Matroids And their Graphs

o.mcdonnell4@nuigalway.ie

April 2018

Contents

Abstract

Matroids are an abstraction of the concept of linear dependence which has since developed into the rich field known as matroid theory[4]. They are extremely flexible systems of sets that can be characterised in a number of diverse ways; in this paper we focus on matroids formed from graphs. In particular, we focus on the optimality of the greedy algorithm in solving optimisation problems in graph theory such as the minimal spanning tree problem which can be solved in this way when the independent sets of the graph form a matroid. Futhermore, we show that matroids are the only hereditary systems for which this is true. The paper concludes with some insight into other set-systems which provide optimal solutions for combinatorial optimisation problems through the greedy algorithm.

1 Introduction

1.1 What is a Matroid

Matroid theory arose from the investigations of the mathematician H.Whitney [4] of the concept of linear dependence and so borrows heavily from and abstracts a significant amount of linear algebra concepts particularly with regards matrices and vector spaces. For example, the properties that can be discussed from collections of subsets of linearly independent columns of a given matrix. Another very natural application of matroid theory is graph theory, looking at for example the presence of closed walks, or trees and in particular spanning trees of graphs. This will be the focus in this paper as we follow Oxley's survey of matroid theory[2] and accompanying textbook.[1]

We will begin by describing what it means for a set to be independent and to clarify what it means to be an independent set in a set-system and to describe the structure needed to describe our system as a matroid. This begins with the following definition.

Definition 1.1. An *independence system* is a pair $(E, \mathscr{S})$, where E is a finite set and $\mathscr{S}$ is a collection of sets satisfying the following conditions:
(I1) $\mathscr{S}$ is non-empty.
(I2) $\mathscr{S}$ is a hereditary subset of the power set of E.
The elements of $\mathscr{S}$ are called the *independent sets*.

These independence systems can then be extended into matroids by adding a third axiom, the *exchange axiom*. As we can see from the following definition, all matroids are independence systems but the converse will not be true in general.

Definition 1.2. A matroid is a pair $(E, \mathscr{I})$ with finite ground set E and $\mathscr{I}$ being a collection of independent subsets of E satisfying the following conditions:
(I1): The empty set is always independent
(I2): Every subset of an independent set is independent
(I3): If A and B are two independent sets of $\mathscr{I}$ and $|A| > |B|$, then there exists $x \in A \setminus B$ such that $B \cup \{x\}$ is in $\mathscr{I}$.

The *exchange axiom* can be further characterised into the following form.

Lemma 1.3. *Prove that $(E, \mathscr{I})$ is a matroid if and only if $\mathscr{I}$ satisfies $(I1), (I2)$ and the following condition:*
$(I3)'$ If I_1, I_2 are in $\mathscr{I}$ and $|I_2| = |I_1| + 1$, then there is an element $e \in I_2 \setminus I_1$ such that $I_1 \cup \{e\} \in \mathscr{I}$.

Proof. Suppose $(E, \mathscr{I})$ is a matroid. Then by hypothesis, $\mathscr{I}$ satisies $(I1), (I2)$. Let $I_1, I_2 \in \mathscr{I}$ and $|I_2| = |I_1| + 1$ then $|I_2| > |I_1|$ and so $(I3)'$ holds trivially due to the defnition of a matroid.
Conversely, Suppose $\mathscr{I}$ satisfies $(I2), (I1), (I3)'$
Let I_1, I_2 be in $\mathscr{I}$ such that $|I_2| = |I_1| + 1$, then there exists e in $I_2 \setminus I_1$ such that $I_1 \cup \{e\}$.
Let I_1' be in $\mathscr{I}$ where $I_1' \supseteq I_1$ then e in $I_1' \setminus I_2$ and $|I_1'| > |I_2|$. From this we can see that every independent set A with cardinality greater than an independent set B can be shrunk by removing elements while retaining independence due to $(I2)$(the hereditary property) until we have $|A| = |B| + 1$ and then we can apply $(I3)'$. Therefore the matroid property $(I3)$ is satisfied through $(I3)'$. $\square$

Note. The above definitions of the *exchange axiom*, defined by $(I3), (I3)'$ will be used interchangeably for the remainder of this paper.

The following example draws attention to an important property of matroids,their intersection. The intersection of a matroid is not gauranteed to also be a matroid. This example gives a demonstration of the structure of a matroid defined on a set E.

Example 1.4. Let M_1, M_2 be matroids on a set E. Let $E = \{1, 2, 3, 4\}$
Let $\mathscr{I}_1 = \{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{1, 3\}, \{2, 4\}, \{3, 4\}\}$
Let $\mathscr{I}_2 = \{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{1, 4\}, \{2, 3\}, \{3, 4\}\}$
Therefore, $\mathscr{I}_1 \cap \mathscr{I}_2 = \{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\, \{3, 4\}\}$
let $(E, \mathscr{I}_1 \cap \mathscr{I}_2)$ be a pair, is it a matroid?
Let $I_1 = \{1, 2\}$ and $I_2 = \{3\}$
If there exists an $e \in I_1$ such that $I_2 \cup \{e\} \in \mathscr{I}$ then we have a matroid. Otherwise we do not have a matroid.
$I_2 \cup \{1\} = \{1, 3\} \notin \mathscr{I}$,
$I_2 \cup \{2\} = \{2, 3\} \notin \mathscr{I}$
$\implies (E, \mathscr{I}_1 \cap \mathscr{I}_2)$ is not a matroid.

We will proceed through this paper discussing the various structures of matroids (i.e circuits,bases and rank), illustrating their graph theoretic counterparts and some applications where matroids allow us to avail of some interesting properties. Concluding with a brief overview at some non-hereditary set-systems that share a common application with matroids.

1.2 Enumeration of Matroids

Definition 1.5. Let $\mathscr{I}$ be the collection of subsets of E that do not contain all of the edges of any *cycle* of G. We get a matroid on the edge set of every graph G by defining $\mathscr{I}$ in this way. This matroid is called the *cycle matroid* of the graph G and is denoted $M(G)$. We will later prove that this is a matroid.

Definition 1.6. If M_i, M_j are matroids , then there exists a bijection from the ground set of M_i to the ground set of M_j, such that a set is independent in the first matroid if and only if it is independent in the second matroid, then M_i and M_j are said to be isomorphic.

