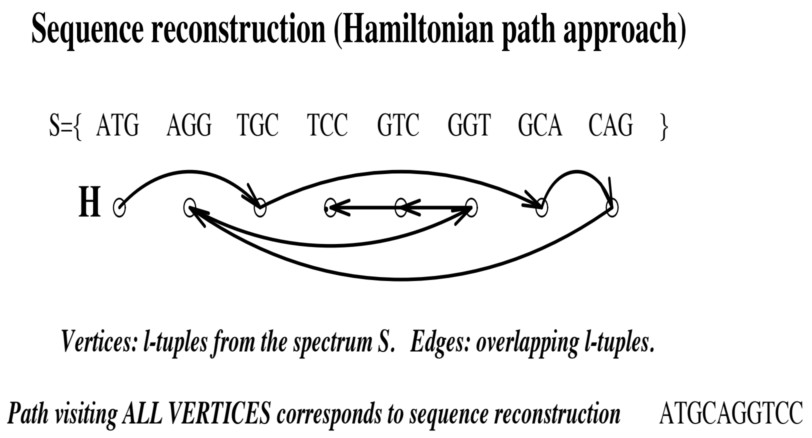

Sequence reconstruction (Hamiltonian path approach)
S={ ATG AGG TGC TCC GTC GGT GCA CAG }
H
Vertices: l-tuples from the spectrum S. Edges: overlapping l-tuples.
Path visiting ALL VERTICES corresponds to sequence reconstruction ATGCAGGTCC

Figure 8.18 SBH and the Hamiltonian path problem.

the (known) sequence S, an array should contain all 25-mers from S, as well as selected mutated versions of these 25-mers.[13]

8.7 SBH as a Hamiltonian Path Problem

Two l-mers p and q *overlap* if $overlap(p,q) = l-1$, that is, the last $l-1$ letters of p coincide with the first $l-1$ letters of q. Given the measured spectrum $Spectrum(s,l)$ of a DNA fragment s, construct a directed graph, H, by introducing a vertex for every l-mer in $Spectrum(s,l)$, and connect every two vertices p and q by the directed edge (p,q) if p and q overlap. There is a one-to-one correspondence between paths that visit each vertex of H exactly once and DNA fragments with the spectrum $Spectrum(s,l)$. The spectrum presented in figure 8.18 corresponds to the sequence reconstruction ATGCAGGTCC, which is the only path visiting all vertices of H:

$$\text{ATG} \rightarrow \text{TGC} \rightarrow \text{GCA} \rightarrow \text{CAG} \rightarrow \text{AGG} \rightarrow \text{GGT} \rightarrow \text{GTC} \rightarrow \text{TCC}$$

The spectrum shown in figure 8.19 yields a more complicated graph with two Hamiltonian paths, each path corresponding to two possible reconstructions: ATGCGTGGCA and ATGGCGTGCA. As the overlap graph becomes

13. In practice, the mutated versions of an l-mer are often limited to the 3 l-mers with mutations at the middle position. Arrays constructed in this manner are called *tiling arrays*.

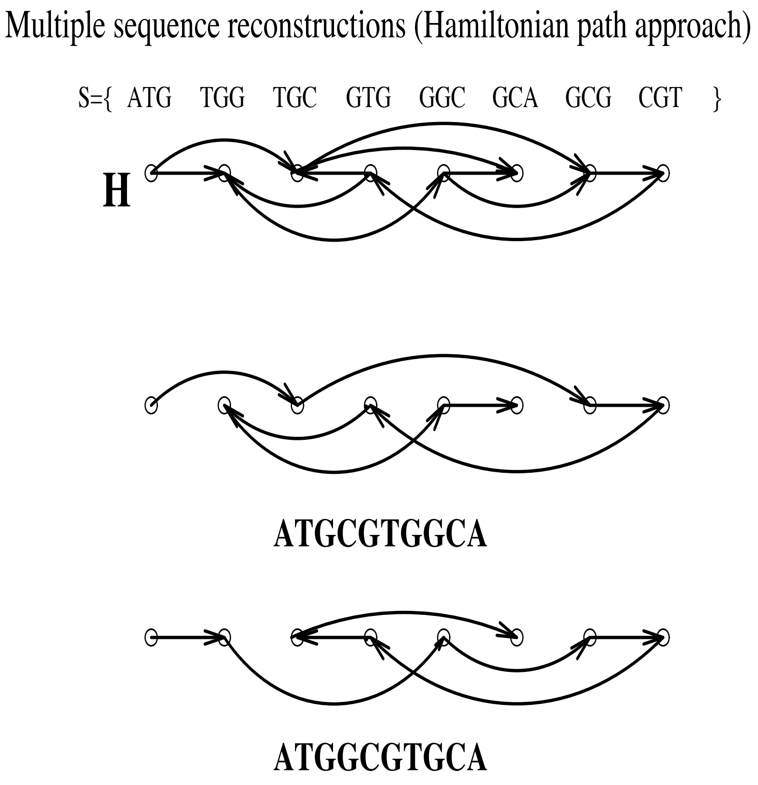

Multiple sequence reconstructions (Hamiltonian path approach)
S={ ATG TGG TGC GTG GGC GCA GCG CGT }
H
ATGCGTGGCA
ATGGCGTGCA

Figure 8.19 Spectrum S yields two possible reconstructions corresponding to distinct Hamiltonian paths.

larger, this approach ceases to be practically useful since the Hamiltonian Path problem is $\mathcal{NP}$-complete.

8.8 SBH as an Eulerian Path Problem

As we have seen, reducing the SBH problem to a Hamiltonian Path problem does not lead to an efficient algorithm. Fortunately, reducing SBH to the *Eulerian Path* problem in a directed graph,[14] which leads to the simple linear-time algorithm for sequence reconstruction mentioned earlier.

14. A directed path is a path $v_1 \to v_2 \to \ldots \to v_n$ from vertex v_1 to vertex v_n in which every edge (v_i, v_{i+1}) is directed from v_i to v_{i+1}.

The reduction of the SBH problem to an Eulerian Path problem is to construct a graph whose edges—rather than vertices—correspond to l-mers from $Spectrum(s, l)$, and then to find a path in this graph visiting every edge exactly once. In this approach we build a graph G on the set of all $(l-1)$-mers, rather than on the set of all l-mers as in the previous section. An $(l-1)$-mer v is joined by a directed edge with an $(l-1)$-mer w if the spectrum contains an l-mer for which the first $l-1$ nucleotides coincide with v and the last $l-1$ nucleotides coincide with w (fig. 8.20). Each l-mer from the spectrum corresponds to a directed edge in G rather than to a vertex as it does in H; compare figures 8.19 and 8.20. Therefore, finding a DNA fragment containing all l-mers from the spectrum corresponds to finding a path visiting all *edges* of G, which is the problem of finding an Eulerian path. Superficially, finding an Eulerian path looks just as hard as finding a Hamiltonian path, but, as we show below, finding Eulerian paths turns out to be simple.

We will first consider *Eulerian cycles*, that is, Eulerian paths in which the first and the last vertices are the same. A directed graph G is *Eulerian* if it contains an Eulerian cycle. A vertex v in a graph is *balanced* if the number of edges entering v equals the number of edges leaving v, that is, if $indegree(v) = outdegree(v)$. For any given vertex v in an Eulerian graph, the number of times the Eulerian cycle enters v is exactly the same as the number of times it leaves v. Thus, $indegree(v) = outdegree(v)$ for every vertex v in an Eulerian graph, motivating the following theorem characterizing Eulerian graphs.

Theorem 8.1 *A connected graph is Eulerian if and only if each of its vertices is balanced.*

Proof First, it is easy to see that if a graph is Eulerian, then each vertex must be balanced. We show that if each vertex in a connected graph is balanced, then the graph is Eulerian.

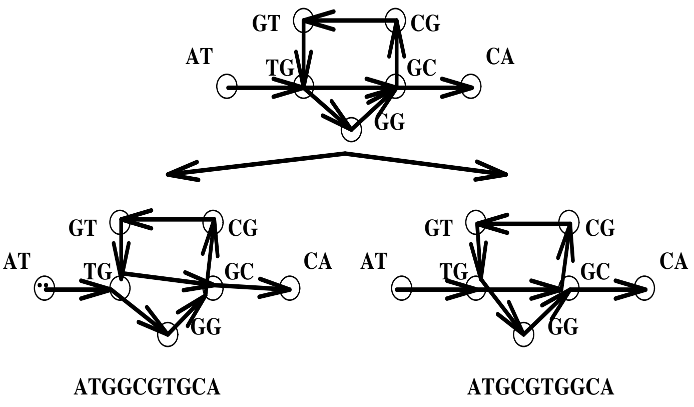
To construct an Eulerian cycle, we start from an arbitrary vertex v and form any arbitrary path by traversing edges that have not already been used. We stop the path when we encounter a vertex with no way out, that is, a vertex whose outgoing edges have already been used in the path. In a balanced graph, the only vertex where this can happen is the starting vertex v since for any other vertex, the balance condition ensures that for every incoming edge there is an outgoing edge that has not yet been used. Therefore, the resulting path will end at the same vertex where it started, and with some luck will be Eulerian. However, if the path is not Eulerian, it must contain a vertex w that still has some number of untraversed edges. The cycle we just constructed

Multiple sequence reconstructions (the Eulerian path approach)

S={ATG, TGG, TGC, GTG, GGC, GCA, GCG , CGT}

Vertices correspond to (l-1)-tuples.

