Chapter 1

Matroids

1.1 Introduction

Exercise 1[*] *Go to the index at the back of this guide and look up 'OR problem': you will find more than a dozen entries. Turn to each and spend a minute thinking about how you might solve it.*

The problems on which you have just reflected are representative of a subclass of those addressed by the models and techniques of Operations Research (OR). You will not have noticed the notorious Travelling Salesman Problem (TSP) among them, nor problems to do job shop scheduling nor timetabling (other than simple room allocation); such mountains in the optimisation world tend to be completely surmounted only by exhaustive search, at least with our current understanding, or must be circumvented using sophisticated approximation methods. The subclass of OR problems we have chosen, on the other hand, are relatively easy to solve. They are the foothills around a well-established base camp and it is this territory which we have chosen to explore in this study guide. It is a territory which may be more or less completely mapped out in terms of a beautiful and far-reaching branch of combinatorics called matroid theory.

The idea of a matroid (pronounced '*may*-troid') is largely due to a single American mathematician, Hassler Whitney, in the mid-1930s[†]. He belonged to the era in which modern abstract algebra was constructed by giants like Emil Artin, Garrett Birkhoff, Elie Cartan, Emmy Noether and Saunders MacLane; Birkhoff and MacLane also had a hand in inventing matroid theory, as did Noether's disciple Bartel Leendert van der Waerden. So matroids are primarily algebraic objects and as such have since become an established field of study for mathematicians, offering a generalising framework for questions in graph theory and finite geometry, coding theory, statics, electrical network theory and even, recently, mathematical physics.

In the 1960s matroids were identified as having a second role in capturing one of the simplest ways to solve an optimisation problem: via the so-called 'greedy' approach. It was found, notably by Jack Edmonds, who was then at the National Bureau of Standards and by Eugene Lawler at the University of Michigan that even non-greedy problems could often be dealt with efficiently in terms of matroid 'intersection'. It is remarkable just how many optimisation problems can be formulated and solved in these terms. The roots of matroid theory are firmly in linear algebra and geometry and these are also the foundation on which convexity theory is based. By the time we have fully explored matroid theory we shall find we already have the background we need to venture further.

We shall first meet matroids, then, as arising out of the idea of linear independence in vector spaces. We shall then start again, as it were, and arrive at matroids by way of certain optimisation problems on graphs (networks). In Chapter 3 we shall return to linear algebra when we discuss the intersection of matroids; indeed, we shall eventually restrict attention to matroids which are 'representable' as systems of vectors since this allows a particularly simple account of matroid intersection.

1.2 Matroids as algebraic objects

Modern algebra is based on proposing collections of abstract axioms and studying the properties of objects which satisfy these axioms. Our first approach to matroids will be in terms of axioms which generalise the

[*]Perhaps this is the first ever study guide to open with an exercise! Incidentally, this is the first and last exercise which does not have an answer in appendix C.

[†]In a striking example of simultaneous co-discovery, the foundations of matroid theory must be attributed also to Takeo Nakasawa, a young Japanese mathematician who joined the settlement of Manchuria in the 1930s and tragically lost his life after World War II, following the Russian annexation of the province. An account is given in *A Lost Mathematician, Takeo Nakasawa: The Forgotten Father of Matroid Theory*, editied by Nishimura, H. and Kuroda, S., see Appendix A: Further Reading.

idea of linear independence of vectors. It will not appear at first to be much related to optimisation. However, it is this route to matroid theory which introduces some basic linear algebra and motivates many of the terms and themes we will use later on.

1.2.1 Linear algebra

We shall be much concerned with $\mathbb{R}^n$, the n-dimensional vector space of real vectors. For now, we can safely regard this as just the collection of n-tuples of the form $(x_1, x_2, \ldots, x_n)$. Thus $(105, 70, -3.5)$ is a triple in $\mathbb{R}^3$, as might represent, say, distance, speed and acceleration; $(1.10, 2.72, 3.14, 0.00, 0.00)$ is a 5-tuple in $\mathbb{R}^5$; as a special case, (1.14) is a 1-tuple in $\mathbb{R}^1 = \mathbb{R}$ which would normally be written without the parentheses (), as 1.14.

Exercise 2 *Is $\mathbb{R}^3$ a subset of $\mathbb{R}^5$? If not why not? Is $\mathbb{R}$ a subset of $\mathbb{R}^5$?*

To say that $\mathbb{R}^n$ is a *vector space* means, among other things,

1. we have the *zero vector* $(0, 0, \ldots, 0)$, which we often just represent by 0;
2. we can add vectors together componentwise, for example, $(1, 2, 3) + (0, -3, 1) = (1 + 0, 2 - 3, 3 + 1) = (1, -1, 4)$;
3. we can multiply vectors by single numbers (referred to as *scalars*), for example, $-0.5 \times (1, -1, 4) = (-0.5 \times 1, -0.5 \times -1, -0.5 \times 4) = (-0.5, 0.5, -2)$; and
4. any vector may be obtained by adding together scalar multiples of the *n unit vectors*, the i-th unit vector having a 1 in position i and zeros everywhere else.

Exercise 3 *Calculate the following and say which vector space you are working in:*

(a) $(1, -2) + (0, 3)$;

(b) $-2 \times (1, 1, 1, 1, 1, 1) - 2 \times (1, -1, 0, -1, 1, 0)$;

(c) $3.142 - 2.718$.

(d) 3.142×2.718.

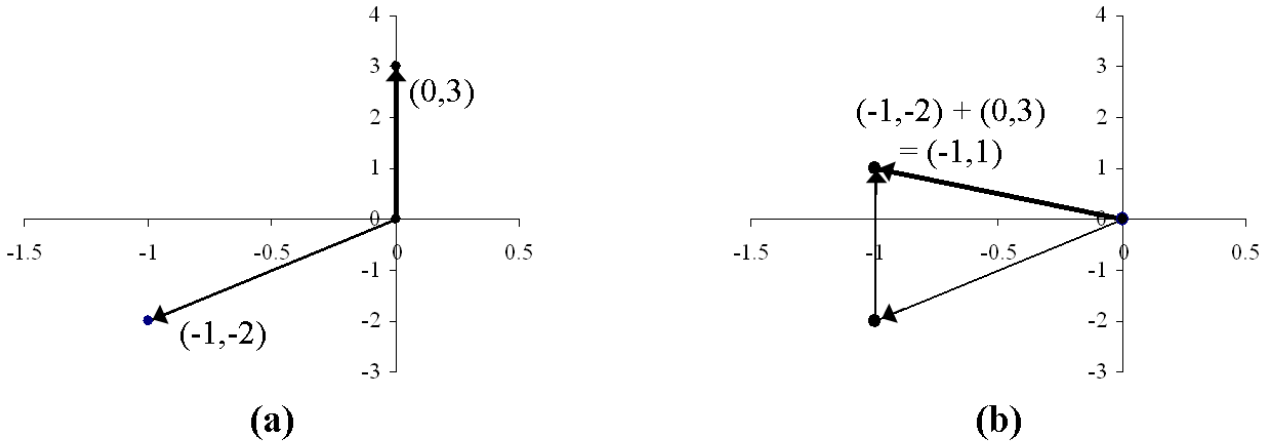


Figure 1.1: vector addition in two dimensions

It does no harm, in two dimensions, to think of a vector (x, y) as being an arrow pointing to the point (x, y) from the origin $(0, 0)$ as illustrated in Fig. 1.1(a). Then the scalar multiplication of (x, y) by λ just scales the length of the arrow by λ. Adding a vector (x', y') can be depicted as 'sliding' this vector along the line of the first vector, from the origin, while maintaining its angle and direction, and then 'completing the triangle', as shown in Fig. 1.1(b).

We will say that a vector v is a *linear combination* of vectors $u_1, \ldots, u_k$ if v can be expressed as a weighted sum of the u_i: $v = a_1u_1 + \ldots + a_ku_k$, the a_i being real numbers. This idea motivates a central concept in vector space theory, that of *linear independence*:

Definition 4 *A collection of vectors $v_1, v_2, \ldots, v_k$ is called* linearly dependent *if there are scalars $a_1, a_2, \ldots, a_k$,* not all zero, *such that $a_1v_1 + a_2v_2 + \ldots + a_kv_k = 0$ the zero vector. If no such scalars can be found then the vectors are called* linearly independent.

Note that the zero vector on its own forms a linearly dependent set.