Note. A matroid that is isomorphic to the cycle matroid of some graph is called graphic. And every graphic matroid is binary

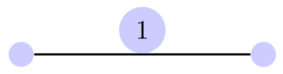
The following table from Oxley's text [1] is enlightening in that it helps us see how many ways a matroid can be defined on a set. The numbers of non-isomorphic matroids and binary matroids on an n-element set for $0 \leq n \leq 8$

n	0	1	2	3	4	5	6	7	8
matroids	1	2	4	8	17	38	98	306	1724
binary matroids	1	2	4	8	16	32	68	148	342

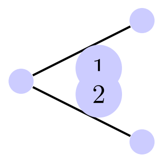
It can be seen from this table, that the number of possible matroids on an n-set grows very rapidly.
Example: Let E be a set, $\{1, 2, 3\}$ then:
Show there are exactly eight non-isomorphic matroids on E. Along with the corresponding Graph of each matroid. This confirms to us the value in the previous table for $n = 3$.
Solution:

$$\{\emptyset\}$$

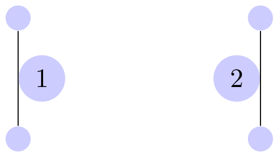
$$\{\{\emptyset\}, \{1\}\} \cong \{\{\emptyset\}, \{2\}\} \cong \{\{\emptyset\}, \{3\}\}$$



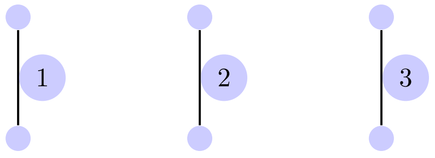
$$\{\{\emptyset\}, \{1\}, \{2\}, \{1,2\}\} \cong \{\{\emptyset\}, \{1\}, \{3\}, \{1,3\}\} \cong \{\{\emptyset\}, \{2\}, \{3\}, \{2,3\}\}$$



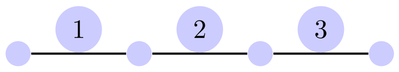
$\{\{\emptyset\},\{1\},\{2\}\}$



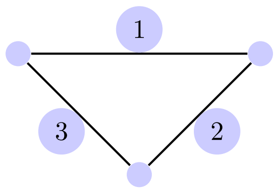
$\{\{\emptyset\},\{1\},\{2\},\{3\}\}$



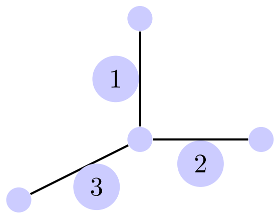
$\{\{\emptyset\},\{1\},\{2\},\{3\},\{1,2\},\{2,3\}\}$



$\{\{\emptyset\},\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\}\}$



$\{\{\emptyset\},\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\},\{1,2,3\}\}$



1.3 Graphs are Matroids

Theorem 1.7. *Let G be a graph and $\mathscr{I}$ be the set of all cyclefree subgraphs of G. Show that if we have the pair $(E, \mathscr{I})$ as defined above by our graph, we have a matroid. In other words, that the cycle matroid $M(G)$ of a graph is a matroid.*

Proof. Let $A, B \in \mathscr{I}$ with $|A| = |B| + 1$. To prove $I3$ of the definition of a *matroid*, We show that for some $a \in A$,
$B \cup \{a\} \in \mathscr{I}$, we should consider $B \cup \{a\}$ for each $a \in A$.
Now suppose $|A| > |B|$ and that $|A| = |B| + 1$
Let $|A \cap B| = s$, $|A \setminus B| = r$, $|A| = s + r$ and $|B| = s + r - 1$
So $|B \setminus A| = r - 1$
Suppose $A \setminus B = \{a_1, a_2, ..., a_r\}$
Suppose $B \cup \{a_i\} \notin \mathscr{I}$ for each $i \in \{1, 2, ...\}$
Consider a_i for $i = 1, 2, ...$ there must be a path $b_{i1}, b_{12}, ..., b_{ir}$ of edges in B such that a_i make a cycle

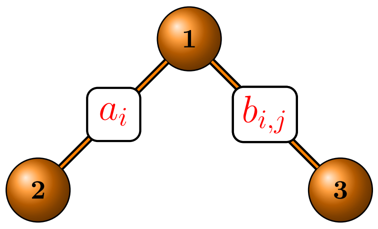


Figure:1.4.1

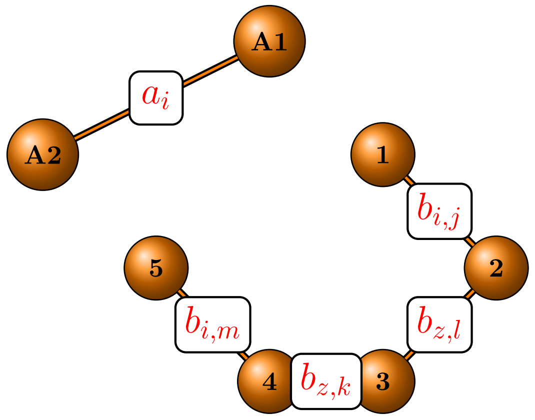


Figure: 1.4.2

Notation: $P(b_j, b_k)$ denotes a set of edges forming path in B from the edges b_j to b_k But $P(b_j, b_k) \cap A$ is not necessarily empty. If $P(b_j, b_k) \subset A$ then $P(b_j, b_k) \cup \{a_i\}$ would be a cycle, then A would not be in $\mathscr{I}$, so at least one of the $b_i \in P(b_j, b_k)$ is contained in $B \setminus A$.
Given $A = \{a_1, ..., a_r\}$ for each a_i associate a $b_i \in B \setminus A$. Let $\hat{B} = \{b_1, ..., b_r\}$
Case 1: The b_i's are distinct
The b_i's are distinct and as shown previously each of the b_i's must be in $|B \setminus A|$ in order to avoid a circuit in A.
Therefore,$|B| \geqslant A$. Contradicting $|A| > |B|$.
Hence, I3 holds.
Case 2: When the b_i's are not all distinct. Let $b_1 = b_2$.

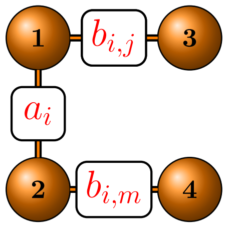


Figure: 1.4.3

Picture *figure 1.4.3* in place of *figure 1.4.1* above and observe how this would affect the graph of B in *figure 1.4.2*

We use the same argument as in Case 1 only here we need two distinct $b_i \in P(b_j, b_k)$ where $b_i \in B \setminus A$ such that $P(b_j, b_k) \cup \{a_i\}$ is a cycle. This can be seen in the diagram above, there must be another edge in the union of the paths which is in $B \setminus A$ or else we get a cycle in A. Otherwise,$P(b_j, b_k) \subset A$ then $P(b_j, b_k) \cup \{a_i\}$ would be a circuit and then $A \notin I$. Therefore, $|B| \geq |A|$, and we have a contradiction.
Hence I3 holds. □

2 Cryptomorphisms

One of the most interesting properties of matroids is their resulting cryptomorphisms.[1] I previously mentioned that matroids are very flexible systems that can be adapted in a number of diverse ways. The cryptomorpshims are the key to this. Once we find some property that a matroid satisfies we can then extend the axioms to include a form of that property. This leads to multiple characterisations of matroids and is what causes matroids to naturally appear so often in seemingly unrelated fields of mathematics such as algorithms,graph theory,finite geometry etc.
In this section we explore some of the more advantageous characterisations of matroids used in this project, namely the circuits and bases.