Edges correspond to l-tuples from the spectrum



Paths visiting ALL EDGES correspond to sequence reconstructions

Figure 8.20 SBH and the Eulerian path problem.

forms a balanced subgraph.[15] Since the original graph was balanced, then the edges that were *not* traversed in the first cycle also form a balanced subgraph. Since all vertices in the graph with untraversed edges are balanced there must exist some other path starting and ending at w, containing only untraversed edges. This process is shown in figures 8.21 and 8.22.

One can now combine the two paths into a single one as follows. Traverse the first path from v to w, then traverse the second path from w back to itself, and then traverse the remainder of the first path from w back to v. Repeating this until there are no more vertices with unused edges will eventually yield an Eulerian cycle. This algorithm can be implemented in time linear in the number of edges in the graph. □

15. A subgraph is a graph obtained by removing some edges from the original graph.

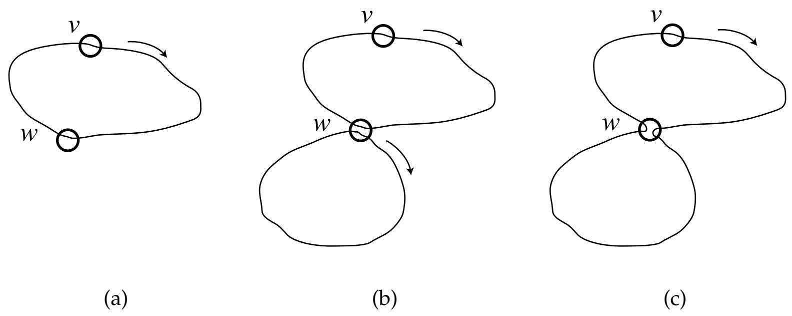


Figure 8.21 Constructing an Eulerian cycle in an Eulerian graph.

Notice that we have described Eulerian graphs as containing an Eulerian *cycle*, rather than an Eulerian path, but we have said that the SBH problem reduces to that of finding an Eulerian path. A vertex v in a graph is called *semibalanced* if $|indegree(v) - outdegree(v)| = 1$. If a graph has an Eulerian path starting at vertex s and ending at vertex t, then all its vertices are balanced, with the possible exception of s and t, which may be semibalanced. The Eulerian path problem can be reduced to the Eulerian cycle problem by adding an edge between two semibalanced vertices. This transformation balances all vertices in the graph and therefore guarantees the existence of an Eulerian cycle in the graph with the added edge. Removing the added edge from the Eulerian cycle transforms it into an Eulerian path. The following theorem characterizes all graphs that contain Eulerian paths.

Theorem 8.2 *A connected graph has an Eulerian path if and only if it contains at most two semibalanced vertices and all other vertices are balanced.*

8.9 Fragment Assembly in DNA Sequencing

As we mentioned previously, after the short 500- to 700-bp *DNA reads* are sequenced, biologists need to assemble them together to reconstruct the entire genomic DNA sequence. This is known as fragment assembly. The Shortest Superstring problem described above is an overly simplified abstraction that does not adequately capture the essence of the fragment assembly problem,

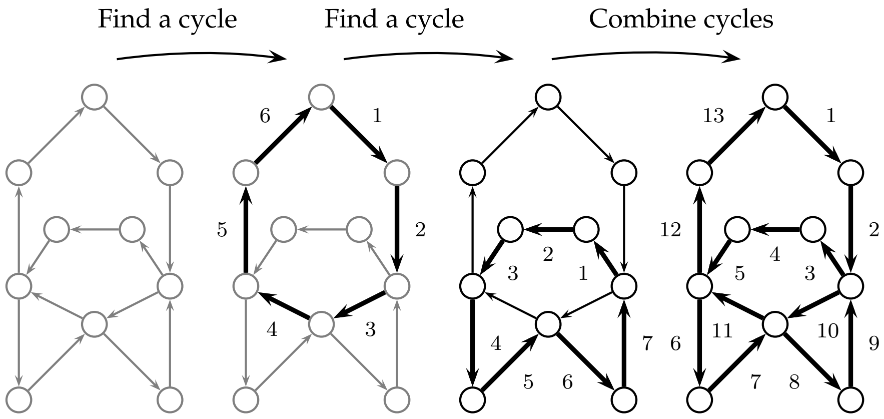


Figure 8.22 Constructing an Eulerian cycle. Untraversed edges are shown in gray.

since it assumes error-free reads. The error rate in DNA reads produced by modern sequencing machines varies from 1% to 3%. A further complication in fragment assembly is the fact that one does not know a priori which of two DNA strands a read came from. DNA is double-stranded, and which of the two strands was sequenced by a read depends on how the insert was oriented in the vector. Since this is essentially arbitrary, one never knows whether a read came from a target strand DNA sequence or from its Watson-Crick complement.

However, sequencing errors and assignments of reads to one of two strands are just minor annoyances compared to the major problem in fragment assembly: repeats in DNA. The human genome contains many sequences that repeat themselves throughout the genome a surprisingly large number of times. For example, the roughly 300 nucleotide *Alu* sequence is repeated more than a million times throughout the genome, with only 5% to 15% sequence variation. Even more troublesome for fragment assembly algorithms is the fact that repeats occur at several scales. The human T-cell receptor locus contains five closely located repeats of the trypsinogen gene, which is 4 kb long and varies only by 3% to 5% between copies. These long repeats are particularly difficult to assemble, since there are no reads with unique portions flanking the repeat region. The human genome contains more than 1 million Alu repeats (≈ 300 bp) and 200,000 $LINE$ repeats (≈ 1000 bp), not to mention that an estimated 25% of genes in the human genome have duplicated copies. A little arithmetic shows that these repeats and duplicated genes represent about half the human genome.[16]

If one models the genome as a 3 billion-letter sequence produced by a random number generator, then assembling it from 500-letter reads is actually relatively simple. However, because of the large number of repeats, it is theoretically impossible to uniquely assemble real reads as long as some repeats are longer than the typical read length. Increasing the length of the reads (to make them longer than most repeats) would solve the problem, but the sequencing technology has not significantly improved the read length yet.

Figure 8.23 (upper) presents a puzzle that looks deceivingly simple and has only sixteen triangular pieces. People usually assemble puzzles by connecting matching pieces. In this case, for every triangle in the puzzle, there is a variety of potentially matching triangles (every frog in the puzzle is repeated several times). As a result, you cannot know which of the potentially matching triangles is the correct one to use at any step. If you proceed without some sort of guidance, you are likely to end up in the situation shown in figure 8.23 (lower). Fourteen of the pieces have been placed completely consistently, but the two remaining pieces are impossible to place. It is difficult to design a strategy that can avoid such dead ends for this particular puzzle, and it is even more difficult to design strategies for the linear puzzle presented by repeats in a genome.

Since repeats present such a challenge in assembling long genomes, the original strategy for sequencing the human genome was first to clone it into BACs,[17] each BAC carrying an approximately $150,000$ bp long insert. After constructing a library of overlapping BACs that covers the entire human genome (which requires approximately 30,000 BACs), each one can be sequenced as if it were a separate minigenome. This BAC-by-BAC sequencing strategy significantly simplifies the computational assembly problem (by virtue of the fact that the number of repeats present within a BAC is 30,000 times smaller than the number of repeats in the entire genome) but makes the sequencing project substantially more cumbersome. Although the Human Genome project demonstrated that this BAC-by-BAC strategy can be successful, recent large-scale sequencing projects (including the 2002 mouse

16. Fortunately, as different copies of these repeats have evolved differently over time, they are not exact repeats.
17. Bacterial Artificial Chromosomes.