Exercise 5 *Which of the following are linear combinations of* $(1, 1, 1)$ *and* $(-1, 0, 0)$*?*

(a) $22/7 \times (1, 1, 1) - \pi \times (-1, 0, 0)$*;*

(b) $0 \times (1, 1, 1) + 0 \times (-1, 0, 0)$*;*

(c) $4 \times (1, 1, 1)$*;*

(d) $4 \times (1, 1, 1) + 5 \times (-1, 0)$*;*

(e) $x^2 \times (1, 1, 1) - y^2 \times (-1, 0, 0)$*, for* $x, y \in \mathbb{R}$*;*

(f) $x \times (1, 1, 1)^2 + y \times (-1, 0, 0)^2$*, for* $x, y \in \mathbb{R}$*;*

(g) $\sqrt{x} \times (1, 1, 1) - \sqrt{y} \times (-1, 0, 0)$*, for* $x, y \in \mathbb{R}$*;*

Example 6 *Are the vectors* $(-1, -2)$ *and* $(0, 3)$ *linearly dependent in* $\mathbb{R}^2$*?*

Solution If v_1 and v_2 are any two vectors then they are linearly dependent if and only if one is a scalar multiple of the other, that is, they are *collinear*. This is not the case for $(-1, -2)$ and $(0, 3)$, as is clear from Fig. 1.1(a) (the arrows point in different directions). To give a formal proof based on definition 4: if $(-1, -2)$ and $(0, 3)$ were linearly dependent then we could find scalars a_1 and a_2, with $a_1(-1, -2) + a_2(0, 3) = (0, 0)$, with at least one of a_1 and a_2 non-zero. Assume that $a_2 \neq 0$. Then, rearranging,

$$(0, 3) = \lambda(1, 2), \quad \text{where } \lambda = \frac{a_1}{a_2}.$$

Comparing the two sides of this equation componentwise, we require

$$0 = \lambda \times 1, \quad 3 = \lambda \times 2,$$

which are incompatible. So no such a_1, a_2 can exist.

Exercise 7 *Show that assuming that* $a_1 \neq 0$ *in Example 6 also leads to an incompatibility and confirms that the vectors are linearly independent.*

Example 8 *Are the three vectors* $(1, -1)$ *and* $(0, 3)$ *and* $(2, -4)$ *linearly dependent?*

Solution This time, we seek a_1, a_2 and a_3 so that

$$a_1(1, -1) + a_2(0, 3) + a_3(2, -4) = 0.$$

If we arrange our three vectors in a 3×2 array or *matrix*, then we can answer the question schematically:

$$\begin{matrix} R1 \\ R2 \\ R3 \end{matrix} \begin{pmatrix} 1 & -1 \\ 0 & 3 \\ 2 & -4 \end{pmatrix} \overset{R3-2\times R1}{\longrightarrow} \begin{pmatrix} 1 & -1 \\ 0 & 3 \\ 0 & -2 \end{pmatrix} \overset{R3+\frac{2}{3}\times R2}{\longrightarrow} \begin{pmatrix} 1 & -1 \\ 0 & 3 \\ 0 & 0 \end{pmatrix}.$$

Here, the calculations over the $\longrightarrow$ symbols are so-called *elementary row operations*: we add to any row (vector) some linear combination of all the other rows (vectors). So $R3 - 2 \times R1$ means "add to Row 3 the combination given by $-2 \times$ Row 1 plus $0 \times$ Row 2". If elementary row operations can produce a zero vector then we have proved linear dependence. In fact, these elementary row operations provide the values

of a_1, a_2 and a_3 that we require: we multiplied Row 1 by -2 and Row 2 by 2/3 and added the results to 1× Row 3:

$$-2 \times (1, -1) + \frac{2}{3}(0, 3) + 1 \times (2, -4) = (0, 0).$$

Exercise 9 *Depict the linear combination in the solution of Example 8 graphically, as in Fig. 1.1(b).*

In Example 8 we used a procedure for showing linear dependence among vectors: we applied elementary row operations to set every entry below the first in column 1 to zero, and then to set every entry below the second in column 2 to zero. We did this by subtracting off multiples of the i-th row under the assumption that its own i-th entry was non-zero. This non-zero i-th entry in row i is often referred to as a *pivot element*. We can always reorder our rows — they are just vectors, which do not come in any particular order — so unless some column is entirely composed of zeros we can always arrange for the i-th entry in the corresponding i-th column to be non-zero. For example, in

$$\begin{matrix} \\ \text{Row 2} \\ \text{Row 3} \\ \\ \\ \end{matrix} \begin{pmatrix} a_{11} & a_{12} & \cdots \\ 0 & 0 & \\ 0 & a_{32} & \cdots \\ 0 & 0 & \\ 0 & a_{52} & \cdots \\ 0 & a_{62} & \cdots \end{pmatrix},$$

assuming that entry a_{32} is non-zero, we can interchange Row 2 and Row 3 to give Row 2 a non-zero pivot element. Now subtract suitable multiples of the new Row 2 from all rows below it: the first column will remain zero, the second column will now be zero from the 3rd row downwards. These ideas lead us to a very important definition and theorem:

Definition 10 *Any matrix whose elements satisfy the following two rules:*

1. *in each row, every element below the leftmost non-zero element is zero; and*
2. *every zero row has only zero rows below it*

is said to be in row-echelon form.

Theorem 11 (Gaussian Elimination) *Any matrix may be reduced to row-echelon form by elementary row operations, that is by repeatedly adding to any row some linear combination of all other rows.*

Exercise 12 *Reduce the matrix*

$$\begin{pmatrix} 2 & 3 & 5 \\ 7 & 11 & 13 \\ 17 & 19 & 23 \end{pmatrix}$$

to row-echelon form.

Example 8 illustrated two important applications of Theorem 11:

Corollary 13 *In $\mathbb{R}^n$, no linearly independent set of vectors can have cardinality greater than n.*

Corollary 14 *The number of linearly independent row vectors in a matrix X is equal to the number of non-zero rows when X is reduced to row echelon form. This number is called the* row rank *of X. The* transpose *of X, denoted X^T, is the matrix whose rows are the columns of X. The* column rank *of X is the row rank of X^T. For any matrix the row rank and column rank have the same value:*

$$\textit{row rank } X = \textit{ column rank } X = \textit{row rank } X^T,$$

and this number is called simply the rank *of the matrix X.*

Thus far, we have just been accumulating background information on linear independence. It is now time to see how this lead Whitney to the notion of a matroid.

1.2.2 Matroids defined by axioms

Let A be a finite subset of the vectors of $\mathbb{R}^n$. Let $\mathcal{I}$ be the collection whose members are all those subsets of A which consist of linearly independent vectors. Then the pair $M = (A, \mathcal{I})$ is an example of what we shall call a *matroid*. A is called the *ground set* of M. The members of $\mathcal{I}$ are called, naturally enough, the *independent sets* of M. A maximal member of $\mathcal{I}$, that is, a set $S \in \mathcal{I}$ such that $S \cup \{v\}$ is not in $\mathcal{I}$ for any v in A, is called a *basis* of M.

Example 15 *Let $A \subset \mathbb{R}^4$ be the set of vectors $\{(1,2,3,4),\ (1,-1,1,-1),\ (-2,2,-2,2),\ (6,0,10,4)\}$. Find the collection $\mathcal{I}$ for which $(A, \mathcal{I})$ is a matroid.*

Solution For convenience, we will refer to the four vectors as v_1, v_2, v_3, and v_4, respectively. We observe at once that v_2 and v_3 are collinear (a line in four dimensions!) so they can certainly not belong together in an independent set in $\mathcal{I}$. So $\{v_1, v_2, v_3\}$, for instance, cannot be in $\mathcal{I}$. We now apply elementary row operations to the set of vectors:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & -1 & 1 & -1 \\ -2 & 2 & -2 & 2 \\ 6 & 0 & 10 & 4 \end{pmatrix} \begin{matrix} \\ R3+2\times R2 \\ \longrightarrow \\ R4-6\times R2 \\ \longrightarrow \\ R2-R1 \\ \longrightarrow \end{matrix} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & -3 & -2 & -5 \\ 0 & 0 & 0 & 0 \\ 0 & 6 & 4 & 10 \end{pmatrix}.$$

Except for the first column we have not achieved row-echelon form: there is a zero row above some non-zero entries; even if we interchanged rows three and four there would still be a non-zero entry below the 2nd (pivot) entry in row 2. Nevertheless we have done enough for our analysis. Firstly, notice that Row 2 and Row 4 have become collinear. Does this mean that v_2 and v_4 cannot belong together in a set of $\mathcal{I}$? No! Because our elementary row operations used Row 1 to produce the new version of Row 2. It does mean, however, that $v_2,\ v_4$ *and* v_1 cannot appear together. Since v_2 and v_3 must also avoid each other, this means that $\{v_1, v_2\}$ is a basis. In fact, we have eliminated all possible subsets of A except:

$$\{\emptyset,\ \{v_1\},\ \{v_2\},\ \{v_3\},\ \{v_4\},\ \{v_1, v_2\},\ \{v_1, v_3\},\ \{v_1, v_4\},\ \{v_2, v_4\},\ \{v_3, v_4\},\ \{v_1, v_3, v_4\}\}.$$

Finally, we decide that the last of these, $\{v_1, v_3, v_4\}$, must also be removed: v_2 and v_3 are collinear, so if v_1, v_2 and v_4 are linearly dependent, so must v_1, v_3 and v_4 be. Thinking of it another way, we could have set Row 2 of our array to zero just as easily as Row 3.