2.1 Circuit characterization of a matroid

In the paper of Oxley [2] it is said that "A set in a matroid that is not independent is called dependent.The hereditary property,(I2), means that a matroid is uniquely determined by its collection of maximal independent sets, which are called bases, or by its collection of minimal dependent sets, which are called circuits."

The circuits of a graph are a rather intuitive concept, and so it is natural to define the cycle matroid $M(G)$ of a graph in terms of its circuits. These circuits are the "edge sets of cycles in a graph G". All dependent sets in a matroid contain a circuit.

A circuit of a matroid is defined as a minimally dependent set due to the fact that a circuit is a dependent set with the minimum number of edges to be dependent. i.e if you remove one edge we have an independent set.

Definition 2.1. By using (I1)–(I3) , it is not difficult to show that the collection $\mathscr{C}$ of circuits of a matroid M has the following three properties:
(C1) The empty set is not in $\mathscr{C}$
(C2) No member of $\mathscr{C}$ is a proper subset of another member of $\mathscr{C}$
(C3) if C_1 and C_2 are distinct members of C and $e \in C_1 \cap C_2$, then $(C_1 \cup C_2) \setminus \{e\}$ contains a member of $\mathscr{C}$

The following two proofs appear in Oxley's text and shows how the circuits define a matroid and how circuits appear in graphs.[1]

Theorem 2.2. *Let M be a matroid and $\mathscr{C}$ be its collection of circuits. Then $\mathscr{C}$ satisfies (C1) - (C3)*

Proof. $(C1)$ is obvious as by $(I1)$ the empty set must always be an independent set.
$(C2)$ is also straightforward because any $C \in \mathscr{C}$ is a minimally independent set by definition. Therefore , if there exists a $C_1 \in \mathscr{C}$ such that $C_1 \subset C$ then $C_1 \in \mathscr{C}$ and C is not a minimally dependent subset of E.

[1]In mathematics, two objects, especially systems of axioms or semantics for them, are called cryptomorphic if they are equivalent but not obviously equivalent.[12]

$(C3)$ Let $A, B \in \mathscr{C}$ and suppose that (seeking a contradiction) $(A \cup B) \setminus \{e\}$ where e is $\in (A \cap B)$ does not contain a circuit.
Then $(A \cup B) \setminus \{e\}$ is independent and therefore in $\mathscr{I}$.

The set $A \setminus B$ is non-empty.
Let $s \in A \setminus B \implies s \in A$
as A is in $\mathscr{C}$ it is minimally dependent. $\implies A \setminus \{s\} \in \mathscr{I}$ i.e is independent.

Let J be a maximal independent set of $(A \cup B)$ with the following properties: $A \setminus \{s\} \subset J$ and therefore $\{s\} \notin J$ but as B is a circuit there must be some element $t \in B$ that is not in J; s and t are distinct.

$\implies |J|$ must be at most equal to $|(A \cup B) \setminus \{s, t\}|$
$\implies |J| \leq |(A \cup B) \setminus \{s, t\}| = |(A \cup B)| - 2 < |(A \cup B) \setminus \{e\}|$

Now by (I3) we can substitute elements from $(A \cup B) \setminus \{e\}$ into $(A \cup B) \setminus \{s, t\}$ that are not in $(A \cup B) \setminus \{s, t\}$ but the only elements that fits this condition are $\{s\}$ and $\{t\}$ and introducing either of these elements breaks the independence of J.
Therefore, $(A \cup B) \setminus \{e\}$ must contain a circuit

□

Theorem 2.3. *Let E be the edge set of a graph G and let $\mathscr{C}$ be the edge sets of cycles in G.*
Then $\mathscr{C}$ is the set of circuits of a matroid.

Proof. Let $A, B \in \mathscr{C}, A \neq B$ and let $e \in A \cap B$.

We must now construct a minimal cycle of G whose edge set is contained in $(A \cup B) \setminus \{e\}$.
For $i = 1, 2, 3, ...$ let P_1 be a path whose edge set is $A \setminus \{e\}$.
$A \setminus \{e\} \in \mathscr{I}$ therefore P_1 is not a cycle of G. This path will traverse from the edge a_j to a_u where u, j were the vertices connecting the edge e to $(A \cup B) \setminus \{e\}$ to make $A \cup B$.
Now perform the same procedure for a path P_2 whose edge set is $B \setminus \{e\}$.
P_1 and P_2 should meet at the junctions u, v, where e was removed to make $(A \cup B) \setminus \{e\}$.
Therefore $P_1 \cup P_2$ should be a cycle of G.
$\implies$ (C3) holds.
$\implies \mathscr{C}$ is the set of circuits in G.

□

2.2 Base Characterisation of a Matroid

As quoted above, the bases of a matroid are the maximally independent sets of $\mathscr{I}$. These are most naturally visualised as being akin to a basis in a matrix. Where the basis of a matrix is a spanning set and every vector is a linearly combination of the basis. Bases are also spanning sets in matroid. And as such have very useful properties that we will use extensively in later sections. Most

notably, the collection of subsets $\mathscr{I}$ can be generated using the bases (bases are not in general unique) due to the hereditary property $(I2)$.
The below theorem gives us a useful property that can be applied to the bases of a matroid, namely that the bases all have the same cardinality.

Theorem 2.4. *Show that if $\mathscr{I}$ is a non-empty hereditary set of subsets of a finite set E, then $(E, \mathscr{I})$ is a matroid if and only if, for all $X \subset E$, all maximal members of $\{I : I \in \mathscr{I}$ and $I \subset X\}$ have the same number of elements.*

Proof. Let B_1, B_2 be maximal elements of $\{I : I \in \mathscr{I} \text{and } I \subset X\}$
Assume $|B_1| < |B_2|, B_1, B_2 \in \mathscr{I}$ and since we have a matroid
there exists $e \in (B_2 \setminus B_1)$ such that $B_1 \cup \{e\} \in \mathscr{I}$.
This contradicts the maximality of B_1.
$\implies$ All maximal elements of the set $\{I : I \in \mathscr{I}$ and $I \subset X\}$ in our matroid M have the same cardinality.
Conversely,... □

And now similar to with circuits, we can characterise a matroid by way of it's bases as proved below.

Definition 2.5. A base is a maximally independent subset of $\mathscr{I}$.
As seen previously all maximally independent sets in a matroid have the same cardinality.

Theorem 2.6. *Let $\mathscr{B}$ be a set of subsets of a finite set E. Then $\mathscr{B}$ is the collection of bases of a matroid on E if and only if $\mathscr{B}$ satisfies the following conditions:*
(B1) $\mathscr{B}$ is non-empty.
(B2) If B_1 and B_2 are members of $\mathscr{B}$ and $x \in B_1 \setminus B_2$, then there is an element y of $B_1 \setminus B_2$ such that $(B_1 \setminus \{x\}) \cup \{y\} \in \mathscr{B}$.