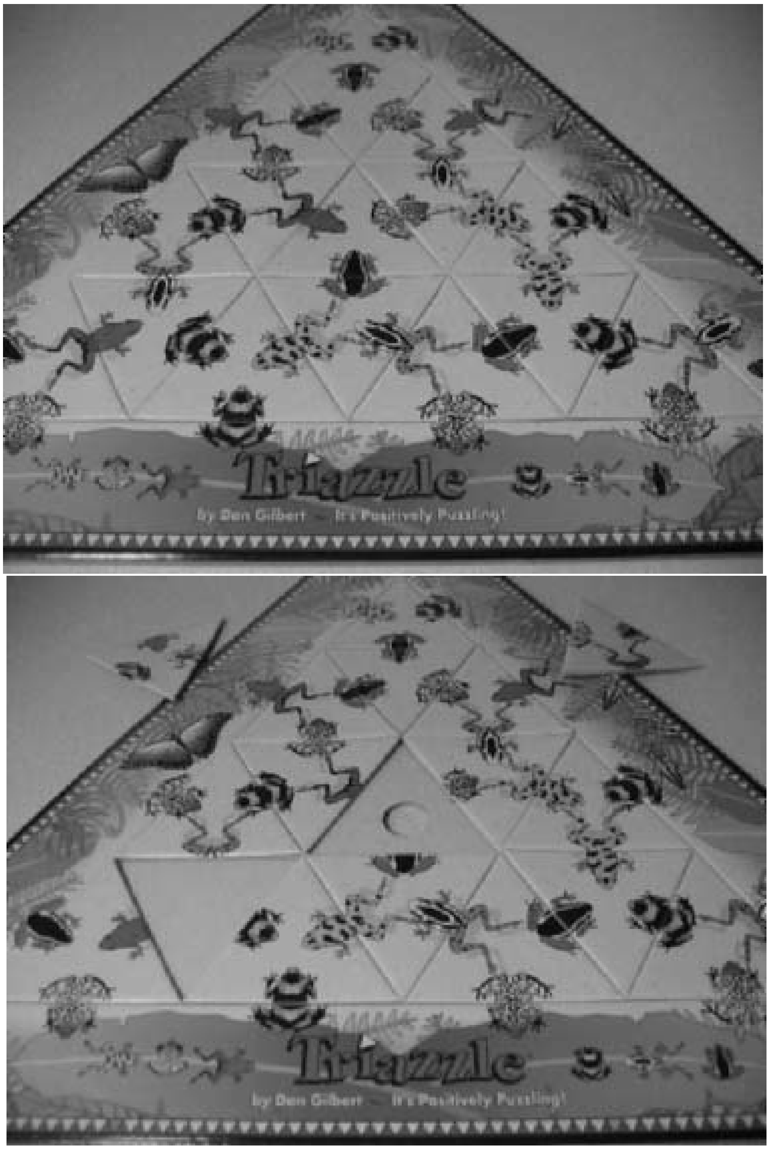

Triazzle
by Dan Gilbert It's Positively Puzzling!
Triazzle
by Dan Gilbert It's Positively Puzzling!

Figure 8.23 Repeats are not just a problem in DNA sequence assembly. This puzzle has deceptively few pieces but is harder than many jigsaw puzzles that have thousands of pieces. (With permission of Dan Gilbert Art Group, Inc.)

genome assembly) mainly follow the whole-genome assembly paradigm advocated by James Weber and Gene Myers in 1997. Myers led the fragment assembly efforts at Celera Genomics, the company that announced the completion of a (draft) human genomic sequence in 2001.[18] Weber and Myers suggested a virtual increase in the length of a read, by pairing reads that were separated by a fixed-size gap. This suggestion resulted in the so-called *mate-pair reads* sequencing technique. In this method, inserts of length approximately L (where L is much longer than the length of a read) are selected, and *both* ends of the insert are sequenced. This produces a pair of reads called *mates* at a known (approximate) distance, L, from each other. The insert length L is chosen in such a way that it is larger than the length of most repeats in the human genome. The advantage of mate-pair reads is that it is unlikely that both reads of the mate-pair will lie in a large-scale DNA repeat. Thus, the read that lies in a unique portion of DNA determines which copy of a repeat its mate is in.

Most fragment assembly algorithms consist of the following three steps:

- *Overlap:* Finding potentially overlapping reads
- *Layout:* Finding the order of reads along DNA
- *Consensus:* Deriving the DNA sequence from the layout

The overlap problem is to find the best match between the suffix of one read and the prefix of another. In the absence of sequencing errors, we could simply find the longest suffix of one string that exactly matches the prefix of another string. However, sequencing errors force us to use a variation of the dynamic programming algorithm for sequence alignment. Since errors are small (1% to 3%), the common practice is to filter out pairs of fragments that do not share a significantly long common substring, an idea we will return to in chapter 9 when we discuss combinatorial pattern matching.

Constructing the layout is the hardest step in fragment assembly. The difficulty is in deciding whether two fragments really overlap (i.e., their differences are caused by sequencing errors) or actually come from two different copies of a repeat. Repeats represent a major challenge for whole-genome shotgun sequencing and make the layout problem very difficult.

The final consensus step of fragment assembly amounts to correcting errors in sequence reads. The simplest way to build the consensus is to report

18. The human genome sequence was sequenced in 2001 by both the publicly-funded Human Genome Consortium and the privately-financed Celera Genomics. As a result, there exist two slightly different versions of the human genome.

the most frequent character in the layout constructed in the layout step. This assumes that each position in the genome was represented by a sufficiently large number of reads to ensure that experimental errors are reduced to minor noise.

8.10 Protein Sequencing and Identification

Few people remember that before DNA sequencing had even been seriously suggested, scientists routinely sequenced proteins. Frederick Sanger was awarded his first (of two) Nobel prize for determining the amino acid sequence of insulin, the protein needed by people suffering from diabetes. Sequencing the 52-amino acid bovine insulin in the late 1940s seemed more challenging than sequencing an entire genome seems today. The computational problem facing protein sequencing at that time was similar to that facing modern DNA sequencing; the main difference was in the length of the sequenced fragments. In the late 1940s, biologists discovered how to apply the *Edman degradation reaction* to chop off one terminal amino acid at a time from the end of a protein and read it. Unfortunately, this only works for a few terminal amino acids before the results become impossible to interpret. To get around this problem, Sanger digested insulin with *proteases* (enzymes that cleave proteins) into *peptides* (short protein fragments) and sequenced each of the resulting fragments independently. He then used these overlapping fragments to reconstruct the entire sequence, exactly like the DNA sequencing "break—read the fragments—assemble" method today, as shown in figure 8.24.

The Edman degradation reaction became the predominant protein sequencing method for the next twenty years, and by the late 1960s protein sequencing machines were on the market. Despite these advances, protein sequencing ceased to be of central interest in the field as DNA sequencing technology underwent rapid improvements in the late 1970s. In DNA sequencing, obtaining reads is relatively easy; it is the assembly that is difficult. In protein sequencing, obtaining reads is the primary problem, while assembly is easy.[19]

Having DNA sequence data for a cell is critical to understanding the molecular processes that the cell goes through. However, it is not the only impor-

19. Modern protein sequencing machines are capable of reading more than fifty residues from a peptide fragment. However, these machines work best when the protein is perfectly purified, which is hard to achieve in biological experiments.

```
GIVE
GIVEECCA
GIVEECCASV
GIVEECCASVC
GIVEECCASVCSL
GIVEECCASVCSLY
        SVC
           SLY
           SLYELEDYC
             YE
             YEL
             YELE
              ELEDY
              ELEDYCD
               LE
               LEDYCD
                EDYCD
                 DYCD
                   CD

                     FVDEHLCG
                     FVDEHLCGSHL
                         HLCGSHL
                             SHLVEA
                                VEAL
                                VEALY
                                  AL
                                  ALY
                                    YLVCG
                                     LVCGERGF
                                     LVCGERGFF
                                        GERG
                                           GF
                                           GFFYTPK
                                              YTPKA
                                               TPKA
```

Figure 8.24 The peptide fragments that Frederick Sanger obtained from insulin through a variety of methods. The protein is split into two parts, the A-chain (shown on the left) and the B-chain (shown on the right) as a result of an enzymatic digestion process. Sanger's further elucidation of the disulfide bridges linking the various cysteine residues was the result of years of painstaking laboratory work. The sequence was published in three parts: the A-chain, the B-chain, and then the disulfide linkages. Insulin is not a particularly large protein, so better techniques would be useful.

tant component: one also needs to know what proteins the cell produces and what they do. On the one hand, we do not yet know the full set of proteins that cells produce, so we need a way to discover the sequence of previously unknown proteins. On the other hand, it is important to identify which specific proteins interact in a biological system (e.g., those proteins involved in DNA replication). Lastly, different cells in an organism have different repertoires of expressed proteins. Brain cells need different proteins to function than liver cells do, and an important problem is to identify proteins that are present or absent in each biological tissue under different conditions.

There are two types of computational problems motivated by protein sequencing. *De novo protein sequencing* is the elucidation of a protein's sequence in the case when a biological sample contains a protein that is either not present in a database or differs from a canonical version present in a database (e.g., mutated proteins or proteins with biochemical modifications). The other problem is the identification of a protein that is present in a database; this is usually referred to as *protein identification*. The main difference between protein sequencing algorithms and protein identification algorithms is the difficulty of the underlying computational problems.

Perhaps the easiest way to illustrate the distinction between protein identification and sequencing is with a gedanken experiment. Suppose a biologist wants to determine which proteins form the DNA polymerase complex in rats. Having the complete rat genome sequence and knowing the location of all the rat genes does not yet allow a biologist to determine what chemical reactions occur during the DNA replication process. However, isolating a rat's DNA polymerase complex, breaking it apart, and sequencing the proteins that form parts of the complex will yield a fairly direct answer to the researcher's question. Of course, if we presume that the biologist has the complete rat genome sequence and all of its gene *products*, he may not actually have to sequence every amino acid in every protein in the DNA polymerase complex—just enough to figure out which proteins are present. This is protein identification. On the other hand, if the researcher decides to study an organism for which complete genome data are not available (perhaps an obscure species of ant), then the researcher will need to perform de novo protein sequencing.[20]

20. As usual with gedanken experiments, reality is more complicated. Even if the complete genomic sequence of a species is known and annotated, the repertoire of all possible proteins usually is not, due to the myriad alternative splicings (different ways of constructing mRNA from the gene's transcript) and post-translational modifications that occur in a living cell.