In conclusion, the independent sets of our matroid are: all subsets of A of size at most two, except $\{v_2, v_3\}$.

Exercise 16 *Apply elementary row operations to the array*

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$$

to reduce two of the four rows to zero. What does this suggest about the matroid whose independent sets are linearly independent subsets of $A = \{(1,2,3,4),\ (5,6,7,8),\ (9,10,11,12),\ (13,14,15,16)\}$?

Example 15 and Exercise 16 give an indication of what is, for us, the key property of any matroid: *all its bases, that is its maximal independent sets, have the same size.* That is, for matroids,

maximal is the same as maximum.

Now we see the relevance to OR where our primary concern is to maximise things: in matroids, we just add things until we have reached maximality and cannot add anything more; then we know we have also maximised — we have added as much as it is possible to add.

Hassler Whitney's great insight was to realise that, while this key property can be proved for vectors in a vector space it could be asserted as a *defining* property for a collection of subsets of *any* set. This need not necessarily have been a useful way to generalise but we shall see that indeed it was.

That maximal = maximum in vector spaces is an old result of linear algebra. It has many equivalent formulations and we will use one which will be found to fit well with the idea of a matroid:

Theorem 17 (Steinitz Exchange Lemma) *Suppose that V is a set of vectors containing a linearly independent subset $S = \{v_1, \ldots, v_s\}$. Suppose that $T = \{u_1, \ldots, u_t\}$ is another linearly independent subset of V with $s < t$. Then there is some $u_i \in T$, $1 \leq i \leq t$, $u_i \notin S$, for which $S \cup \{u_i\}$ is linearly independent.*

Proof. We shall prove this for vector spaces over the field of real numbers $\mathbb{R}$, just to avoid having to think about other fields. Suppose that no such u_i existed. Then every $u_i \in T$ must be linearly dependent on the elements of S. But then we can write $T = \{a_{11}v_1 + \ldots + a_{1s}v_s, a_{21}v_1 + \ldots + a_{2s}v_s, \ldots, a_{t1}v_1 + \ldots + a_{ts}v_s\}$, with the $a_{ij} \in \mathbb{R}$. Now the a_{ij} may be taken to be the entries in t vectors: $(a_{11}, \ldots, a_{1s}), \ldots, (a_{t1}, \ldots, a_{ts})$ but since $s < t$ these vectors cannot be linearly independent, by Corollary 13. It follows that the u_i cannot be linearly independent either (see Exercise 19), which is a contradiction.

Corollary 18 *Every basis in a vector space, that is, every maximal linearly independent set, has the same size.*

Exercise 19 Linearity *of vector arithmetic is defined to mean that, for vectors v_1 and v_2 and scalars a and b,*

1. $a(v_1 + v_2) = av_1 + av_2$;
2. $a(bv_1) = (ab)v_1$.

Deduce that if $b_1(a_{11}, a_{12}) + b_2(a_{21}, a_{22}) + b_3(a_{31}, a_{32}) = 0$, for $a_{ij}, b_i \in \mathbb{R}, 1 \leq i \leq 3, 1 \leq j \leq 2$, with not all the b_i being zero, then the vectors $a_{11}v_1 + a_{12}v_2$, $a_{21}v_1 + a_{22}v_2$ and $a_{31}v_1 + a_{32}v_2$ are not linearly independent, for any choice of vectors v_1 and v_2.

Exercise 20 *Prove Corollary 18.*

Whitney turned Theorem 17, together with two intuitive facts about linear independence into *axioms* for a mathematical structure:

Definition 21 *A* matroid *is any collection of subsets $\mathcal{I}$, the* independent sets*, of a finite set A (the* ground set*) satisfying the following axioms:*

1. *the empty set $\emptyset$ is independent;*
2. *the collection of independent sets is subset-closed: if X is an independent set and $Y \subseteq X$ then Y is also an independent set; and*
3. *(the exchange — or sometimes the 'augmentation' — axiom) if X and Y are independent with $|X| < |Y|$ then there is some element $y \in Y$ such that y is not in X and $X \cup \{y\}$ is independent.*

(Note that (1) follows from (2) automatically but is usually given emphasis as a separate axiom).

Example 22 *Suppose that $A = \{red, blue, green, purple\}$ and that each colour on its own constitutes an independent set of a matroid M defined on A. Suppose that M has eight independent sets in all. Use the matroid axioms to determine the structure of M.*

Solution The empty set $\emptyset$ must always be independent, so we actually have only three other independent sets to find. We can reason as follows:

1. Suppose some independent set contains three elements. It has three subsets of 2 elements which, by axiom (2) of definition 21, are also independent, giving a total of nine independent sets. This is too many, so no independent set has three elements and we must find three 2-element independent sets.
2. Now suppose two of these are disjoint; say, {red, blue} and {green, purple} are independent. By applying definition 21, axiom (3) to the pair $X =$ {green} and $Y =$ {red, blue} we see that either {red, green} or {blue, green} must also be independent. But if we now choose $X =$ {purple} the same argument makes one of {red, purple} and {blue, purple} independent. This gives four 2-element sets in all which is too many. So no pair of independent sets of size two can be disjoint.

3. Suppose, then, that two of the independent sets are {red, blue} and {red, green}. Apply definition 21, axiom (3) to the pair X = {purple} and Y = {red, blue}: we must have {purple, red} independent, since {purple, blue} would be disjoint from {red, green}.

We conclude that the structure of the matroid is that its independent sets are $\emptyset$, all singleton sets and three 2-elements independent sets which intersect in a common element and whose union is A.

Notice that we did not completely determine the matroid in this example: there were four different possible matroids depending on which colour was found in all the size 2 independent sets. However, we had discovered the *structure* of the matroid in the sense that all four possibilities are identical except for a relabelling of the ground set. Two matroids whose independent sets become identical under a relabelling of the ground set are called *isomorphic*.

Exercise 23 *List the eight independent sets of two* different *matroids which would fit the specification of Example 22 and specify the relabelling which shows they are isomorphic.*

Exercise 24 *Let $A = \{1, 2, 4, 6\}$ be the ground set of a matroid whose independent sets are precisely* the subsets of A whose elements are mutually coprime, that is have greatest common divisor 1. List the independent sets.*

Corollary 18 of Theorem 17 now reappears as a fundamental theorem of matroid theory:

Theorem 25 *If $M = (A, \mathcal{I})$ is a matroid then every maximal member of $\mathcal{I}$ has the same size.*

Given a subset X of the ground set A of a matroid M, by analogy with Corollary 14, the size of a maximal independent set contained in X is called the *rank* of X, written rank(X). It is well-defined by Theorem 25. The rank of A itself is then the size of any basis for the matroid and is called the rank of the matroid, rank(M).

Exercise 26 *Define a matroid M whose ground set A is the set of columns of the matrix*

$$\begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & -1 & 1 & -1 & 1 \end{pmatrix} \end{array}$$

and whose independent sets are the linearly independent subsets of columns, including the empty set. What is rank(M)*? What are the ranks of the subsets of A (specified by column numbers)* $\{1, 2, 3\}$, $\{2, 3\}$ *and* $\{4, 5\}$ *and which of these subsets is a basis for M?*

Note an important subtlety in the definition of the matroid in Exercise 26: the ground set A had six elements but two, columns 4 and 6, were identical as vectors. It is best to think always of A as a set of possibly identical but *differently labelled* objects. We shall meet this subtlety again almost at once in the next section.

