Project Report

On

MATROIDS & MINIMUM SPANNING TREE

Submitted

in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

MATHEMATICS

by

KRISHNAVENI M

(Register No. SM20MAT009)

(2020-2022)

Under the Supervision of

DHANALAKSHMI O M

DEPARTMENT OF MATHEMATICS

ST. TERESA'S COLLEGE (AUTONOMOUS)

ERNAKULAM, KOCHI - 682011

APRIL 2022

ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM

THE FEAR OF THE LORD IS THE BEGINNING OF WISDOM

ST. TERESA'S COLLEGE (AUTONOMOUS) ESTD 1925

CERTIFICATE

This is to certify that the dissertation entitled, **MATROIDS AND MINIMUM SPANNING TREE** is a bonafide record of the work done by Ms. **KRISHNAVENI M** under my guidance as partial fulfillment of the award of the degree of **Master of Science in Mathematics** at St. Teresa's College (Autonomous), Ernakulam affiliated to Mahatma Gandhi University, Kottayam. No part of this work has been submitted for any other degree elsewhere.

Date: 27.05.2022
Place: Ernakulam

[illegible]

Dhanalakshmi O M
Assistant Professor,
Department of Mathematics,
St. Teresa's College(Autonomous),
Ernakulam.

ST. TERESA'S COLLEGE
DEPT. OF MATHEMATICS
ERNAKULAM

[illegible]

Dr.Ursala Paul
Assistant Professor & HOD,
Department of Mathematics,
St. Teresa's College(Autonomous),
Ernakulam.

External Examiners

1: DIVYA MARY DAISE.S
[illegible]
27/05/2022.

2: MARY RUNIYA
[illegible]
27/05/2022

DECLARATION

I hereby declare that the work presented in this project is based on the original work done by me under the guidance of **DHANALAKSHMI O M**, Assistant Professor, Department of Mathematics, St. Teresa's College(Autonomous), Ernakulam and has not been included in any other project submitted previously for the award of any degree.

Krishnaveni.

Ernakulam. **KRISHNAVENI M**

Date: 27.05.2022 **SM20MAT009**

ACKNOWLEDGEMENTS

I express my deep sense of gratitude to Dhanalakshmi O M, Assistant Professor of department of Mathematics, St.Teresa's College, Ernakulam for her valuable guidance, constant suggestion and encouragement throughout my course of the project.

I am also extremely grateful to our HOD Dr.Ursala Paul for her continuous support and all the teachers of our department who provided us with the necessities for the completion of the project.

In addition, very energetic and competitive atmosphere of the Department had much to do with this work. Above all, I thank God almighty and my parents for giving me the blessings to take over this project.

Krishnaveni

Ernakulam.

KRISHNAVENI M

Date: 27.05.2022

SM20MAT009

ABSTRACT

The purpose of studying Matroids is to provide a unifying abstract treatment of independence in Graph Theory and Linear Algebra. The basic concepts and properties of Matroids and Minimum Spanning Tree have been studied in this article. Examined, how we can obtain a Minimum Spanning Tree using the most powerful optimization technique, Greedy Algorithm. Thus concluded how we can obtain the Minimum Spanning Tree using Matroid Algorithm
i.e. Kruskal's Algorithm in an edge weighted graph, and the analysis of a problem to find a Minimum Spanning Tree by applying Kruskal's Python Code helped to find the optimal solution.

Keywords: Matroids, Independent set, Base, Minimum Spanning Tree, Greedy Algorithm

PRELIMINARY

Graph – A graph G is an ordered triplet G=(V(G),E(G),I) where V(G) is a non-empty set, E(G) is a set disjoint from V(G) and I is an incidence map that associates with each elements of E(G), an unordered pair of elements (same or distinct) of V(G). Elements of V(G) are called vertices and the elements of E(G) are called the edges of G.

End vertices – A vertex v is incident with an edge e if there is an edge at v. Two vertices are said to be end vertices if they are incident with the same edge.

Self-loop – An edge with same end vertex as end vertices is called self-loop.

Adjacent edges – Two edges are said to be adjacent edges if they are incident on a same vertex.

Adjacent vertices – Two vertices that are the end vertices of the same edge is said to be adjacent vertices.

Parallel Edges – Two or more edges are said to be parallel if they are incident with a same pair of vertices.

Simple Graph – A graph is simple if it does not have any self-loop or parallel edges.

Degree of a vertex – Degree of a vertex is defined as the number of edges incident on a vertex.

Walk – It is a finite alternating sequence of edges and vertices in such a way that it begins and end with a vertex and each edge have end vertices. Vertices can be repeated in a walk.
If a walk begins and ends at the same vertex then it is called a closed walk. A walk that is not closed is called open.

Trail – A walk in which no edge is repeated is called a trail.