For many problems, protein sequencing and identification remain the only ways to probe a biological process. For example, gene splicing (see chapter 6) is a complex process performed by the large molecular complex called the *spliceosome*, which consists of over 100 different proteins complexed with some functional RNA. Biologists want to determine the "parts list" of the spliceosome, that is, the identity of proteins that form the complex. DNA sequencing is not capable of solving this problem directly: even if all the proteins in the genome were known, it is not clear which of them are parts of the spliceosome. Protein sequencing and identification, on the other hand, are very helpful in discovering this parts list. Recently, Matthias Mann and colleagues purified the spliceosome complex and used protein sequencing and protein identification techniques to find a detailed parts list for it.

Another application of these technologies is the study of proteins involved in *programmed cell death*, or *apoptosis*. In the development of many organisms cells must die at specific times. A cell dies if it fails to acquire certain survival factors, and the death process can be initiated by the expression of certain genes. In a developing nematode, for example, the death of individual cells in the nervous system may be prevented by mutations in several genes that are the subject of active investigation. DNA sequence data alone are not sufficient to find the genes involved in programmed cell death, and until recently, nobody knew the identity of these proteins. Protein analysis by *mass spectrometry* allowed the sequencing of proteins involved in programmed cell death, and the discovery of some proteins involved in the death-inducing signaling complex.

The exceptional sensitivity of mass spectrometry has opened up new experimental and computational possibilities for protein studies. A protein can be digested into peptides by proteases like trypsin. In a matter of seconds, a *tandem mass spectrometer* breaks a peptide into even smaller fragments and measures the mass of each. The *mass spectrum* of a peptide is a collection of masses of these fragments. The protein sequencing problem is to derive the sequence of a peptide given its mass spectrum. For an ideal fragmentation process where every fragment of a peptide is generated, and in an ideal mass spectrometer, the peptide sequencing problem is simple. However, the fragmentation process is not ideal, and mass spectrometers measure mass with some imprecision. These details make peptide sequencing difficult.

A mass spectrometer works like a charged sieve. A large molecule (peptide) gets broken into smaller fragments that have an electrical charge. These fragments are then spun around and accelerated in a magnetic field until they hit a detector. Because large fragments are harder to spin than small ones, one can distinguish between fragments with different masses based on the amount of energy required to fling the different fragments around. It happens that most molecules can be broken in several places, generating several different *ion types*. The problem is to reconstruct the amino acid sequence of the peptide from the masses of these broken pieces.

8.11 The Peptide Sequencing Problem

Let $A = \{a_1, a_2, \dots, a_{20}\}$ be the set of amino acids, each with molecular masses $m(a_i)$. A *peptide* $P = p_1 \cdots p_n$ is a sequence of amino acids, with *parent* mass $m(P) = \sum_{i=1}^{n} m(p_i)$. We will denote the partial *N-terminal* peptide $p_1, \dots, p_i$ of mass $m_i = \sum_{j=1}^{i} m(p_j)$ as P_i and the partial *C-terminal* peptide $p_{i+1}, \dots, p_n$ of mass $m(P) - m_i$ as P_i^-, for $1 \leq i \leq n$. Mass spectra obtained by tandem mass spectrometry (MS/MS) consist predominantly of partial N-terminal peptides and C-terminal peptides.[21]

A mass spectrometer typically breaks a peptide $p_1 p_2 \cdots p_n$ at different peptide bonds and detects the masses of the resulting partial N-terminal and C-terminal peptides.[22] For example, the peptide GPFNA may be broken into the N-terminal peptides G, GP, GPF, GPFN, and C-terminal peptides PFNA, FNA, NA, A. Moreover, while breaking GPFNA into GP and FNA, it may lose some small parts of GP and FNA, resulting in fragments of a lower mass. For example, the peptide GP might lose a water (H_2O), and the peptide FNA might lose an ammonia (NH_3). The resulting masses detected by the spectrometer will be equal to the mass of GP minus the mass of water (water happens to weigh $1 + 1 + 16 = 18$ daltons) , and the mass of FNA minus the mass of ammonia ($1 + 1 + 1 + 14 = 17$ daltons). Peptides missing water and ammonia are two different *ion types* that can occur in fragmenting a peptide in a mass spectrometer.[23]

21. Every protein is a linear chain of amino acids, connected by a peptide bond. The peptide bond starts with a nitrogen (N) and ends with a carbon (C); therefore, every protein begins with an "unstarted" peptide bond that begins with N and another "unfinished" peptide bond that ends with C. An *N-terminal* peptide is a fragment of a protein that includes the "leftmost" end (i.e., the N-terminus). A *C-terminal* peptide is a fragment of a protein that includes the "rightmost" end (i.e., the C-terminus).

22. Biologists typically work with billions of identical peptides in a solution. A mass spectrometry machine breaks different peptide molecules at different peptide bonds (some peptide bonds are more prone to breakage than others). As a result, many N-terminal and C-terminal peptide may be detected by a mass spectrometer.

23. This is a simplified description of the complex and messy fragmentation process. In this section we intentionally hide many of the technical details and focus only on the computational challenges.

Peptide fragmentation in a tandem mass spectrometer can be characterized by a set of numbers $\Delta = \{\delta_1, \ldots, \delta_k\}$ representing the different types of ions that correspond to the removal of a certain chemical group from a peptide fragment. We will call Δ the set of ion types. A *δ-ion* of an N-terminal partial peptide P_i is a modification of P_i that has mass $m_i - \delta$, corresponding to the loss of a (typically small) chemical group of mass δ when P was fragmented into P_i. The *δ-ion* of C-terminal peptides is defined similarly. The most frequent N-terminal ions are called b-ions (ion b_i corresponds to P_i with $\delta = -1$) and the most frequent C-terminal ions are called y-ions (ion y_i corresponds to P_i^- with $\delta = 19$), shown in figure 8.25 (a). Examples of other frequent N-terminal ions are represented by b-H_2O (a b-fragment that loses a water) or y-NH_3 and some others like b-H_2O-NH_3.

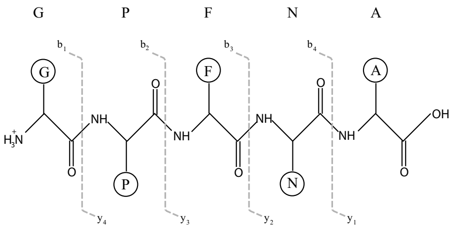
For tandem mass spectrometry, the *theoretical* spectrum $T(P)$ of peptide P can be calculated by subtracting all possible ion types $\delta_1, \ldots, \delta_k$ from the masses of all partial peptides of P, such that every partial peptide generates k masses in the theoretical spectrum, as in figure 8.25 (b).[24]

An *experimental* spectrum $S = \{s_1, \ldots, s_q\}$ is a set of numbers obtained in a mass spectrometry experiment that includes masses of some fragment ions as well as chemical noise.[25] Note that the distinction between the theoretical spectrum $T(P)$ and the experimental spectrum S is that you mathematically generate $T(P)$ given the peptide sequence P, but you experimentally generate S without knowing what the peptide sequence is that generated it.

The *match* between the experimentally measured spectrum S and peptide P is the number of masses in S that are equal to masses in $T(P)$. This is often referred to as the *shared peaks count*. In reality, peptide sequencing algorithms use more sophisticated objective functions than a simple shared peaks count, incorporating different weighting functions for the matching masses. We formulate the Peptide Sequencing problem as follows.

24. A theoretical spectrum of a peptide may contain as many as $2nk$ masses but it sometimes contains less since some of these masses are not unique.

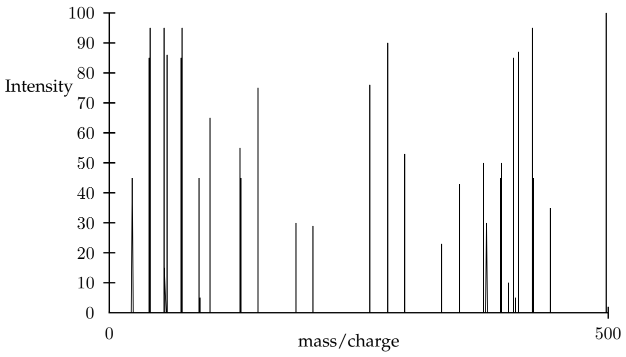
25. In reality, a mass spectrometer detects *charged ions* and measures mass-to-charge ratios. As a result, an experimental spectrum contains the values $\frac{m}{z}$ where m is the mass and z is an integer (typically, 1 or 2) equal to the charge of a fragment ion. For simplicity we assume that $z = 1$ through the remainder of this chapter.



(a) The fragmentation pattern of the peptide GPFNA.