1.3 Matroids and the greedy algorithm

We will appear to completely change direction now, looking at a classical optimisation problem: that of finding a minimum-weight spanning tree in a graph or network. It will be seen that, in fact, constructing optimal solutions which are of maximal size can be done so that all such solutions are guaranteed also to have *maximum* size. Problems for which this is possible are said to be solved 'greedily'; in fact, you have met this idea before in the level 2 unit *Software Engineering, Algorithm Design and Analysis*, in connection with Huffman coding. We shall see that the optimal solutions found by the greedy approach can, for many problems, be identified with the minimum-weight bases of an associated matroid. Before we return to matroids by this route we again need some background theory.

*The word 'precisely' in mathematics usually means "these and no others" or "then and only then", as in "the prime numbers less than 10 are precisely 2, 3, 5 and 7," or "a prime number is even precisely if it is two."

1.3.1 Graph theory

The unit *Software Engineering, Algorithm Design and Analysis* dealt with graph theory from the point of view of abstract data types and graphs were also covered in the level 1 unit *Mathematics for Computing*, so we will be rather informal and recall the main definitions using an example. Consider Figure 1.2.

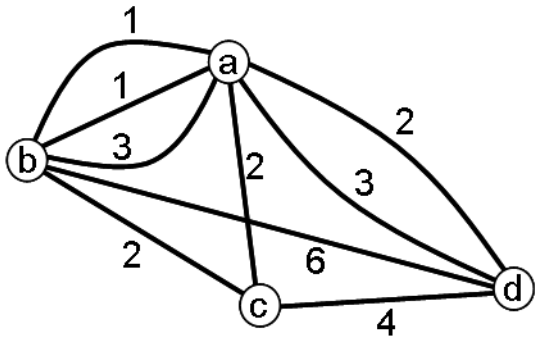


Figure 1.2: a graph on four vertices.

In the language of *graph theory* we refer to this object as a *graph*, consisting of four points called *vertices* and nine point-to-point connections called *edges*. Vertices thus connected are said to be *adjacent* to each other and to be *incident* with the edges connecting them. The letters on the vertices in Figure 1.2 are labels used for ease of reference; the numbers on the edges are *weights* over which we will want to optimise in some way. In general, we will denote the weight of edge e by $w(e)$, with w being some function from the edge set to the real numbers $\mathbb{R}$. We can refer to edges in terms of the vertices which they interconnect but we must be careful when we have *multiple edges*: in Figure 1.2, $ab = ba$ denotes a set of three edges joining a and b, so $w(ab)$ is not well-defined; $w(ac) = w(ca)$ is unambiguous and has value 2. (Remember the subtlety in Exercise 26 — here we have a set of edges which are all different but among which some appear identical when represented by their end vertices.)

It is essential to realise that only the edge weights and interconnections are of interest to us: the actual way in which they are drawn in Figure 1.2 is not important. So the following table is an equally valid and informative representation. A drawing is often a helpful visualisation, however.

b	**c**	**d**	
1,1,3	2	2,3	**a**
	2	6	**b**
		4	**c**

Graphs can be used to formulate and solve a great variety of problems, many of which are to do with connectivity. A *walk* in a graph is any sequence of edges $u_1v_1, \ldots, u_tv_t$ which are *consecutive* in the sense that $u_{i+1} = v_i$, for $i = 1, \ldots, t-1$; this walk has length t and can be represented (modulo choice of multiple edges) by the vertex sequence $u_1, u_2, \ldots, u_t, v_t$; if all the u_i are distinct, the walk is called a *path*. The graph is *connected* if there is a walk connecting any two vertices. The graph of Figure 1.2 is connected; in fact, every vertex can be reached from any other via a walk of length 1 — a single edge. Of course, any vertex v is connected to itself by the *trivial walk*[*] having zero edges. A walk of any length from v to itself is said to be *closed*; if it repeats no edges and no vertices other than v it is a *cycle* (so a cycle is a closed path). A cycle of length 1, which is an edge joining a vertex to itself, is called a *self-loop*.

Example 27 *How many closed walks of length 2 begin and end at vertex a in Figure 1.2? How many of these are cycles?*

[*]The word ‘trivial’ in mathematics means ‘empty’ or ‘of size zero’. It may also mean ‘self-evident’ as in the anecdote of the mathematics professor who says to his class “The proof of what I have just written on the blackboard is trivial.” He then stares at what he has written with a puzzled frown. Then he begins pacing up and down, knitting his brows in thought. Finally after a quarter of an hour his face clears: “Yes!” he says, “it *is* trivial!” And he placidly continues his lecture.

Solution There are 14 closed walks of length 2 from a back to itself: going via b gives three choices of edge out and three choices back again, giving a total of nine walks; similarly going via d gives four walks and via c, one walk. In a cycle no edge may be repeated, so there is no cycle via vertex c and only $3 \times 2 + 2 \times 1 = 8$ via b and d.

We mentioned that we would be looking for something called a minimum-weight spanning tree. A *tree* is a graph which has no non-trivial cycles* and which is connected; if it fails to be connected then it must consist of a number of trees, and is accordingly referred to as a *forest*. Given a graph we may look for a collection of edges and vertices (a subgraph) which together form a tree; if all vertices are included then it is said to be *spanning*. We will be trying to minimise the weight of such a tree, the weight being the sum of its edge weights.

Let us summarise the main definitions:

Definition 28 *A graph $G = (V, E)$ with vertex set V and edge set E is*

1. *said to be* connected *if any vertex in V may reach any other vertex along some* walk, *i.e., sequence of consecutive edges in E;*
2. *called a* forest *if it does not contain any non-trivial cycles;*
3. *called a* tree *if it is a connected forest.*

A collection H of weighted edges in a graph G, together with a subset V' of the vertex set V of G is

4. *defined to have weight equal to the sum of the weights of the edges in H;*
5. *called a* subgraph *of G if no edge of G appears more than once in H, and V' contains all vertices incident with edges in H;*
6. *called a* spanning subgraph *of G if it is a subgraph and contains every vertex of G;*
7. *called a* spanning tree *of G if it is a spanning subgraph which is a tree.*

Exercise 29 *In the graph of Figure 1.2 list all closed walks and cycles beginning and ending at vertex c which have length at most 3 and give their weights.*

Exercise 30 *In the graph of Figure 1.2 find all spanning forests of weight 4 which are not spanning trees.*

1.3.2 Minimum-weight spanning trees

We can be precise now about what is meant by a minimum-weight spanning tree for a graph G: it is a collection of distinct edges of G which form a spanning tree and whose weights have the smallest possible sum over all such collections. We are going to define a matroid whose ground set is the edge set of G and whose independent sets are the forests in G. If we 'greedily' add the cheapest possible edges while building such independent sets then we will find that we always arrive at a basis — a spanning tree. Now instead of the axiomatic view of matroids we have an OR view: matroids are systems of subsets over a weighted set such that bases of minimum weight may be constructed greedily.

Before we specify the greedy algorithm and our matroid formally we will take some time to motivate the minimum-weight spanning tree problem from an OR standpoint, and we shall also digress to introduce another OR problem, maximum matching, which *cannot* be solved greedily.

Firstly, here are two problems which might lead an operations researcher to want to find a minimum-weight spanning tree:

Peer-to-peer Updating: a number of nodes in a network need to update each other regularly. A set of connections is to be nominated by which this may be done as cheaply as possible;

*The 'trivial' cycle, in line with the footnote on the previous page, is one which has zero edges—it just 'stays where it is'.

Justified Product Elimination: a set of products is to be given a partial ranking via a number of comparisons, each involving a pair of products, as cheaply as possible, in order to justify the elimination of some 'worst' product.