Proof:
By (I1) $\emptyset$ is always independent, so $\mathscr{B}$ must always contain at least the $\emptyset$, (B1) holds.
Let $B_1, B_2 \in \mathscr{B}, B_1 \neq B_2$.
$|B_1| = |B_2|$ so (I3) does not directly apply here, as any element of cardinality larger that $|B_i|$, where $B_i \in \mathscr{B}$ will be contained in $\mathscr{C}$(the set of circuits).
Let $x \in B_1 \setminus B_2 \implies x \in B_1, x \notin B_2$.
$|B_1| = |B_1 \setminus \{x\}| + 1 \implies B_1 \setminus \{x\} \in \mathscr{I}$ but not in $\mathscr{B}$.
$|B_2| = |B_1 \setminus \{x\}| + 1$ so now we can apply (I3).
Therefore, there exists a $y \in B_2 \setminus B_1$ such that $(B_1 \setminus \{x\}) \cup \{y\} \in \mathscr{I}$
$|(B_1 \setminus \{x\}) \cup \{y\}| = |B_1 \setminus \{x\}| + 1 = |B_1| = ... = |B_r|$ where the B_i are maximal elements of $\mathscr{I}$ and have the same cardinality.
$\implies (B_1 \setminus \{x\}) \cup \{y\}$ is maximal in $\mathscr{I}$.
$\implies (B_1 \setminus \{x\}) \cup \{y\} \in \mathscr{B}$, (B2) holds.

Conversely, suppose that $\mathscr{B}$ satisfies (B1) and (B2).
By (B1), $\mathscr{B}$ is always non-empty which shows (I1) holds.

By definition, a base $B_1 \in \mathscr{B}$ is a maximally independent subset of E. Then for all $B_i \in \mathscr{B}$ the subsets $b_{i,k} \subseteq B_i$ are independent. Therefore all the $b_{i,k}$ are in $\mathscr{I}$.
$\implies$ (I2) holds. Showing a matroid can be generated through the bases.
Next, assume that (I3) fails. That, for $I_1, I_2 \in \mathscr{I}$, where $|I_1| = |I_2| + 1$,
there exists a $y \in I_1 \setminus I_2$ such that $I_2 \cup \{y\} \in \mathscr{I}$.
Let $B_1, B_2 \in \mathscr{B}, |B_1| = |B_2|$.
Let $x \in B_1 \setminus B_2$ then $B_1 \setminus \{x\} \subset B_1$.
$\implies$ $B_1 \setminus \{x\}$ is independent.
Then there exists a $y \in B_2 \setminus (B_1 \setminus \{x\})$ such that $(B_1 \setminus \{x\}) \cup \{y\} \in \mathscr{B}$ from (B2). And if $(B_1 \setminus \{x\}) \cup \{y\} \in \mathscr{B}$ it is also in $\mathscr{I}$. A contradiction.
$\implies$ (I3) holds and we have a matroid.

$\square$

3 Graph Theory

3.1 Graphs and their corresponding Matroid Theory

In this section we introduce the graph theoretic definitions and results needed to understand the algorithms and subsequent optimisation in later sections. These definitions and results come mainly from Jungnickel's[3] text. We will also see how these properties of graphs naturally align with the properties of matroids and in particular to bases of matroids which are the cornerstone of the results in the later sections.

Definition 3.1 (Connected)**.** A graph is connected when there is a path between each pair of vertices.

Definition 3.2 (Acyclic)**.** An acyclic graph is a graph which contains no closed walks.

Definition 3.3 (Walk)**.** If there are vertices $v_{i-1}v_i$ for $i = 1, ..., n$ the sequence is called a walk. If $v_0 = v_n$ this sequence is called a closed walk.

Definition 3.4 (Tree)**.** A connected graph containing no circuits. In other words, acyclic graphs.
A forest is a disconnected graph containing no circuits

We know from *theorem 1.7.* If we let E be the edge set of our graph G. And we define $\mathscr{I}$ as the subsets of E not containing all of the edges of any cycle in G we have $M(G)$, the cycle matroid of G. We know that this is a matroid, and it is easy to see that in this case the elements of $\mathscr{I}$ correspond exactly to the trees of G.

Definition 3.5 (Spanning Tree)**.** A spanning tree T of an graph G is a subgraph that is a *tree* which includes all of the vertices of G.
A disconnected graph cannot contain a spanning tree as we cannot find a walk which brings us to all of the disconnected vertices.

Proposition 3.6. *A theorem of Cayley(1889) states that the number of distinct laballed trees which can be drawn using n labelled points is n^{n-2}.*

Corollary. *The number of distinct labelled spanning trees which can be drawn using n labelled points is n^2.*

Remark. Determining the number of spanning trees of a graph in polynomial time is NP-hard.

Lemma 3.7. *[3] Any acyclic graph on n vertices has at most $n - 1$ edges.*

Proof: Let G be an acyclic graph with n vertices.
If n = 1 then we have no edges hence nothing to prove.
Assume $n > 1$, let e be an edge in G connecting two vertices ab in the vertex

set of G.
Let $H = G\backslash\{e\}, H$ has one more connected component than G. H has two maximal acyclic connected components, and thus can be decomposed into acyclic connected graphs $H_1, H_2, ..., H_k$ where $k \geq 2$.
By induction, we can assume each graph H_i contains at most $n_i - 1$ edges where n_i is the number of vertices of H_i.
Then G has at most $n - 1$ edges.
$(n_1 - 1) + ... + (n_k - 1) + 1 = (n_1 + ... + n_k) - (k - 1) \leq n - 1$ edges.

□

Definition 3.8 (Bridge)**.** A bridge(or cut edge) is an edge of a graph whose deletion increases the number of connected components. Equivalently, an edge is a bridge if and only if it is not contained in any cycle.

The above definition is mainly useful for us in proving the next lemma, which comes from Jungnickel's text and will help us visualise the characteristics of a spanning tree of a graph. In particular condition (1) and (2), will allow us to see that the spanning trees of a graph directly correspond to the bases of a matroid.

Lemma 3.9. *[3] Let G be a graph. Then the following conditions are equivalent:*
1) G is a tree.
2) G does not contain any cycles, but adding any further edge yields a cycle.
3) Any two vertices of G are connected by a unique path.
4) G is connected, and any edge of G is a bridge.

Proof: (1) $\Longrightarrow$ (2)
Suppose that G is a tree,then G is a connected graph with no circuits. Let e be a new edge in G with $e = g_i g_k$ where g_i, g_k are in the vertex set of G. Then as $G \cup \{e\}$ must be connected, there exists a walk between any pair of vertices of G. So there is a walk K from $g_j \rightarrow g_k$ and there is also a walk L from $g_k \rightarrow g_j$ where K does not traverse e and L does traverse e and so we have a cycle.

Proof: (2) $\Longrightarrow$ (3)
Let u, v be vertices of G. If there was no path joining uv in G then $e = uv$ does not create a cycle in G. Thus G must be connected.
Suppose G contained two different paths W_1, W_2 from u to v.
Then $u \longrightarrow v \longrightarrow u$ would be a closed walk in G.
$\Longrightarrow$ G contains a cycle. Which is a contradiction.