	Sequence	mass	less H_2O	less NH_3	less both
	GPFNA	498	480	481	463
b_1	G	58	40	41	23
y_4	PFNA	442	424	425	405
b_2	GP	149	131	132	114
y_3	FNA	351	333	334	316
b_3	GPF	296	278	279	261
y_2	NA	204	186	187	169
b_4	GPFN	410	392	393	375
y_l	A	90	72	73	55

(b) A theoretical mass spectrum of GPFNA.



(c) An "experimental" mass spectrum of GPFNA.

Figure 8.25 Tandem MS of the peptide $GPFNA$. Two different types of fragment ions, b-ions and y-ions are created (a) when the carbon-nitrogen bond breaks in the spectrometer. Each of these ion types can also lose H_2O or NH_3, or both, resulting in the masses presented in (b). Many other ion types are seen in typical experiments. If we were to measure the mass spectrum of this peptide, we would see a result similar to (c), where some peaks are missing and other noise peaks are present.

Peptide Sequencing Problem:
Find a peptide whose theoretical spectrum has a maximum match to a measured experimental spectrum.

> **Input:** Experimental spectrum S, the set of possible ion types Δ, and the parent mass m.
>
> **Output:** A peptide P of mass m whose theoretical spectrum matches S better than any other peptide of mass m.

In reality, mass spectrometers measure both mass and *intensity*, which reflects the number of fragment ions of a given mass are detected in the mass spectrometer. As a result, mass spectrometrists often represent spectra in two dimensions, as in figure 8.25 (c), and refer to the masses in the spectrum as "peaks."[26]

8.12 Spectrum Graphs

There are two main approaches to solving the Peptide Sequencing problem that researchers have tried: either through exhaustive search among all amino acid sequences of a certain length, or by analyzing the *spectrum graph* which we define below. The former approach involves the generation of all 20^l amino acid sequences of length l and their corresponding theoretical spectra, with the goal of finding a sequence with the best match between the experimental spectrum and the sequence's theoretical spectrum. Since the number of sequences grows exponentially with the length of the peptide, different branch-and-bound techniques have been designed to limit the combinatorial explosion in these methods. *Prefix pruning* restricts the computational space to sequences whose prefixes match the experimental spectrum well. The difficulty with the prefix pruning is that it frequently discards the correct sequence if its prefixes are poorly represented in the spectrum.

The spectrum graph approach, on the other hand, does not involve generating all amino acid sequences, and leads to a fast algorithm for peptide sequencing. In this approach, we construct a graph from the experimental spectrum. Assume for simplicity that an experimental spectrum $S =$

26. A match of a theoretical spectrum against an experimental spectrum with varying intensity, then, needs to reflect the intensity of the fragment ions. While accounting for intensities is important for statistical analysis, it does not seriously affect the algorithmic details and we ignore intensities in the remainder of this chapter.

$\{s_1, \ldots, s_q\}$ consists of N-terminal ions and we will ignore the C-terminal ions for a while. Every mass $s \in S$ may have been created from a partial peptide by one of the k different ion types. Since we do not know which ion type from $\Delta = (\delta_1, \ldots, \delta_k)$ created the mass s in the experimental spectrum, we generate k different "guesses" for each of masses in the experimental spectrum. Every guess corresponds to the hypothesis that $s = x - \delta_j$, where x is the mass of some partial peptide and $1 \leq j \leq k$. Therefore, for every mass s in the experimental spectrum, there are k guesses for the mass x of some partial peptide: $s + \delta_1, s + \delta_2, \ldots, s + \delta_k$. As a result, each mass in the experimental spectrum is transformed into a set of k vertices in the spectrum graph, one for each possible ion type. The vertex for δ_i for the mass s is labeled with mass $s + \delta_i$. We connect any two vertices u and v in the graph by the directed edge (u, v) if the mass of v is larger than that of u by the mass of a single amino acid. If we add a vertex at 0 and a vertex at the parent mass m (connecting them to other vertices as before), then the Peptide Sequencing problem can be cast as finding a path from 0 to m in the resulting DAG.[27]

In summary, the vertex set of the resulting spectrum graph is a set of numbers $s_i + \delta_j$ representing potential masses of N-terminal peptides adjusted by the ion type δ_j. Every mass s_i of spectrum S generates k distinct vertices $V_i(s) = \{s_i + \delta_1, \ldots, s_i + \delta_k\}$, though the sets V_i and V_j may overlap if s_i and s_j are close. The set of vertices in a spectrum graph is therefore $\{s_{initial}\} \cup V_1 \cup \cdots \cup V_q \cup \{s_{final}\}$, where $s_{initial} = 0$ and $s_{final} = m$. The spectrum graph may have at most $qk + 2$ vertices. We label the edges of the spectrum graph by the amino acid whose mass is equal to difference between vertex masses. If we look at vertices as putative N-terminal peptides,[28] the edge from u to v implies that the N-terminal sequence corresponding to v may be obtained by extending the sequence at u by the amino acid that labels (u, v).

A spectrum S of a peptide $P = p_1 \ldots p_n$ is called *complete* if S contains at least one ion type corresponding to every N-terminal partial peptide P_i for every $1 \leq i \leq n$. The use of a spectrum graph is based on the observation that for a complete spectrum there exists a path of length $n+1$ from $s_{initial}$ to s_{final} in the spectrum graph that is labeled by P. This observation casts the Peptide Sequencing problem as one of finding the "correct" path in the set of all paths between two vertices in a directed acyclic graph. If the spectrum

27. In addition to the experimental spectrum, every mass spectrometry experiment always produces the parent mass m of a peptide.

28. Although we ignored C-terminal ions in this simplified construction of the spectrum graph, these ions can be taken into account by combining the spectrum S with its "reversed" version.

is complete, then the correct path that we are looking for is often the path with the maximum number of edges, the familiar Longest Path in a DAG problem.

Unfortunately, experimental spectra are frequently incomplete. Moreover, even if the experimental spectrum is complete, there are often many paths in the spectrum graph to choose from that have the same (or even larger) length, preventing one from unambiguously reconstructing the peptide.

The problem with choosing a path with a maximum number of edges is that it does not adequately reflect the "importance" of different vertices. For example, a vertex in the spectrum graph obtained by a shift of $+1$ as $s_i + 1$ (corresponding to the most frequent b-ions) should be scored higher than a vertex obtained by a shift of the rare b-H_2O-NH_3 ion ($s_i + 1 - 18 - 17 = s_i - 34$). Further, whenever there are two peaks s_i and $s_{i'}$ such that $s_i + \delta_j = s_{i'} + \delta_{j'}$, the vertex corresponding to that mass should also get a higher score than a vertex obtained by a single shift.

In the probabilistic approach to peptide sequencing, each ion type δ_i has some probability of occurring, which we write as $p(\delta_i)$. Under the simplest assumption, the probability that δ_i occurs for some partial peptide is independent of whether δ_j also occurs for the same partial peptide. Under this assumption, any given partial peptide may contribute as many as k masses in the spectrum [this happens with probability $\prod_{i=1}^{k} p(\delta_i)$] and as few as 0 [this happens with probability $\prod_{i=1}^{k}(1 - p(\delta_i))$]. The probabilistic model below scores the vertices of the spectrum graph based on these simple assumptions.

Suppose that an N-terminal partial peptide P_i with mass m_i produces ions $\delta_1, \ldots, \delta_l$ ("present" ions of mass $m_i - \delta_1, m_i - \delta_2, \ldots, m_i - \delta_l$) but fails to produce ions $\delta_{l+1}, \ldots, \delta_k$ ("missing" ions) in the experimental spectrum. All l present ions will result in a vertex in the spectrum graph at mass m_i, corresponding to P_i. How should we score this vertex? A naive approach would be to reward P_i for every ion type that explains it, suggesting a score of $\prod_{i=1}^{l} p(\delta_i)$. However, this approach has the disadvantage of not considering the missing ions, so we combine those in by defining the score for the partial peptide to be

$$\left(\prod_{i=1}^{l} p(\delta_i)\right)\left(\prod_{i=l+1}^{k} (1 - p(\delta_i))\right).$$

However, there is some inherent probability of chemical noise, that is, it can produce *any* mass (that has nothing to do with a peptide of interest) with certain probability p_R. Therefore, we adjust the probabilistic score as

$$\left(\prod_{i=1}^{l} \frac{p(\delta_i)}{p_R}\right)\left(\prod_{i=l+1}^{k} \frac{1-p(\delta_i)}{1-p_R}\right).$$

8.13 Protein Identification via Database Search

De novo protein sequencing algorithms are invaluable for identification of (both known and unknown) proteins, but they are most useful when working with complete or nearly complete high-quality spectra. Many spectra are far from complete, and de novo peptide sequencing algorithms often produce ambiguous solutions for such spectra. If we had access to a database of all proteins from a genome, then we would no longer need to consider all 20^l peptide sequences to interpret an MS/MS spectrum, but could instead limit our search to peptides present in this database.

Currently, most proteins are identified by database search— effectively looking the answer up in "the back of a book." Indeed, an experimental spectrum can be compared with the theoretical spectrum for each peptide in such a database, and the entry in the database that best matches the observed spectrum usually provides the sequence of the experimental peptide. This forms the basis of the popular SEQUEST algorithm developed by John Yates and colleagues. We formulate the Protein Identification problem as follows.