These are rather vague problem specifications! They might be approached in countless ways and it is certainly one of the hardest parts of the operations researcher's job to recast them as well-defined, mathematical problems whose solution will be both feasible and adequate for the needs of the client or end-user. In this case the mathematical reformulation will be:

MIN-WEIGHT SPANNING TREE

Input:	an edge-weighted graph the edge weights being in $\mathbb{R}$
Output:	a minimum-weight spanning tree

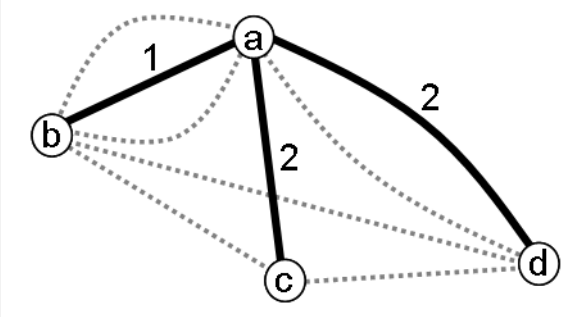
This grid will serve as a pattern for us in this study guide and for the most part it will be our starting point, since our focus is on mathematical techniques. In the present case, however, we will discuss in some detail how it comes about.

Let us take **Peer-to-peer Updating** first. Suppose the graph we met in Figure 1.2 in the previous subsection models the network, with the edge weights being the cost of individual connections (perhaps as offered by network service providers). We need a set B of edges along which any vertex may update any other, with the sum of the edge weights being as small as possible. So B must be a spanning, connected subgraph. Moreover, B should contain no cycles, for if $u_1u_2, u_2u_3, u_3u_4, \ldots, u_tu_1$ were such a cycle then we could remove, say, u_2u_3, breaking the cycle but maintaining the connection between u_2 and u_3 via the path $u_2u_1u_tu_{t-1}\ldots u_4u_3$. Then $B\backslash\{u_2u_3\}$ is a cheaper spanning connected subgraph, contradicting the assertion that B had minimum weight.

Now any spanning tree in an edge-weighted graph is a candidate solution for peer-to-peer updating in that graph. Optimal solutions are precisely those spanning trees whose edge weights sum to the minimum possible.

Example 31 *Solve the peer-to-peer updating problem for the graph of Figure 1.2.*

Solution We proceed in the most natural way, by first choosing one of the *ab* edges of weight 1 — clearly it would be counterproductive to choose the weight 3 edge — and then adding in vertex *c* using a weight 2 edge, either choice looking equally suitable. We remark at this point that adding *both* edges *ac* and *bc* would create a cycle *abc*; while this might increase the *reliability* of our updating network, our only concern is to minimise *cost*: any two edges of this cycle are sufficient to connect *a*, *b* and *c*, and we have already achieved this with a cost of 1 + 2 = 3. Finally, we add vertex *d* to our subset of edges using edge *ad*, increasing the total cost to 5:



Exercise 32 *There are four solutions to the peer-to-peer updating problem for Figure 1.2; find the other three.*

Exercise 33 *Find a minimum weight spanning tree in the graph shown in Figure 1.3.*

Let us now turn to the other problem: **justified product elimination**. This time the problem does not directly involve a graph but we can construct one: we will use a vertex to represent each product and an edge joining vertices *u* and *v* will represent a comparison between products *u* and *v*, with the weight of the

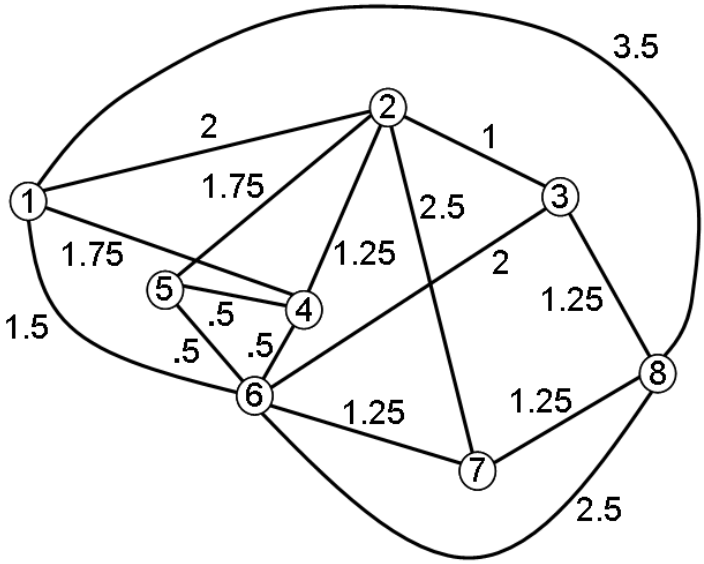


Figure 1.3: graph for Exercise 33.

edge being the cost of making the comparison (this might be the cost of a proposed customer survey, for example). We shall see that a minimum weight spanning tree is one way of ranking our products. Suppose Figure 1.2 now represents our comparison costings. We will compare a and b at cost 1; as before, a natural continuation is to compare c to either a or b at cost 2. Suppose we compare a to c. Figure 1.4 now shows two possible outcomes. Next to each subgraph is a diagram showing the ranking of a, b and c, so far as we

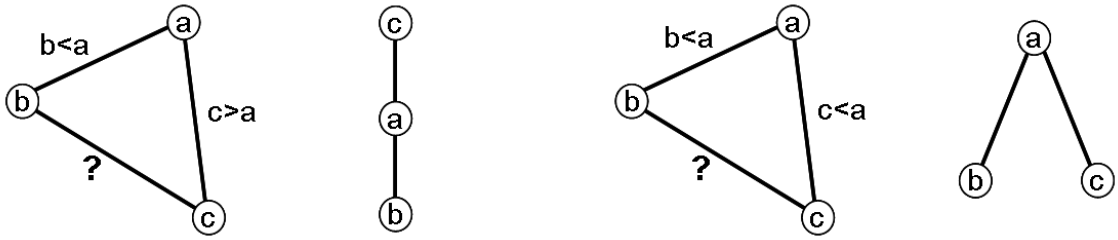


Figure 1.4: comparisons in the graph of Figure 1.2, with corresponding partial orders.

have discovered, the higher ranked products being placed above the lower ranked ones. The diagram on the left suggests that $b < c$ but we do not know this because we have not made the comparison. And we do not want to! We have been asked to rank the products, at minimal cost, to enable elimination. If we created a cycle by comparing b to c we would either (1) confirm that $b < c$ but at extra cost, or (2) get the result $c < b$, contradicting the previous ranking and undermining our justification for elimination of b. (Of course, this latter result suggests that further expenditure on product comparison might be important but you will rarely be thanked for giving your client this advice!)

The ranking on the right of Figure 1.4 indicates that we might eliminate either b or c (we have not justified choosing either one over the other). It is usually legitimate in management terms to toss a coin to make the final elimination of just one product.

Exercise 34 *For each of your solutions to Exercise 32, add inequalities to each spanning tree edge (as in Figure 1.4) which (a) avoid the necessity of coin tossing, and (b) require coin tossing, in order to make a final choice of product to eliminate. Suggest another approach to solving the justified product elimination problem.*

1.3.3 Minimum-cost matching

We have now gained a little experience in solving things ‘greedily’: we have been building our spanning trees guided only by the cost of the edges. We are going to describe the greedy algorithm more formally in subsection 1.3.4; before that it seems appropriate to introduce a problem which *cannot* be solved greedily and which will be of interest later.

In Oliver Stone’s 1987 film *Wall Street* the financier Gordon Gekko famously declares “Greed is right. Greed works.” For optimisation problems in general, the greedy approach is *not* sound. An alternative to the minimum spanning tree approach suggested in the solution to Exercise 34 was to compare products in pairs at minimum cost and then eliminate one of the losers at random. Let us try to apply this approach greedily to the graph of Figure 1.3. We might reasonably start by pairing vertices 4 and 6; the next cheapest pairing is then vertices 2 and 3 and then 7 with 8. This gives us the three edges shown in Figure 1.5(a).