Proof: (3) $\Longrightarrow$ (4)
G is connected by hypothesis. Let $e = uv$ be an edge in G.
Suppose e is not a bridge, then $G \setminus \{e\}$ is still connected. But then we have two distinct paths from u to v in G.

Proof: (4) $\Longrightarrow$ (1)
G is connected by hypothesis.
Suppose G contains a cycle K. Then any edge of K could be ommited from G,

and the resulting graph would still be connected. In other words, no edge of K would be a bridge, a contradiction.

□

The vertices of a graph/network can be labelled and referred to as nodes. Information may then be recorded in them along with a cost, penalty or probability associated with each edge. For example, the problem of joining all nodes in a graph by the minimum length using our respective metric to a tree known as a *minimum spanning tree.*
Weights can be assigned using a process as detailed below. We will later redefine this process in a more suitable way to take advantage of matroid properties in order to determine the minimum spanning trees in graphs.

Definition 3.10. Let (G, ω) be a network. For any subset T of the edge set of G, ω is called the weight of T.

$$\omega(T) = \sum_{e \in T} \omega(T) \tag{1}$$

Definition 3.11 (Minimal Spanning Tree)**.** A spanning tree is a *minimal* spanning tree if its weight is minimal of all the weights of spanning trees. A forest can be considered by finding a minimal spanning tree for each connected component of G.

Remark. If the weight ω is constant, any spanning tree is minimal.
In this case, determining a minimal spanning tree could be done using a breadth-first search. Which is a similar procedure to the described depth-first section in *appendix,algorithm 6.*

4 Greedy Algorithm

4.1 Generic greedy algorithm

The greedy algorithm is an algorithmic paradigm. It does not always provide a solution in general, not least an optimal solution. As such the greedy algorithm is a description of a problem solving heuristic. It says that when trying to solve a problem we should choose the local optimum at each iteration in the hope of finding a global optimum. In this section, we will introduce the generic procedure that makes the gredy algorithm and showing how it can be modified in a variety of ways, including for compatibility with matroids.

Algorithm 1 Greedy algorithm

Let $(E, \mathscr{S})$ be an independence system and $\omega : E \longrightarrow \mathbb{R}^+$

```
1: procedure GREEDY($E, \mathscr{S}, \omega, T$)
2:     order the elements of $E$ according to their weight
3:     $E = \{e_1, ..., e_m\}$ with $\omega(e_1) \geq \omega(e_2) \geq ... \geq \omega(e_m)$
4:     $T \leftarrow \emptyset$
5:     for $k = 1$ to $m$ do
6:         if $T \cup \{e_k\} \in \mathscr{S}$ then
7:             append $e_k$ to $T$
```

We can see in the above pseudocode (taken from [3]) that this is a sequential iterative algorithm. Our data is kept in a list sorted by weight: heaviest to lightest. We then select the heaviest entry in our list at each iteration and append it to our initial empty variable T. By the end of the process we should have joined a certain number of the elements to our variable T. Where T at the time of termination is the greedy algorithm solution.
This is how the algorithm looks in the most abstract sense, later we will see it used as a way of solving the *minimal spanning tree* problem mentioned in *definition 3.11.*

4.2 Modified greedy for Matroids

Here, we will see how this algorithm corresponds to matroids. The greedy algorithm above, constructed a maximal weight element from our list called T. Depending on the structure of our system and the termination clause this element's cardinality can differ. It is our hope to show that when the greedy algorithm is applied to a matroid we get an optimal solution, meaning that we generate a base of the matroid. However, for now it is enough to show that the greedy algorithm always produces a solution in general. This is illustrated by the below pseudocode, which was adapted from Oxley's[1] description.

Algorithm 2 Greedy algorithm

The *greedy algorithm* for the pair $(\mathscr{I}, \omega)$ is as follows:

```
1: procedure GREEDY($\mathscr{I}, \omega$)
2:     Set $x_0 = \emptyset$ and $j = 0$
3:     if $\exists e \in E \setminus x_j$ such that $x \cup \{e\} \in \mathscr{I}$ then
4:         Choose such an element $e_{j+1}$ of maximum weight,
5:         let $x_{j+1} = x_j \cup \{e_{j+1}\}$ and
6:         GREEDY($\mathscr{I}, \omega$)
7:     else
8:         Let $x_j = B_G$
9:         return $x_j$
10:    $j++$
```

The procedure described above is as follows: Select the emptyset as your first element denoted x_0, as the empty set is always independent. We want to build the set x. Now as before, order your elements from heaviest to lightest. Then select the heaviest weighted element possible such that $x \cup \{e\} \in \mathscr{I}$ where e is the selected element. Otherwise we want to return x, as x is then equal to the solution of the greedy algorithm, i.e B_G (the base generated by the greedy algorithm). The element e should then be removed from further selection, and the process should be called recursively until there is no possible element that can be added to the set x without inducing a circuit.

Remark. It is interesting to note that in this way we should find a maximal member B_{max} of $\mathscr{B}$ (the collection of bases of a matroid) and we will prove that this is certainly the case later in *theorem 5.2.* But if we negate this process in the following way we can use this exact procedure to find a minimal member B_{min} of $\mathscr{B}$. This works as follows:
Let $\omega : E \longrightarrow \mathbb{R}^-$ be the weight function, we want to find an independent set A whose weight is maximal, where

$$\omega(A) := \sum_{e \in A} |\omega(e)| \tag{2}$$

Due to the fact that our weights are now negative real numbers, finding the maximal element at each iteration corresponds to finding the value with the minimal absolute value. And so through completing the greedy algorithm process we should successfully find our minimal element B_{min} of $\mathscr{B}$.

5 Optimization Problems

5.1 Optimisation Example: Kruskal's Algorithm

Example 5.1. Suppose you are from a country called Xzghtyta which contains n cities that are currently all isolated from each other. As the new minister for transport it is your idea to correct this transport issue and to lay a railroad which should connect all the cities of your country. However, you have a budget. Each railway line will cost a certain predefined amount to lay (with no difficulties or unforeseen costs). How will you decide which city-links are the optimal ones to lay railtracks on?

We will soon see that this can be done by finding a minimal spanning tree. Which can be found through a greedy algorithm process. A suitable example of this kind of algorithm which should solve our problem is Kruskal's Algorithm, which is detailed below.

Algorithm 3 Kruskal's algorithm

Let G be a connected graph with vertex set $V = \{1, ..., n\}$ and $\omega : E \longrightarrow \mathbb{R}^+$ a weight function. The edges of G are ordered according to their weight, that is, $E = \{e_1, ..., e_m\}$ and $\omega(e_1) \leq ... \leq \omega(e_m)$.

```
1: procedure KRUSKAL($G, \omega, T$)
2:     $T \leftarrow \emptyset$
3:     for $k = 1$ to $m$ do
4:         if ACYCLIC($T \cup \{e_k\}$) then
5:             append $e_k$ to $T$
```

This process is very similar to our prior description of the generic greedy algorithm. Although this time, we sort the elements of our edge set from lightest to heaviest. And then we iterate through. At each iteration, we check if the created set created by $T \cup \{e_k\}$ at each iteration remains independent and if so we add e_k to the generated set and if not we ignore that element and move to the next in the list E.