Protein Identification Problem:
Find a protein from a database that best matches the experimental spectrum.

Input: A database of proteins, an experimental spectrum S, a set of ion types Δ, and a parent mass m.

Output: A protein of mass m from the database with the best match to spectrum S.

Though the logic that SEQUEST uses to determine whether a database entry matches an experimental spectrum is somewhat involved, the basic approach of the algorithm is just a linear search through the database. One drawback to MS/MS database search algorithms like SEQUEST is that peptides in a cell are often slightly different from the "canonical" peptides present in databases. The synthesis of proteins on ribosomes is not the final step in a protein's life: many proteins are subject to further modifications that regulate protein activities and these modifications may be either permanent or reversible. For example, the enzymatic activity of some proteins is regulated by the addition or removal of a phosphate group at a specific residue.[29] *Phosphorylation* is a reversible process: *protein kinases* add phosphate groups while *phosphatases* remove them.

Proteins form complex systems necessary for cellular signaling and metabolic regulation, and are therefore often subject to a large number of biochemical modifications (e.g., phosphorylation and glycosylation). In fact, almost all protein sequences are modified after they have been constructed from their mRNA template, and as many as 200 distinct types of modifications of amino acid residues are known. Since we are unable to predict these *post-translational modifications* from DNA sequences, finding naturally occurring modifications remains an important open problem. Computationally, a chemical modification of the protein $p_1p_2\cdots p_i\cdots p_n$ at position i results in increased mass of the N-terminal peptides $P_i, P_{i+1}, \ldots, P_n$ and increased mass of the C-terminal peptides $P_1^-, P_2^-, \ldots, P_{i-1}^-$.

The computational analysis of modified peptides was also pioneered by John Yates, who suggested an exhaustive search approach that (implicitly) generates a virtual database of all modified peptides from a small set of potential modifications, and matches the experimental spectrum against this virtual database. It leads to a large combinatorial problem, even for a small set of modification types.

Modified Protein Identification Problem:
Find a peptide from the database that best matches the experimental spectrum with up to k modifications.

Input: A database of proteins, an experimental spectrum S, a set of ion types Δ, a parent mass m, and a parameter k capping the number of modifications.

Output: A protein of mass m with the best match to spectrum S that is at most k modifications away from an entry in the database.

The major difficulty with the Modified Protein Identification problem is

29. Phosphorylation uses serine, threonine, or tyrosine residues to add a phosphate group.

that very similar peptides P_1 and P_2 may have very different spectra S_1 and S_2.[30] Our goal is to define a notion of spectral similarity that correlates well with sequence similarity. In other words, if P_1 and P_2 are a few modifications apart, the spectral similarity between S_1 and S_2 should be high. The shared peaks count is, of course, an intuitive measure of spectral similarity. However, this measure diminishes very quickly as the number of mutations increases, thus leading to limitations in detecting similarities by database search. Moreover, there are many correlations between the spectra of related peptides, and only a small proportion of these correlations is captured by the shared peaks count.

The spectral convolution algorithm, below, reveals potential peptide modifications without an exhaustive search and therefore does not require generating a virtual database of modified peptides.

8.14 Spectral Convolution

Let S_1 and S_2 be two spectra. Define the *spectral convolution* to be the multiset $S_2 \ominus S_1 = \{s_2 - s_1 : \ s_1 \in S_1, s_2 \in S_2\}$ and let $(S_2 \ominus S_1)(x)$ be the multiplicity of element x in this multiset. In other words, $(S_2 \ominus S_1)(x)$ is the number of pairs $(s_1 \in S_1, s_2 \in S_2)$ such that $s_2 - s_1 = x$ (fig. 8.26).

The shared peak count that we introduced earlier in this chapter is the number of masses common to both S_1 and S_2, and is simply $(S_2 \ominus S_1)(0)$. MS/MS database search algorithms that maximize the shared peak count find a peptide in the database that maximizes $(S_2 \ominus S_1)(0)$, where S_2 is an experimental spectrum and S_1 is the theoretical spectrum of a peptide in the database. However, if S_1 and S_2 correspond to peptides that differ by k mutations or modifications, the value of $(S_2 \ominus S_1)(0)$ may be too small to determine that S_1 and S_2 really were generated by similar peptides. As a result, the power of the shared peak count to discern that two peptides are similar diminishes rapidly as the number of modifications increases—it is bad at $k = 1$, and nearly useless for $k > 1$.

The peaks in the spectral convolution allow us to detect mutations and modifications, even if the shared peak count is small. If peptides P_2 and P_1 (corresponding to spectra S_2 and S_1) differ by only one mutation ($k = 1$) with amino acid difference $\delta = m(P_2) - m(P_1)$, then $S_2 \ominus S_1$ is expected to have two approximately equal peaks at $x = 0$ and $x = \delta$. If the mutation

30. P_1 corresponds to a peptide from the database, while P_2 corresponds to the modified version of P_1, whose experimental spectrum is being used to search the database.

occurs at position t in the peptide, then the peak at $(S_2 \ominus S_1)(0)$ corresponds to N-terminal peptides P_i for $i < t$ and C-terminal peptides P_i^- for $i \geq t$. The peak at $(S_2 \ominus S_1)(\delta)$ corresponds to N-terminal peptides P_i for $i \geq t$ and C-terminal peptides P_i^- for $i < t$.

Now assume that P_2 and P_1 are two substitutions apart, one with mass difference δ' and another with mass difference $\delta - \delta'$, where δ denotes the difference between the parent masses of P_1 and P_2. These modifications generate two new peaks in the spectral convolution at $(S_2 \ominus S_1)(\delta')$ and at $(S_2 \ominus S_1)(\delta - \delta')$. It is therefore reasonable to define the similarity between spectra S_1 and S_2 as the overall height of the k highest peaks in $S_2 \ominus S_1$.

Although spectral convolution helps to identify modified peptides, it does have a limitation. Let

$$S = \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$$

be a spectrum of peptide P, and assume for simplicity that P produces only b-ions. Let

$$S' = \{10, 20, 30, 40, 50, 55, 65, 75, 85, 95\}$$

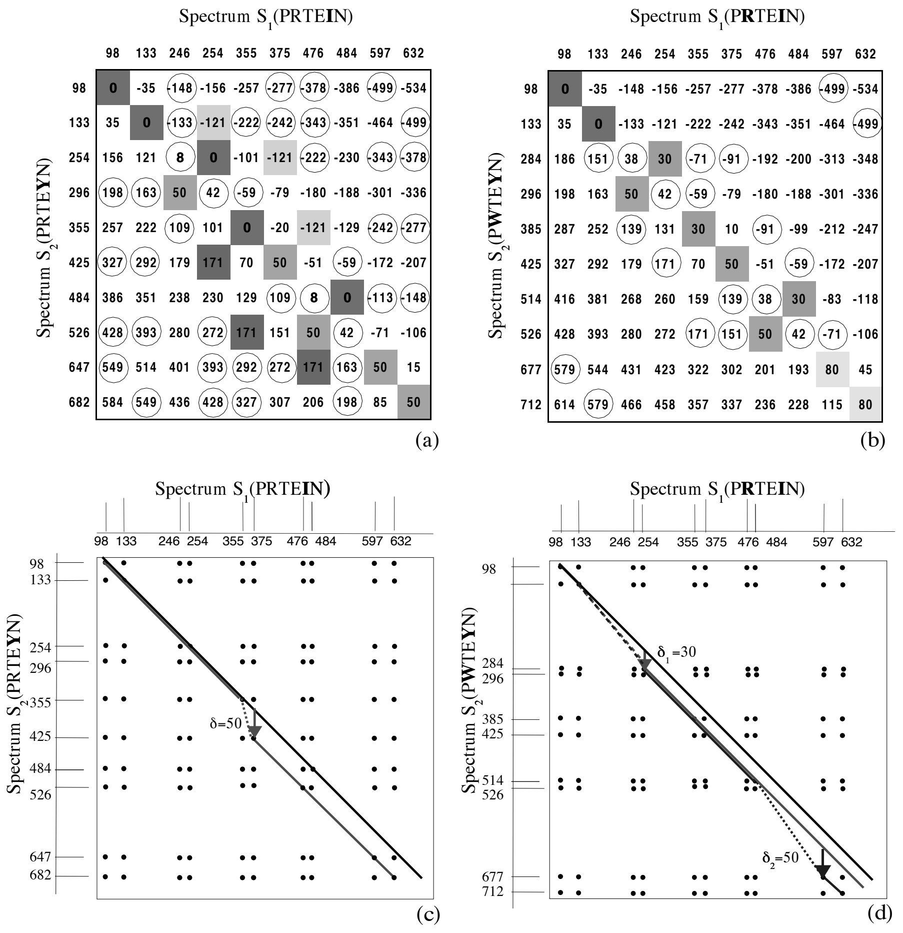
and

$$S'' = \{10, 15, 30, 35, 50, 55, 70, 75, 90, 95\}$$

be two theoretical spectra corresponding to peptides P' and P'' from the database. Which of the two peptides fits S better? The shared peaks count does not allow one to answer this question, since both S' and S'' have five peaks in common with S. Moreover, the spectral convolution also does not answer this question, since both $S \ominus S'$ and $S \ominus S''$ reveal strong peaks of the same height at 0 and 5. This suggests that both P' and P'' can be obtained from P by a single mutation with mass difference 5. However, a more careful analysis shows that although this mutation can be realized for P' by introducing a shift 5 after mass 50, it cannot be realized for P''. The major difference between S' and S'' is that the matching positions in S' come in clumps while the matching positions in S'' do not. Below we describe the spectral alignment approach, which addresses this problem.