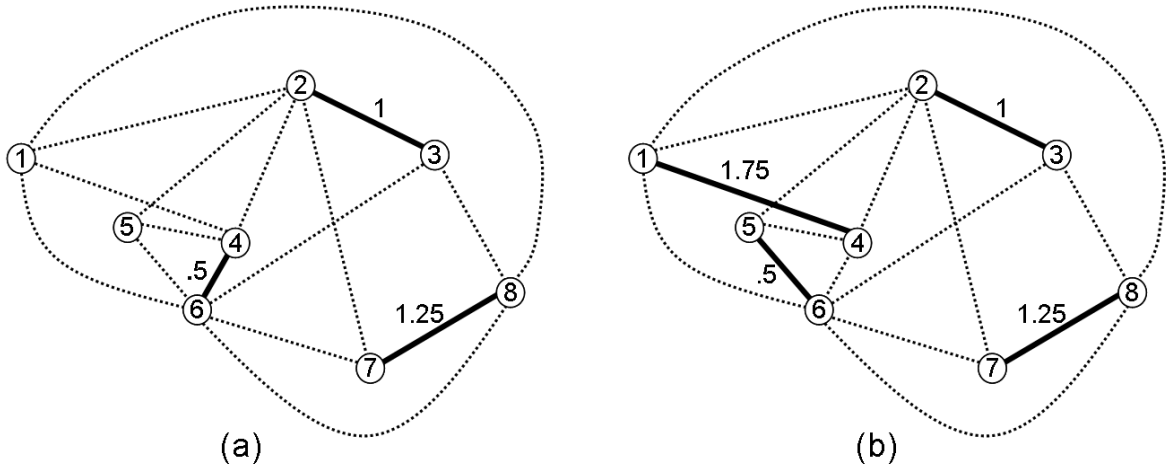


Figure 1.5: two matchings in the graph of Figure 1.3.

A collection of edges in a graph, such as the one highlighted in the figure, with no two edges sharing any vertex is called a *matching*. The matching we have constructed (greedily) is *maximal* because, even though vertices 1 and 5 are unpaired, no edge joins them (in our product elimination scenario, no comparison of these products has been proposed). But it is not *maximum*; indeed, in Figure 1.5(b) we see that, by choosing to pair vertices 5 and 6 first, instead of 4 and 6, we can successfully pair up all the vertices. Such a matching is said to be *perfect*.

We suggested earlier that the link between the greedy approach and matroids was even stronger: matroids are subset systems in which you can minimise cost while successfully constructing a maximum independent set. Although Figure 1.5(b) offers a perfect matching, it fails this minimisation criterion. Pairing vertices 4 and 5 first leads to a perfect matching of lower cost than Figure 1.5(b), as displayed in Figure 1.6.

So for the minimum-weight maximal matching problem the greedy approach fails completely: it fails to guarantee a maximal matching and if, by luck, one is found it fails to guarantee it will be minimum weight. Let us nevertheless formally identify this problem for later reference:

MIN-WEIGHT MAXIMAL MATCHING

Input:	graph G with edge set E, function $w : E \to \mathbb{R}$
Output:	a maximal matching of G of minimum weight

We shall see in Chapter 3 that matroids can help with this problem to some extent: the idea of a ‘matroid intersection’ can at least find maximum matchings in a restricted case: that of *bipartite* graphs, those which have no odd-length cycles. So for bipartite graphs this will at least avoid the maximal≠maximum failure of Figure 1.5(a), even though it will not avoid the non-minimum-cost failure of Figure 1.5(b). At

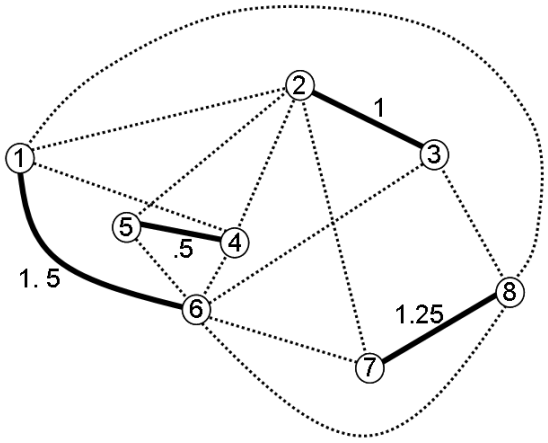


Figure 1.6: a minimum-cost perfect matching in the graph of Figure 1.3.

the end of Chapter 2 we shall even see that the greedy algorithm does 'solve' the general matching problem, but in a way which seems to highlight the limitations of the greedy approach!

Exercise 35 *Show that the greedy approach will give a perfect matching in the graph of Figure 1.2 but will fail to give a minimum-cost perfect matching.*

Before moving on, we will summarise, a little more formally, the new ideas from graph theory which we have just met.

Definition 36 *In a graph G, the* neighbourhood set *of a vertex v is the subset of those vertices, excluding v, which are adjacent to v. The* degree *of v is the cardinality of its neighbourhood set + twice the number of self-loops at v. A subgraph of G in which every vertex has degree 1 consists of* independent *edges and is called a* matching. *A spanning matching is called* perfect.

The following exercises use the graph of Figure 1.3 but without the edge weights. We reproduce the unweighted version in Figure 1.7 for convenience.

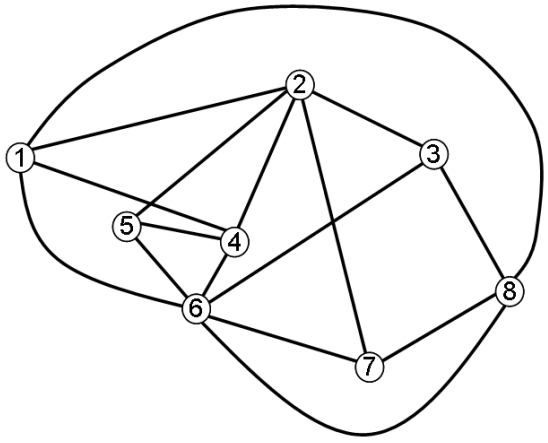


Figure 1.7: unweighted version of Figure 1.3.

Exercise 37 *The* **Hand-shaking Lemma** *asserts that in any graph G there is an even number of vertices of odd degree. Confirm that this holds for the graph of Figure 1.7. If an additional edge is added between vertices 3 and 7, what happens to the number of vertices of odd degree?*

Exercise 38 Euler's Theorem* *asserts that in any graph G there is a walk which traverses every edge exactly once (an* Euler trail*) if and only there are at most two vertices of odd degree. Convince yourself that no such walk exists in the graph of Figure 1.7 but confirm that such a walk can be found if an additional edge is added between vertices 3 and 7. (Hint: begin your walk at a vertex of odd degree).*

Exercise 39 *(Harder) Suppose a graph G has n vertices and its vertex degrees, in ascending order, are* $d_1, d_2, \ldots, d_n$. **Chvatál's Theorem** *asserts that if, for each* $i < n/2$, *either* $d_i > i$ *or* $d_{n-i} \geq n - i$, *then G has a* Hamilton cycle*: a cycle which passes each vertex exactly once. Show that:*

1. *the degrees of the graph of Figure 1.7 fail to satisfy this condition;*
2. *the condition becomes satisfied if an additional edge is added between vertices 3 and 7;*
3. *the condition is sufficient but not necessary since, in fact, a Hamilton cycle can be found in the graph of Figure 1.7; and hence*
4. *the graph has two edge-disjoint perfect matchings (i.e. two disjoint sets of edges each of which is a perfect matching).*

Exercise 40 *(Tricky!) Show that the graph of Figure 1.7 does not have* three *edge-disjoint matchings.*

1.3.4 The greedy algorithm

We have just seen several problems whose solutions seem unavoidably to require *backtracking*. We get so far towards constructing an Euler trail or a Hamilton cycle or a perfect matching and then we are blocked. For instance, Figure 1.8 shows a well-known little 'brain-teaser': draw the envelope without taking your pen off the paper and without drawing any part twice. Having gained experience with Euler's Theorem

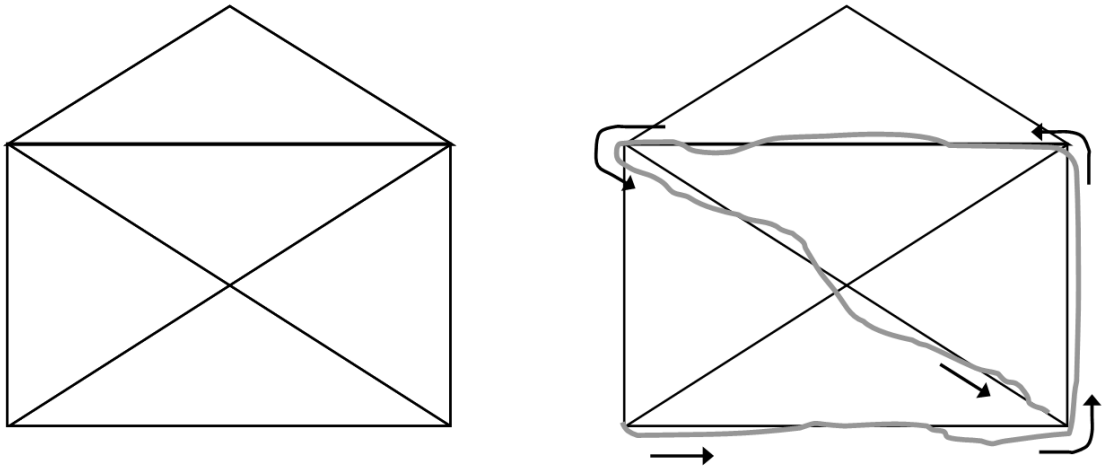


Figure 1.8: find an Euler trail in this 'graph'.