For graphs, the process above is equivalent to the following semi-pseudocode. That we make a forest containing just the vertices of G; and then we iteratively add the desired edges to that forest as long as no cycles are induced in the resulting subgraph of G.

```
1) Create a graph $F$ containing just the vertices of $G$.
2) Create a set $S = E(G)$; the edge set of $G$.
3) While $S$ is non-empty and $F$ is not yet spanning
3($a$) Remove an edge with minimum weight from S.
3($b$) If the removed edge introduces no cycles to $F$
then add the edge to $F$
```

In the next sub-section we will prove the correctness of algorithms such as this. It is also notable to mention that Kruskal's algorithm will terminate after exactly $n-1$ iterations, where n is the number of vertices in a graph. This corresponds to *lemma 3.7* which says a spanning tree has exactly $n-1$ edges. Any more edges added at that point will induce a cycle and so kruskal's algorithm terminates.

5.2 Proofs of correctness

Our optimisation problem described previously can now be restated as follows for mathematical clarity.
Problem: *Find a maximal member B of $\mathscr{I}$ of maximum weight.*

Note. Let B_G be a base of a matroid generated by the greedy algorithm.

Theorem 5.2. *If $(E, \mathscr{I})$ is a matroid M, then B_G is a solution to the optimization problem.*

Proof. If $r(M) = r$, then $B_G = \{e_1, e_2, ..., e_r\}$ is a basis of M. Let B be another basis of M, $B = \{f_1, f_2, ..., f_r\}$ where $\omega(f_1) \geq \omega(f_2) \geq ... \geq \omega(f_r)$. We claim that $\omega(e_j) \geq \omega(f_f)\ \forall j$, then it follows that $\omega(B_G) \geq \omega(B)$ for any other basis in $\mathscr{B}$. □

Lemma 5.3. *If $1 \leq j \leq r$, then $\omega(e_j) \geq \omega(f_j)$.*

Proof. Suppose (seeking a contradiction) that k is the least integer for which $\omega(e_k) < \omega(f_k)$. Take $I_1 = \{e_1, e_2, ..., e_{k-1}\}$ and $I_2 = \{f_1, f_2, ..., f_k\}$. Since $|I_2| = |I_1| + 1$ $(I3)$ implies $I_1 \cup \{f_t\} \in \mathscr{I}$ for some $f_t \in I_2 \setminus I_1$. But this means that $\omega(f_t) \geq \omega(f_k) > \omega(e_k)$ and hence the Greedy algorithm would have chosen f_t over e_k, which gives us our contradiction. □

The following two propositions appear as excercises in Oxley's text[1]

Proposition 5.4. *Let M be a matroid and $\omega : E(M) \longrightarrow \mathbb{R}^+$ be a one-to-one function. Then M has a unique basis of maximum weight.*

Proof. Let ω be an injective function, this will mean that each edge has a unique weight.
We want to find an independent set A whose weight is maximal, where

$$\omega(A) := \sum_{e \in A} \omega(e). \tag{3}$$

We can then arrange our edges in a set S by order of decreasing weight such that $\omega(e_1) \geq \omega(e_2) \geq ... \geq \omega(e_k)$.
We have already seen in *theorem 5.1* that the greedy algorithm as described in *algorithm 2/4.1* provides a solution to this optimisation problem. And since there is no repitition in weights there is no point in the algorithm where there is more than a single choice as to the next chosen edge. Therefore there is only one possible solution when our weight function is injective. □

Remark. If we lose the injectivity condition, then this is not the case and we cannot guarantee uniqueness in general. This can be caused by selecting edges of equal weight in differing orders, and by choosing in this way a cycle may be induced and so the order in which those edges of equal weights are selected affects the resulting spanning trees as will be seen in the next theorem.

Proposition 5.5. *Let $M = (E, \mathscr{I})$ be a matroid and $\omega : E(M) \longrightarrow \mathbb{R}^+$. When the greedy algorithm is applied to the pair $(\mathscr{I}, \omega)$, each iteration of the greedy algorithm involves a potential choice. Thus, in general, there are a number of different sets that the algorithm can produce as solutions to the optimisation problem $(\mathscr{I}, \omega)$. Let $\mathscr{B}_G$ be the set of such sets and let $\mathscr{B}_{max}$ be the set of maximum weight bases of M, then $\mathscr{B}_G = \mathscr{B}_{max}$.*

Proof. Suppose ω is an injective function, then we have shown there is a unique maximum weight basis for M in *theorem 5.4* and so the proof of this trivial.
Now suppose ω is not injective.This means maximal weight bases of M are not in general unique.
If $r(M) = r$, then $B_G = \{e_1, e_2, ..., e_r\}$ is a basis of M. Let B'_G be another basis of M, $B'_G = \{f_1, f_2, ..., f_r\}$. Both B_G, B'_G are bases generated through the greedy algorithm as described in *section 4.1.* We arrange both these bases in terms of decreasing order where $\omega(e_1), \omega(f_1)$ are the heaviest elements in their respective bases.
As ω is not an injective function, at least one element in B_g and B'_G is distinct. We also know from *theorem 5.1* that the greedy algorithm finds a maximal member B of $\mathscr{I}$ of maximum weight.

Therefore, any base generated through the greedy algorithm is maximally weighted. Meaning, that $\omega(B_G) = \omega(B'_G)$. As if $\omega(B_G) < \omega(B'_G)$ then this would mean the greedy algorithm does not find a solution to our optimisation problem, contradicting *theorem 5.1.* And therefore, $\omega(e_j) = \omega(f_j)\ \forall j$ and since all these bases are maximally weighted, $\implies \mathscr{B}_G = \mathscr{B}_{max}$. $\square$

5.3 Cryptomorphism: Greedy

Theorem 5.6. *Let $\mathscr{I}$ be a collection of subsets of a set E. Then $(E, \mathscr{I})$ is a matroid if and only if $\mathscr{I}$ satisfies the following conditions:*
$(I1)$ $\emptyset \in \mathscr{I}$
$(I2)$ If $I \in \mathscr{I}$ and $I' \subset I$ then $I' \in \mathscr{I}$
(G) For all weight functions $\omega : E \longrightarrow \mathbb{R}^+$, the greedy algorithm produces a maximal member of $\mathscr{I}$ of maximum weight.

Proof. Suppose $(E, \mathscr{I})$ is a matroid. Then $\emptyset \in \mathscr{I}$ and (I2) holds trivially. And by *theorem 5.2* we know that the greedy algorithm can find a maximal member $B \in \mathscr{B}$ of maximum weight if $(E, \mathscr{I})$ is a matroid.
Conversely, suppose $(E, \mathscr{I})$ is a pair satisfying $(I1)$, $(I2)$ and (G). Need to prove $\mathscr{I}$ satisfies $(I3)$ in order to have a matroid.
Suppose that (seeking a contradiction), if $I_1, I_2 \in \mathscr{I}$ with $|I_2| > |I_1|$ such that $I_1 \cup \{e\} \in \mathscr{I}$.