8.15 Spectral Alignment

Let $A = \{a_1, \ldots, a_n\}$ be an ordered set of integers $a_1 < a_2 < \cdots < a_n$. A *shift* Δ_i transforms A into $\{a_1, \ldots a_{i-1}, a_i + \Delta_i, \ldots, a_n + \Delta_i\}$. That is, Δ_i alters all elements in the sequence except for the first $i - 1$ elements. We only



(a)

Spectrum S_1(PRTEIN) / Spectrum S_2(PRTEYN)

	98	133	246	254	355	375	476	484	597	632
98	0	-35	-148	-156	-257	-277	-378	-386	-499	-534
133	35	0	-133	-121	-222	-242	-343	-351	-464	-499
254	156	121	8	0	-101	-121	-222	-230	-343	-378
296	198	163	50	42	-59	-79	-180	-188	-301	-336
355	257	222	109	101	0	-20	-121	-129	-242	-277
425	327	292	179	171	70	50	-51	-59	-172	-207
484	386	351	238	230	129	109	8	0	-113	-148
526	428	393	280	272	171	151	50	42	-71	-106
647	549	514	401	393	292	272	171	163	50	15
682	584	549	436	428	327	307	206	198	85	50

(b)

Spectrum S_1(PRTEIN) / Spectrum S_2(PWTEYN)

	98	133	246	254	355	375	476	484	597	632
98	0	-35	-148	-156	-257	-277	-378	-386	-499	-534
133	35	0	-133	-121	-222	-242	-343	-351	-464	-499
284	186	151	38	30	-71	-91	-192	-200	-313	-348
296	198	163	50	42	-59	-79	-180	-188	-301	-336
385	287	252	139	131	30	10	-91	-99	-212	-247
425	327	292	179	171	70	50	-51	-59	-172	-207
514	416	381	268	260	159	139	38	30	-83	-118
526	428	393	280	272	171	151	50	42	-71	-106
677	579	544	431	423	322	302	201	193	80	45
712	614	579	466	458	357	337	236	228	115	80

Figure 8.26 Detecting modifications of the peptide $PRTEIN$. (a) Elements of the spectral convolution $S_2 \ominus S_1$ represented as elements of a difference matrix. S_1 and S_2 are the theoretical spectra of the peptides $PRTEIN$ and $PRTEYN$, respectively. Elements in the spectral convolution that have multiplicity larger than 2 are shaded, while the elements with multiplicity exactly 2 are shown circled. The high multiplicity element 0 corresponds to all of the shared masses between the two spectra, while another high multiplicity element (50) corresponds to the shift of masses by $\delta = 50$, due to the mutation of I to Y in $PRTEIN$ (the difference in mass between Y and I is 50). In (b), two mutations have occurred in $PRTEIN$: $R \rightarrow W$ with $\delta' = 30$, and $I \rightarrow Y$ with $\delta'' = 50$. Spectral alignments for (a) and (b) are shown in (c) and (d), respectively. The main diagonals represent the paths for $k = 0$. The lines parallel to the main diagonals represent the paths for $k > 0$. Every jump between diagonals corresponds to an increase in k. Mutations and modifications to a peptide can be detected as jumps between the diagonals.

consider shifts that do not change the order of elements, that is, the shifts with $\Delta_i \geq a_{i-1} - a_i$. The *k-similarity*, $D(k)$, between sets A and B is defined as the maximum number of elements in common between these sets after k shifts. For example, a shift -5_6 transforms

$$S = \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$$

into

$$S' = \{10, 20, 30, 40, 50, \mathbf{55}, \mathbf{65}, \mathbf{75}, \mathbf{85}, \mathbf{95}\}.$$

Therefore $D(1) = 10$ for these sets. The set

$$S'' = \{10, 15, 30, 35, 50, 55, 70, 75, 90, 95\}$$

has five elements in common with S (the same as S') but there is no single shift transforming S into S'' ($D(1) = 6$). Below we analyze and solve the following Spectral Alignment problem:

Spectral Alignment Problem:
Find the k-similarity between two sets.

Input: Sets A and B, which represent the two spectra, and a number k (number of shifts).

Output: The k-similarity, $D(k)$, between sets A and B.

One can represent sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_m\}$ as 0–1 arrays $\mathbf{a}$ and $\mathbf{b}$ of length a_n and b_m correspondingly. The array $\mathbf{a}$ will contain n ones (at positions $a_1, \ldots, a_n$) and $a_n - n$ zeros, while the array $\mathbf{b}$ will contain m ones (at positions $b_1, \ldots, b_m$) and $b_m - m$ zeros.[31] In such a model, a shift $\Delta_i < 0$ is simply a deletion of Δ_i zeros from $\mathbf{a}$, while a shift $\Delta_i > 0$ is simply an insertion of Δ_i zeros in $\mathbf{a}$. With this model in mind, the Spectral Alignment problem is simply to find the edit distance between $\mathbf{a}$ and $\mathbf{b}$ when the elementary operations are deletions and insertions of blocks of zeros. As we saw in chapter 6, these operations can be modeled by long horizontal and vertical edges in a Manhattan-like graph. The only differences between the traditional Edit Distance problem and the Spectral Alignment problem are a somewhat unusual alphabet and the scoring of paths in the resulting graph. The analogy between the Edit Distance problem and the Spectral Alignment

31. We remark that this is not a particularly dense encoding of the spectrum.

problem leads us to frame spectral alignment as a type of longest path problem.

Define a *spectral product* $A \otimes B$ to be the $a_n \times b_m$ two-dimensional matrix with nm ones corresponding to all pairs of indices (a_i, b_j) and all remaining elements zero. The number of ones on the main diagonal of this matrix describes the shared peaks count between spectra A and B, or in other words, 0-similarity between A and B. Figure 8.27 shows the spectral products $S \otimes S'$ and $S \otimes S''$ for the example from the previous section. In both cases the number of ones on the main diagonal is the same, and $D(0) = 5$. The δ-shifted peaks count is the number of ones on the diagonal that is δ away from the main diagonal. The limitation of the spectral convolution is that it considers diagonals separately without combining them into feasible mutation scenarios. The *k-Similarity* between spectra is defined as the maximum number of ones on a path through the spectral matrix that uses at most $k + 1$ diagonals, and the *k-optimal spectral alignment* is defined as the path that uses these $k + 1$ diagonals. For example, 1-similarity is defined by the maximum number of ones on a path through this matrix that uses at most two diagonals. Figure 8.27 demonstrates the notion that 1-similarity shows that S is closer to S' than to S''; in the first case the optimal two-diagonal path covers ten 1s (left matrix), versus six in the second case (right matrix). Figure 8.28 illustrates that the spectral alignment detects more and more subtle similarities between spectra, simply by increasing k [compare figures 8.26 (c) and (d)].[32] Below we describe a dynamic programming algorithm for spectral alignment.

Let A_i and B_j be the i-prefix of A and j-prefix of B, respectively. Define $D_{ij}(k)$ as the k-similarity between A_i and B_j such that the last elements of A_i and B_j are matched. In other words, $D_{ij}(k)$ is the maximum number of ones on a path to (a_i, b_j) that uses at most $k + 1$ different diagonals. We say that (i', j') and (i, j) are *codiagonal* if $a_i - a_{i'} = b_j - b_{j'}$ and that $(i', j') < (i, j)$ if $i' < i$ and $j' < j$. To take care of the initial conditions, we introduce a fictitious element $(0, 0)$ with $D_{0,0}(k) = 0$ and assume that $(0, 0)$ is codiagonal with any other (i, j). The dynamic programming recurrence for $D_{ij}(k)$ is then

$$D_{ij}(k) = \max_{(i',j')<(i,j)} \begin{cases} D_{i'j'}(k) + 1, & \text{if } (i',j') \text{ and } (i,j) \text{ are codiagonal} \\ D_{i'j'}(k-1) + 1, & \text{otherwise.} \end{cases}$$

The k-similarity between A and B is given by $D(k) = \max_{ij} D_{ij}(k)$.