(Exercise 38) we know to begin and end our drawing at the two 'vertices' of degree 3 (the bottom corners of the envelope). But we still cannot draw 'greedily': the route taken on the right of Figure 1.8 fails to complete the drawing since we are blocked at the bottom right-hand corner; we must backtrack and choose a different direction from the top left-hand corner.

Constructions which can be represented in terms of the independent sets of some matroid, on the other hand, can always proceed without backtracking. What is more, we can optimise as we go. Let us make this formal. Here is the general problem:

*Named after the Swiss mathematician Leonhard Euler (1707–1783); his name is pronouced '**Oil**-er'.

MIN-WEIGHT MAXIMAL SUBSET

Input:	finite set A, function $w : A \to \mathbb{R}$, collection of subsets $\mathcal{A}$ of A
Output:	a maximal subset belonging to $\mathcal{A}$ of minimum weight

Here is the algorithm we wish to apply to it*:

Algorithm 41 (The Greedy Algorithm)

INPUT: instance of MIN-WEIGHT MAXIMAL SUBSET problem
OUTPUT: optimal solution for this problem instance

1 X:=A; Y:=$\emptyset$;
2 **while** some $x \in X$ has $Y \cup \{x\} \in \mathcal{A}$ **do**
3 choose such an x of least weight;
4 X:=$X \backslash \{x\}$;
5 Y:=$Y \cup \{x\}$
6 **od**
7 **return** Y

And here are the conditions under which it works:

Theorem 42 *Suppose that the collection of subsets $\mathcal{A}$ of A satisfies the first two matroid axioms of definition 21, i.e. $\mathcal{A}$ is subset-closed: $X \in \mathcal{A}$ and $X' \subset X$ implies $X' \in \mathcal{A}$. Then Algorithm 41 correctly solves MIN-WEIGHT MAXIMAL SUBSET for all choices of weight function w if and only if $\mathcal{A}$ is the collection of independent sets of a matroid, i.e. $\mathcal{A}$ also satisfies the exchange axiom of definition 21.*

Exercise 43 *Let $A = \{a, b, c, d\}$ and let $\mathcal{A}$ be a collection of subsets $\{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{c, d\}\}$. Find a weight function for A which causes Algorithm 41 to fail on $\mathcal{A}$. Deduce that $\mathcal{A}$ is not a matroid and explain how the matroid axioms (definition 21) are violated by $\mathcal{A}$.*

The Greedy Algorithm is usually formulated in terms of a seemingly simpler problem:

MAXIMUM-WEIGHT SUBSET

Input:	finite set A, function $w : A \to \mathbb{R}$, collection of subsets $\mathcal{A}$ of A
Output:	a subset belonging to $\mathcal{A}$ of maximum weight

This is actually equivalent to the MIN-WEIGHT MAXIMAL SUBSET. Maximising and minimising amount to the same thing since, if A is a set of weighted objects then the maximum-weight subset among a collection of subsets of A is the same as the minimum-weight subset when, for each x in A, $w(x)$ is replaced by $-w(x)$. And as long as we are dealing with matroids, maximal is the same as maximum size, so that a subset of maximum weight will also be maximal and vice versa.

The remainder of this subsection will be spent on defining two classes of matroids and showing how the greedy algorithm applies to them. Firstly, we revisit the familiar example of the spanning tree.

Definition 44 *Let G be a graph with edge set E. Let $\mathcal{A}$ be the collection containing precisely those subsets of E which contain no cycles. Then the pair $(E, \mathcal{A})$ forms a matroid, called the* cycle matroid *of G, denoted $M(G)$.*

The name *cycle matroid* derives historically from the definition of this matroid in terms of *dependent sets*: those which contain cycles. We remark that, although a graph G will usually be defined in terms of its vertices and edges, the cycle matroid $M(G)$ makes no reference to vertices. The cycle matroid cannot, therefore, completely specify a graph and, indeed, two different graphs may have the same cycle matroid; see Figure 1.9. An important implication of this is that the cycle matroid is often less helpful with optimisation problems in which vertices play an essential role.

Exercise 45 *Suppose a graph G has vertex set $V = \{x, y, z\}$ and edge set $E = \{xy, xz, yz\}$. Write down the collection $\mathcal{I}$ of independent sets of the cycle matroid $M(G)$.*

*Algorithms are presented in a pseudocode which it is hoped will be self-explanatory. In particular, notice that the scope of an **if** statement is delimited by a matching **fi** while **do** loops end with a matching **od** (the convention used in MAPLE, for example).

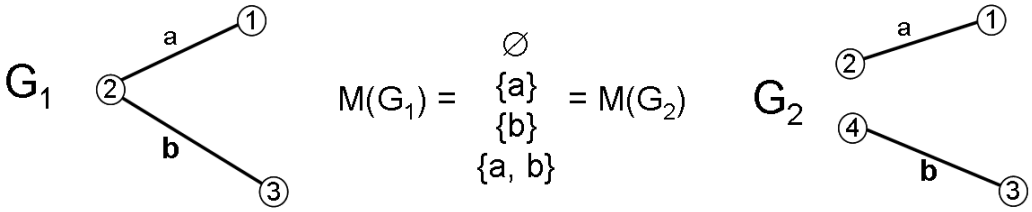


Figure 1.9: two different graphs having the same cycle matroid.

Example 46 *Let $A = \{a, b, c, d, e\}$. Let a weight function w be defined by $w(a) = 3,\ w(b) = 2,\ w(c) = 1,\ w(d) = 4,\ w(e) = 5$. Apply the greedy algorithm to the collection $\mathcal{A}$ which consists of all subsets of A except $\{a, b\}$ and $\{c, d, e\}$.*

Solution Algorithm 41 proceeds as follows:

$X = \{a, b, c, d, e\},\ Y = \emptyset$
neither $\{a, b\}$ nor $\{c, d, e\}$ is contained in $Y \cup \{c\} = \{c\}$ (so $\{c\}$ is independent)
$X = \{a, b, d, e\},\ Y = \{c\}$, total weight=1
neither $\{a, b\}$ nor $\{c, d, e\}$ is contained in $Y \cup \{b\}$
$X = \{a, d, e\},\ Y = \{b, c\}$, total weight=3
neither $\{a, b\}$ nor $\{c, d, e\}$ is contained in $Y \cup \{d\}$
$X = \{a, e\},\ Y = \{b, c, d\}$, total weight=7
either $\{a, b\}$ or $\{c, d, e\}$ is contained in $Y \cup \{x\}$ for all $x \in \{a, e\}$
Output: $\{b, c, d\}$ with weight 7

Although Example 46 was presented purely in terms of a collection of subsets, it was a minimum-weight spanning tree problem in disguise:

Exercise 47 *Show how the application of algorithm 41 in Example 46 corresponds to selecting a sequence of edges in the graph of Figure 1.10.*

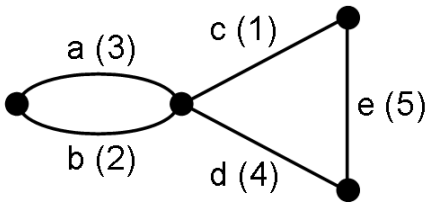


Figure 1.10: graph for Exercise 47 (edge weights in parenthesis).

Matroids, such as the one in Example 46, which are isomorphic to the cycle matroid of some graph are called *graphic*.