Now, $|I_1 \setminus I_2| < |I_2 \setminus I_1|$ and $I_1 \setminus I_2$ is non-empty.
So we can choose an $\epsilon > 0$ such that

$$0 < (1+\epsilon)(|I_1 \setminus I_2|) < |I_2 \setminus I_1| \tag{4}$$

Define $\omega : E \longrightarrow \mathbb{R}^+$ by:

$$\omega(e) = \begin{cases} 2 if e \in I_1 \cap I_2 \\ \frac{1}{|I_1 \setminus I_2|} if e \in I_1 \setminus I_2 \\ \frac{1+\epsilon}{|I_2 \setminus I_1|} if e \in I_2 \setminus I_1 \\ 0, otherwise. \end{cases}$$

We need the greedy algorithm to fail for only one weight function to get our contradiction.

- The greedy algorithm will choose all the elements of $I_1 \cap I_2$ first as they are the heaviest elements.
- Then it will choose all the elements of $I_1 \setminus I_2$.
- By assumption, it cannot then pick any element of $I_2 \setminus I_1$. Thus the remaining elements of B_G will be in $E \setminus (I_1 \cup I_2)$.

Hence,

$$\omega(B_G) = 2|I_1 \cap I_2| + |I_1 \setminus I_2| \left(\frac{1}{|I_1 \setminus I_2|} \right) = 2|I_1 \cap I_2| + 1$$

But by (I2), I_2 is contained in a maximal member B_2 of $\mathscr{I}$ and, $I_2 \subset B_2$.

$$\omega(B_2) \geq \omega(I_2) = 2|I_1 \cap I_2| + |I_2 \setminus I_1| \left(\frac{1+\epsilon}{|I_2 \setminus I_1|} \right) > 2|I_1 \cap I_2| + 1 = \omega(B_G)$$

$\implies \omega(B_2) > \omega(B_G)$
Which means the greedy algorithm does not find a solution to our optimisation problem shown by *theorem 5.2*, so the greedy algorithm fails for this weight function. We have a contradiction.
$\implies$ (I3) holds.
$\implies$ $(E, \mathscr{I})$ is a matroid. $\square$

6 Other Optimizable set-systems

Matroids have now been shown to be a hereditary set-system which naturally works well with the greedy algorithm, guaranteeing optimal solution to optimisation problems. We have also seen in *theorem 5.6* that matroids are the only hereditary set-system that have this property. But what about non-hereditary set-systems? Are their other systems that yield optimal solutions to optimisation problems by way of the gredy algorithm? and if so how much structure must be imposed on the set-systems to guarantee this property? These are the questions we try to explore using Jungnickel's text.[3] In this section, proofs are omitted as they are also omitted in the text and are beyond the scope of this project. For proofs or further investigation, refer to [10], [9], [8].

6.1 Accessible Set-Systems

Definition 6.1. An *accessible set-system* is a pair $M = (E, \mathscr{S})$ where E is a finite ground set and $\mathscr{S}$ is a non-empty subset of the power set of E. Elements of $\mathscr{S}$ are called the *feasible sets* of M. Maximal feasible sets are also called *bases*. $\mathscr{S}$ satisfies the following *accessibility axiom*:
(A) For any non-empty feasible set $X \in \mathscr{S}$ there exists an element $e \in X$ such that $X \setminus \{e\} \in \mathscr{S}$.

Remark. Matroid $\supseteq$ Independence System $\supseteq$ Accessible Set-System.
All matroids are independence systems and all independence systems are accessible set-systems but the converse is not true in general.

The axiom (A) is needed due to the process of the greedy algorithm. We require the ability to sequentially select a single element and then union it to our constructed solution at each step so an arbitrary set system cannot guarantee this. Axiom (A) however ensures that at each step such an element does in fact exist, although the subset of possible choices for this element may not be the same at each iteration of the process, which leads to further possible issues.
Our general problem can now be restated as:

Proposition 6.2. *For any accessible set-system M and any weight function $\omega : E \longrightarrow \mathbb{R}^+$, the optimisation problem is:*
Maximise $\omega(B)$ such that B is a basis of M.

We can now apply a modified greedy algorithm in order to find a solution to this generalised version of our matroid/independence system problem.

Algorithm 4 Greedy algorithm for accessible set-systems

Let $M = (E, \mathscr{S})$ be an accessible set-system and $\omega : E \longrightarrow \mathbb{R}^+$ a weight function.

```
1: procedure GREEDY($E, \mathscr{S}, \omega, T$)
2:     $T \leftarrow \emptyset$, $X \leftarrow E$
3:     while there $\exists x \in X$ with $T \cup \{x\} \in \mathscr{S}$ do
4:         choose some $x \in X$ with $T \cup \{x\} \in \mathscr{S}$ and
5:         $\omega(x) \geq \omega(y)$ $\forall y \in X$ with $T \cup \{y\} \in \mathscr{S}$
6:         $T \leftarrow T \cup \{x\}$, $X \leftarrow X \setminus \{x\}$
```

Using our above definitions and algorithm we can now begin finding solutions to our problem for any accessible set-system. However, we are interested in learning about the characterisations of these set-systems that lead to optimal solutions through the use of our greedy algorithm as described in *section 5.2*. One such characterisation is the concept of a *greedoid*. A greedoid is an accessible set-system satisfying (I3). Formally this means,

Definition 6.3. A *greedoid* is a pair $(E, \mathscr{S})$ where E is a finite ground set and $\mathscr{S}$ is a collection of the feasible subsets of E satisfying the following conditions:
(I1): $\mathscr{S}$ is non-empty, $\emptyset \in \mathscr{S}$.
(A): For any non-empty feasible set $X \in \mathscr{S}$ there exists an element $e \in X$ such that $X \setminus \{e\} \in \mathscr{S}$.
(I3): If A and B are two independent sets of $\mathscr{S}$ and $|A| > |B|$, then there exists $x \in A \setminus B$ such that $B \cup \{x\}$ is in $\mathscr{S}$.

Unfortunately the greedy algorithm while providing solutions does not gaurantee optimal solutions for all greedoids. To characterise the greedoids that do result in optimal solutions when the greedy algorithm is applied we must add to our existing machinery an additional axiom. This is called the *strong exchange axiom*. This axiom is a strong version of (I3): the exchange axiom of a matroid.

Proposition 6.4. *Let $M = (E, \mathscr{S})$ be a greedoid. Then the above modified greedy algorithm finds an optimal soltuion to our problem for any weight function $\omega : E \longrightarrow \mathbb{R}^+$ if and only if M satisfies the follwing axiom:*
(SE): For $A, B \in \mathscr{S}$ with $|A| = |B| + 1$, there always exists some $e \in A \setminus B$ such that $B \cup \{e\}$ and $A \setminus \{e\}$ are contained in $\mathscr{S}$. Which is very reminiscent of lemma 1.3.