32. To a limit, of course. When k is too large, the spectral alignment is not very useful.

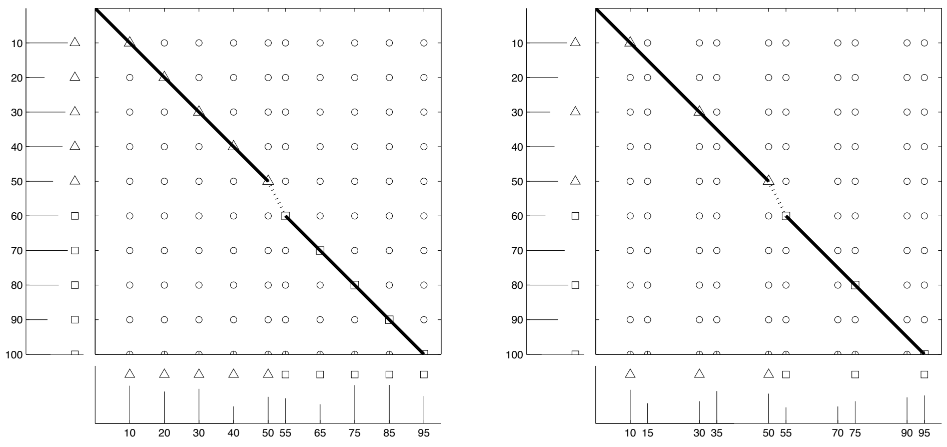


Figure 8.27 Spectral products $S \otimes S'$ (left) and $S \otimes S''$ (right), where $S = \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$, $S' = \{10, 20, 30, 40, 50, 55, 65, 75, 85, 95\}$, and $S'' = \{10, 15, 30, 35, 50, 55, 70, 75, 90, 95\}$. The matrices have dimensions 100×95, with ones shown by circles (zeros are too numerous to show). The spectrum S can be transformed into S' by a single shift and $D(1) = 10$. However, the spectrum S cannot be transformed into S'' by a single shift and $D(1) = 6$.

The above dynamic programming algorithm for spectral alignment is rather slow, with a running time of $O(n^4k)$ for two n-element spectra, and below we describe an $O(n^2k)$ algorithm for solving this problem. Define $diag(i,j)$ as the maximal codiagonal pair of (i,j) such that $diag(i,j) < (i,j)$. In other words, $diag(i,j)$ is the position of the previous "1" on the same diagonal as (a_i, b_j) or $(0,0)$ if such a position does not exist. Define

$$M_{ij}(k) = max_{(i',j') \leq (i,j)} D_{i'j'}(k).$$

Then the recurrence for $D_{ij}(k)$ can be re-written as

$$D_{ij}(k) = \max \begin{cases} D_{diag(i,j)}(k) + 1, \\ M_{i-1,j-1}(k-1) + 1. \end{cases}$$

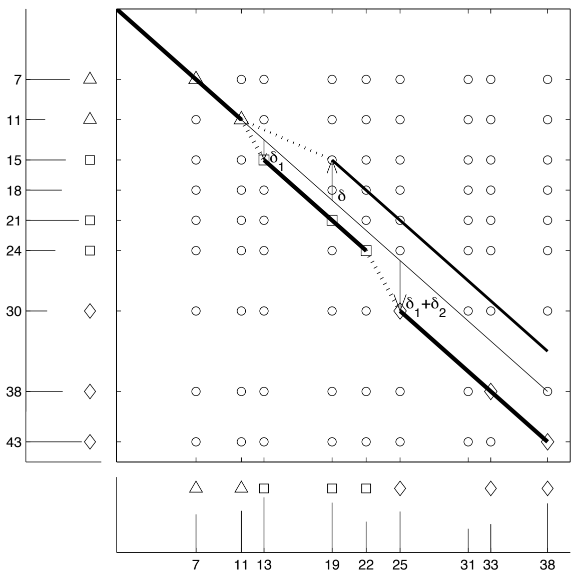


Figure 8.28 Aligning spectra. The shared peaks count reveals only $D(0) = 3$ matching peaks on the main diagonal, while spectral alignment reveals more hidden similarities between spectra ($D(1) = 5$ and $D(2) = 8$) and detects the corresponding mutations.

The recurrence for $M_{ij}(k)$ is given by

$$M_{ij}(k) = \max \begin{cases} D_{ij}(k), \\ M_{i-1,j}(k), \\ M_{i,j-1}(k). \end{cases}$$

The transformation of the dynamic programming graph can be achieved by introducing horizontal and vertical edges that provide the ability to switch between diagonals (fig. 8.29). The score of a path is the number of ones on this path, while k corresponds to the number of switches (number of used diagonals minus 1).

The simple dynamic programming algorithm outlined above hides many details that make the spectral alignment problem difficult. A spectrum can be thought of as a combination of *two* series of numbers, one increasing (the N-terminal ions) and the other decreasing (the C-terminal ions). These two

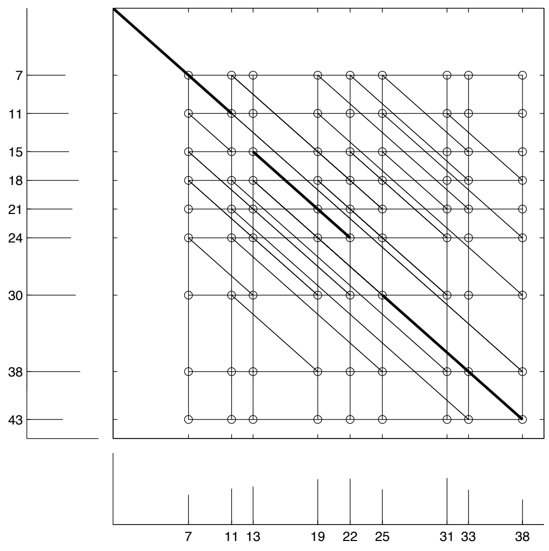


Figure 8.29 Modification of a dynamic programming graph leads to a fast spectral alignment algorithm.

series form diagonals in the spectral product $S \otimes S$, the main diagonal and the perpendicular diagonal. These correspond, respectively, to pairings of N-terminal and C-terminal ions. The algorithm we have described deals with the main diagonal only. Finding post-translationally modified proteins via mass spectrometry remains a difficult problem that nobody has yet solved, and significant efforts are underway to extend the Spectral Alignment algorithm to handle these complications and to develop new algorithmic ideas for protein identification.

8.16 Notes

The earliest paper on graph theory seems to be that of Leonhard Euler, who, in 1736, discussed whether or not it was possible to stroll around Königsberg crossing each of its bridges across the Pregel River exactly once. Euler remains one of the most prolific writers in mathematics: aside from graph theory, we owe him the notation $f(x)$ for a function, i for the square root of -1, and π for pi. He worked hard throughout his entire life only to become blind. He commented: “Now I will have fewer distractions,” and proceeded to write hundreds of papers more.

Graph theory was forgotten for a century, but was revived in the second half of the nineteenth century by prominent scientists such as Sir William Hamilton (who, among many other things, invented quaternions) and Gustav Kirchhoff (who is responsible for Kirchhoff’s laws).

DNA arrays were proposed simultaneously and independently in 1988 by Radoje Drmanac and colleagues in Yugoslavia (29), Andrey Mirzabenov and colleagues in Russia (69), and Ed Southern in the United Kingdom (100). The inventors of DNA arrays suggested using them for DNA sequencing, and the original name for this technology was sequencing by hybridization. A major breakthrough in DNA array technology was made by Steve Fodor and colleagues in 1991 (38) when they adapted photolithography (a process similar to computer chip manufacturing) to DNA synthesis. The Eulerian path approach to SBH was described in (83).

Sanger’s approach to protein sequencing influenced work on RNA sequencing. Before biologists figured out how to sequence DNA, they routinely sequenced RNA. The first RNA sequencing project resulted in seventy-seven ribonucleotides and took seven years to complete, though in 1965 RNA sequencing used the same “break—read the fragments—assemble” approach that is used for DNA sequencing today. For many years, DNA sequencing was done by first transcribing DNA to RNA and then sequencing the RNA.

DNA sequencing methods were invented independently and simultaneously in 1977 by Frederick Sanger and colleagues (91) and Walter Gilbert and colleagues (74). The overlap-layout-consensus approach to DNA sequencing was first outlined in 1984 (82) and further developed by John Kececioglu and Eugene Myers in 1995 (55). DNA sequencing progressed to handle the entire 1800 kb *H. influenzae* bacterium genome in the mid-1990s. In 1997, inspired by this breakthrough, James Weber and Eugene Myers (110) proposed the whole-genome shotgun approach (first outlined by Jared Roach and colleagues in 1995 (87)) to sequence the entire human genome. The human genome was sequenced in 2001 by J. Craig Venter and his team at Celera Genomics (104) with the whole genome shotgun approach, and independently by Eric Lander and his colleagues at the Human Genome Consortium (62) using the BAC-by-BAC approach.

Early approaches to protein sequencing by mass spectrometry were based on manual peptide reconstruction and the assembly of those peptides into protein sequences (51). The description of the spectrum graph approach presented in this chapter is from Vlado Dancik and colleagues (25). Searching a database for the purposes of protein identification in mass spectrometry was pioneered by Matthias Mann and John Yates in 1994 (71; 34). In 1995 Yates (112) extended his original SEQUEST algorithm to search for modified peptides based on a virtual database of all modified peptides. The spectral alignment algorithm was introduced five years later (84).