Exercise 48 *In Exercise 24 in subsection 1.2.2 we defined a matroid on the ground set $A = \{1, 2, 4, 6\}$ by declaring any subset of coprime elements of A to be independent. Show that this matroid is graphic.*

Exercise 49 *Let $A = \{a, b, c\}$ be the ground set of a matroid M whose only independent sets are $\emptyset$ and $\{a\}$. Find two different graphs whose cycle matroids are isomorphic to M. (Hint: use self-loops — edges joining a vertex to itself).*

We shall now see that by choosing a different type of matroid the greedy algorithm can partially solve a very different optimisation problem. Let us do what we did in subsection 1.3.2 and motivate this using two management problems which might confront the OR professional:

Job Assignment: a number of jobs $J_1, \ldots, J_m$ need doing and we have a number of workers $W_1, \ldots, W_n$ each charging different rates and each offering to do any one of a subset of the jobs. Maximise the number of jobs done while minimising the cost;

Committee Selection: an organisation sells m products $P_1, P_2, \ldots, P_m$. A certain committee needs to include a product representative for each. A pool of potential members, $M_1, M_2, \ldots, M_n$, has been identified. We want to maximise the number of committee members who are familiar with each product.

We shall see that both problems can be *partially* solved in terms of searching for a cheapest *transversal* in a collection $\mathcal{A}$ of subsets of a set A, that is, a subset T of A which has a distinct element from each member of $\mathcal{A}$. We shall define transversals more formally in a moment (definition 52). First here is an example:

Example 50 *Suppose that there are three workers W_1, W_2, and W_3 and two jobs J_1 and J_2. Suppose W_1 offers to do either job while W_2 is only prepared to do J_2 and W_3 is only prepared to do job J_1. Find an assignment of workers which gets both jobs done. If W_1 charges \$2 to do any job; W_2 charges \$3 and W_3 only \$1, what is the cheapest such assignment?*

Solution We may list the ways each job can get done as subsets of the set $A = \{W_1, W_2, W_3\}$ of workers: $\mathcal{A} = \{\{W_1, W_3\}, \{W_1, W_2\}\}$. The first set specifies the workers who will do job J_1; the second set specifies workers who will do job J_2. Now any transversal for $\mathcal{A}$ will consist of two different workers, one who will do job J_1 and one who will do job J_2. One such transversal is $\{W_1, W_2\}$; another is $\{W_1, W_3\}$ and a third is $\{W_2, W_3\}$. Since W_2 is expensive we choose the second of these three options. So far, this is an *incomplete* solution because we have yet to actually assign jobs to the workers in our chosen transversal $\{W_1, W_3\}$. In fact, there is only one option for us: since W_3 is only offering to do job J_1, W_1 must take J_2.

A transversal may also be referred to as a *system of distinct representatives*. This is a more descriptive name: we may imagine a collection of groups of people each nominating one of their members to represent them; a person may belong to more than one group but they may only accept one nomination, otherwise their loyalties will be divided.

Exercise 51 *Suppose there are six workers, ranked in order of increasing cost: W_1, W_2, W_3, W_4, W_5 and W_6. Suppose there are four jobs J_1, J_2, J_3 and J_4 and the workers offer themselves for these jobs as shown in the following table:*

	W_1	W_2	W_3	W_4	W_5	W_6
Jobs	*1,3*	*1,2*	*2*	*3*	*1,3,4*	*1,4*

Construct the subset collection whose **four** *members are the sets of workers who will offer to do each of the four jobs. Find the cheapest complete transversal of this collection and the most expensive.*

Let us put our discussion on a more formal footing:

Definition 52 *Let A be a finite, non-empty set and $\mathcal{A}$ a collection, $A_1, \ldots, A_s$, of non-empty subsets of A, with $s \geq 1$. A* partial transversal *for $\mathcal{A}$ is a selection of t distinct elements of A from t distinct members of $\mathcal{A}$, where $t \leq s$. If $t = s$ then T is a* complete transversal *(or simply* transversal*) for $\mathcal{A}$. If no (partial) transversal of cardinality $t' > t$ contains T then T is called a* maximal transversal. *If the elements of A are weighted over $\mathbb{R}$ then the* weight *of a (partial) transversal T is the sum of the weights of the elements of T.*

It is very important to remember that the actual subsets of the collection $\mathcal{A}$ in this definition may be *identical* even though they are distinct as members of the collection. This is the case in the following exercise:

Exercise 53 *Suppose $A = \{a, b, c, d\}$ and $\mathcal{A}$ is the collection of four (non-distinct) subsets: $A_1 = \{a, c\}$, $A_2 = \{a, b, d\}$, $A_3 = \{a, c\}$, $A_4 = \{c\}$. Find a maximal transversal for $\mathcal{A}$. If the elements of A are weighted such that $w(a) > w(b) > w(c) > w(d)$ find a minimum-weight maximal transversal.*

We shall now find that the following problem:

MIN-WEIGHT MAXIMAL TRANSVERSAL

Input:	a collection of subsets of a weighted ground set, weights in $\mathbb{R}$
Output:	a minimum-weight maximal transversal for the subsets

is solved by the greedy algorithm.

Theorem 54 *Let A be a finite, non-empty set and let $\mathcal{A}$ be a collection of non-empty subsets of A. Then A, together with the collection of all partial transversals of $\mathcal{A}$, forms a matroid, called the* transversal matroid *of $\mathcal{A}$ and denoted $M(\mathcal{A})$.*

Since MIN-WEIGHT MAXIMAL TRANSVERSAL is a specific instance of the MIN-WEIGHT MAXIMAL MEMBER problem and Theorem 54 guarantees that it satisfies the conditions of Theorem 42 we have:

Corollary 55 *The greedy algorithm solves the MIN-WEIGHT MAXIMAL TRANSVERSAL problem.*

We said earlier that searching for a cheapest transversal could *partially* solve the Job Assignment Problem. Let us return to Exercise 51 to expand on this point, depicting the data from that exercise as a graph: Figure 1.11(a).

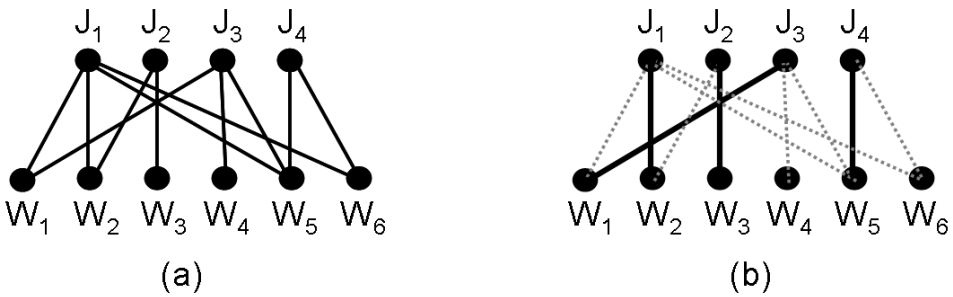


Figure 1.11: graphical representation of Exercise 51.

The collection of four subsets of $\{W_1, W_2, W_3, W_4, W_5, W_6\}$ for which transversals are assignments of workers to jobs is precisely the collection of neighbourhood sets of J_1, J_2, J_3 and J_4: i.e. $\{W_1, W_2, W_5, W_6\}$, $\{W_2, W_3\}$,$\{W_1, W_4, W_5\}$ and $\{W_5, W_6\}$. The solution to Exercise 51 offered the set $\{W_1, W_2, W_3, W_5\}$ as a minimum-weight transversal and this (which happens to be the unique optimal solution) would be found by the greedy algorithm. However, this is *not* a solution to the Job Assignment problem because we have now to select the job which each worker will do: we have chosen the best workers but we do not know why! An actual assignment matching our transversal is shown in Figure 1.11(b) and harks back directly to subsection 1.3.3: it is a maximal matching. As far as our problem is concerned it is automatically minimum-weight, since our greedily constructed transversal achieves that part of the solution; but finding the edges in Figure 1.11(b) is *not* automatic and, as seen in subsection 1.3.3, it cannot be done greedily. Even if we know that only edges joining to W_1, W_2, W_3 and W_5 need be considered, a greedy approach may still fail: we will give J_1 to W_1, then J_2 to W_2; then we are forced to give J_3 to W_5, after which we are stuck!

In fact, the matching problem here is more easily solved than our example in subsection 1.3.3 because it involves a *bipartite graph*; we remarked at the time that in such graphs, precisely those having no odd-length cycles, maximal matchings may be found by 'matroid intersection'. This will be our concern in the next chapter.

Exercise 56 *The Committee Selection problem: suppose an organization sells products P_1, P_2, P_3 and P_4. Sales representatives Abdul, Cheung, Dominique and Enrique have experience with product P_1; Dominique, Enrique and Franz with P_2; Abdul and Baldish with P_3; and Cheung and Dominique with P_4. Draw a bipartite graph representing this information. Form a committee of four so that each product is represented by as many people as possible with experience of that product.*