Remark. It can be seen that this axiom holds trivially for matroids due to the hereditary condition.

Remark. The amount of the possible greedoids which can yield optimal solutions can be further characterised using further abstracted properties of matroids, similar to axiom (SE), for this I refer you to Jungnickel's text[3] and the papers cited above.

Greedoids are an interesting area of research which I would love the oportunity to fully investigate. However this is just the most basic characterisation of greedoids and some appropriate axioms which at least allow us to find a subset of the possible greedoids and set-systems that yield optimal soltuions by way of the greedy algorithm. Thank you for reading!

7 Implementation

This final section describes my work concerning implementing the optimisation procedures described throughout this paper. I provide any of the auxillary algorithms mentioned or necessary to implement for example Kruskal's algorithm or to adapt your own form of the generic greedy algorithm. Any code used during this project can be found at my github repository for this project [11], including an almost complete implementation of Kruskal's Algorithm.

7.1 Depth-first Search

The following search algorithm is how we will determine the connected components of a disconnected graph. We do this by selecting an arbitrary vertex of the graph as the "root". And then exploring along the branches from the vertex as far as possible until we need to backtrack, at which point we choose a new undiscovered arbitrary root and repeat. Labelling each vertex discovered along that exploration as discovered. The aim is to discover all the vertices of the graph.

Algorithm 5 DFS

Let G be a graph with vertex set $V = \{1, ..., n\}$

```
1: procedure DFS($G, V$)
2:     label $v$ as discovered
3:     for all edges from $v$ to $w$ in $G$.adjacentEdges($V$) do
4:         if (vertex w is not labelled as discovered) then
5:             recursively call DFS($G, w$)
```

This algorithm will allow you to find the connected components of a disconnected graph. Then using the following algorithm we can check if our forest at each step of our algorithm is acyclic. We do this by counting the number of edges that each component has since we know a tree can have at most $n - 1$ edges by *lemma 3.7.* This procedure is invoked in *algorithm 3.*

Algorithm 6 Acyclic Check

Let G be a graph with the set of connected components C as found by DFS(G,v) where v is an arbitrary vertex in G.

```
1: procedure ACYCLIC($G, C$)
2:     for all $i$ in $C$ do
3:         if $i$.edgeCount() $> n - 1$ then return False
       return True
```

The following algorithm is useful towards implementing the greedy algorithm as described in *section 4* due to the fact that the elements of the edge set of a graph G must be sorted in terms of their weights. The following process is a straightforward method to arrange your elements in the desired order.

Algorithm 7 Insertion Sort

Let A be an unsorted list containing real number positive values (to match our weight functions).

```
1: procedure INSERTION($A$)
2:     $i \leftarrow 1$
3:     while $i <$ length($A$) do
4:         $j \leftarrow i$
5:         while $j > 0$ and $A[j-1] > A[j]$ do
6:             Swap $A[j]$ and $A[j-1]$
7:             $j \leftarrow j - 1$
8:         $i \leftarrow i + 1$
```

This is a Perl5 implementation of an insertion sort I wrote, for sufficiently small lists this is a suitable sorting algorithm. For larger graphs, a more sophisticated algorithm such as quicksort might be more appropriate.

```perl
use strict;
use warnings;
use Data::Alias;
sub InsertionSort{
    my $i = 1;
    while($i < scalar @li){
        my $j = $i;
        while($j>0 && $li[$j-1] > $li[$j]){
            alias @li[$j,$j-1] = @li[$j-1,$j];
            $j = $j-1;
        }
        $i = $i+1;
    }
}
InsertionSort(@li);
```

In terms of storing the graphs and weights while coding, the method I used in my implementation of Kruskal's algorithm involved use of Perl5's hash structure. A perl has is simply a list defined by key-values pairs, also known as an associative array. The code snippets below demonstrate how weights are associated with the edges, how the graph is initialised and how the edges are associated with a particular vertex.

```perl
use strict;
use warnings;
use Data::Dumper qw(Dumper);
use Tie::IxHash;
my @V = ('A', 'B', 'C', 'D');
my @E = ('AB', 'AC', 'AD', 'BD', 'BC', 'CD');

my %WEIGHTS;
tie %WEIGHTS, 'Tie::IxHash';
foreach my $i (@E){
    my $range = $MAX_WEIGHT;
    my $random_number = int(rand($range));
    $WEIGHTS{$i} = $random_number;
}
print Dumper(\%WEIGHTS);
```

```perl
my %FOREST;
tie %FOREST, 'Tie::IxHash';
foreach my $j (@V){
    $FOREST{$j} = [];
}
print Dumper(\%FOREST);
```

```perl
my $temp_edge = 'AB';
sub MatchVertexEdge{
    my $ver_edge = $temp_edge;
    my @edge_components = split(//, $ver_edge);
    print @edge_components;
    foreach my $vertex (@edge_components){
        push @{$FOREST{$vertex}},
    }
}
MatchVertexEdge($temp_edge);
```

With all this it should be possible to implement these processes in almost any language. The full implementation can be viewed at the repository[11] and any improvements are welcome as pull requests.

References

[1] Oxley, James (1992), Matroid Theory, Oxford: Oxford University Press, ISBN 0-19-853563-5, MR 1207587, Zbl 0784.05002.

[2] James Oxley : What is a matroid? https://www.math.lsu.edu/ oxley/survey4.pdf

[3] Jungnickel, Dieter (2013), Graphs,Networks and Algorithms, Springer-Verlag Berlin Heidelberg, ISBN 978-3-642-32277-8

[4] Whitney, Hassler. "On the Abstract Properties of Linear Dependence." American Journal of Mathematics, vol. 57, no. 3, 1935, pp. 509–533. JSTOR, JSTOR, www.jstor.org/stable/2371182.

[5] Kruskal, Joseph B., Jr. On the shortest spanning subtree of a graph and the traveling salesman problem. Proc. Amer. Math. Soc. 7 (1956), 48–50.

[6] Otakar Borůvka, On a minimal problem, Práce Moravské Pridovedecké Spolecnosti, vol. 3, 1926.

[7] Dominic J.A. Welsh, A bound for the number of matroids, Journal of Combinatorial Theory, Volume 6, Issue 3, 1969, Pages 313-316, ISSN 0021-9800, https://doi.org/10.1016/S0021-9800(69)80094-3. (http://www.sciencedirect.com/science/article/pii/ S0021980069800943)

[8] Bjørner, A., Ziegler, G.M.: Introduction to greedoids. In: White, N.(ed) Matroid Applications, pp. 284–357. Cambridge University Press, Cambridge (1992)

[9] Korte and Lovasz (1984) Korte, B., Lovasz, L.: Greedoids and linear objective functions. SIAM J. Algebr. Discr. Math.5, 229– 238 (1984)

[10] Bryant, V., Brooksbank, P.: Greedy algorithm compatibility and heavy-set structures. Europ. J. Comb. 13 , 81–86 (1992)

[11] https://github.com/emcd123/Matroids

[12] https://www.wikipedia.org/