Path – A trail in which no vertices and no edges are repeated is a path. As a path is a trail, it is also an open walk.

Cycle – A closed walk in which no edges and no vertices except the initial and final vertex appears more than once is known as a cycle.

Connected Graph – A graph in which if every pair of vertices has at least one path between them is known as connected.

Complete graph – A graph in which every pair of vertices is connected by an edge.

Vertex index - usually a vertex has a name associated with it. Internally, within the implementation of the graph in computer programs, for the computer to recognise it may be more convenient to refer a vertex using an integer number

SOFTWARE USED:

Python - Python is an Object -Oriented language that was developed in the late 1980s as a scripting langauge. Python maybe viewed as an emerging language, because it is still being developed and refined.
In its current state, it is an excellent language for developing engeneering applications. The Python programs are run by an interpreter and its great advantage is that programs can be tested and debuged quickly, allowing
the user to concentrate more on the principles behind the program and less on the programming itself.
Python program can be developed in much shorter time than equivalent C programs.

INTRODUCTION

Many results of graph theory extend or simplify in the theory of matroids. These include the greedy algorithm for minimum spanning trees, the strong duality between maximum and minimum vertex cover in bipartite graphs, and the geometric duality relating planar graphs and their duals.

Matroids arise in many contexts but are special enough to have rich combinatorial structure. When a result from graph theory generalizes to matroids, it can then be interpreted in other special cases. Several difficult theorems about graphs have found easier proofs using matroids.

Matroids were introduced by Whitney (1935) to study planarity and algebraic spaces of graphs, by MacLane (1936) to study geometric lattices, and by van der Waerden (1937) to study independence in vector spaces.

Contents

Chapter 1

INTRODUCTION TO MATROIDS

1.1 HISTORY

Originally and independently studied by Whitney & B. L. van der Waerden in the 1930's. Whitney began to see similarities between ideas in graph theory & linear independence & dimension in vector spaces. After recognizing the properties of independence, he decided to introduce the concept of Matroids. Eventually, we would see Matroids as a link between graph theory, linear algebra, and other fields.

1.2 INDEPENDENT SETS

The sets that avoid conflicts is called Independent Set in a graph. It is clear that, subsets of independent sets are independent and the empty set is independent. For e.g. Acyclic sets of edges. Let E be the edge set of a graph G, and let X $\leq$ E be "independent", if it contains no cycle.

Every subset of an independent set is independent and the empty set is independent. The cycles are the minimal Dependent sets. The Bases are the maximal independent sets and the Circuits are the minimal dependent sets; B_M & C_M denote these families of subsets of E. The Rank of a subset of E is the maximum size of an independent set in it. The rank function γ_M is defined by r(x) = max{$|Y|$: Y $\leq$ X, Y € I} A hereditary family is a collection of sets F, such that every subset of a set in F is also in F. A hereditary system M on E consists of a nonempty ideal I_m of subsets of E.

1.3 DEFINITION

A matroid M is a pair (E, I) consisting of a finite set E and a collection of subsets of E satisfying the following properties;
(i) I is non-empty
(ii)Every subset of every member of I is also in I
More significantly, I satisfies the following augmentation condition:
(iii) If X and Y are in I and $|X| = |Y| + 1$, then there is an element x in X $-$ Y such that YU{x} is in I.

1.4 EXAMPLE

Let E={1,2,3} & I = {$\emptyset$,{1},{2},{3} } we will show that (E,I) is a matroid.
1. I is non-empty since we have some element x in I, for example, x={1}
2. Since $\emptyset$ is in I, the subset of every other element is in I, namely {1}, {2}, {3}. Thus every subset of every member of I is also in I
3. Let X= {1}, then Y= $\emptyset$ since $|X| = |Y| + 1$. That gives us X- Y = {1}. Then, there is an element x such that Y U {x} is in I, namely x=1. We could show similarly with X = {2} & X= {3}. Therefore, this is a MATROID since it satisfies all three properties.

1.5 DIFFERENT TYPES OF MATROIDS

1.5.1 TRIVIAL MATROIDS

Given any non-empty finite set E, we can define on it a matroid whose only independent set is the empty set $\emptyset$. This matroid is the trivial matroid on E, and has rank 0.

1.5.2 DISCRETE MATROID

At the other extreme is the discrete matroid on E, in which every subset of E is independent. Note that the discrete matroid on E has only one base, E itself, and that the rank of any subset A is the number of elements in A.

1.5.3 CYCLE MATROID

Consider a graph G & the underlying set E is the set of edges E(G). A subset X $<$ E is independent if and only if does not contain any cycle of G. The rank function is given by r(X)=V(G)-K(X), where V(G) is the number of vertices of G & K(X)

is the number of connected components of the spanning sub-graph of G consisting of all the vertices of G and edges of X.

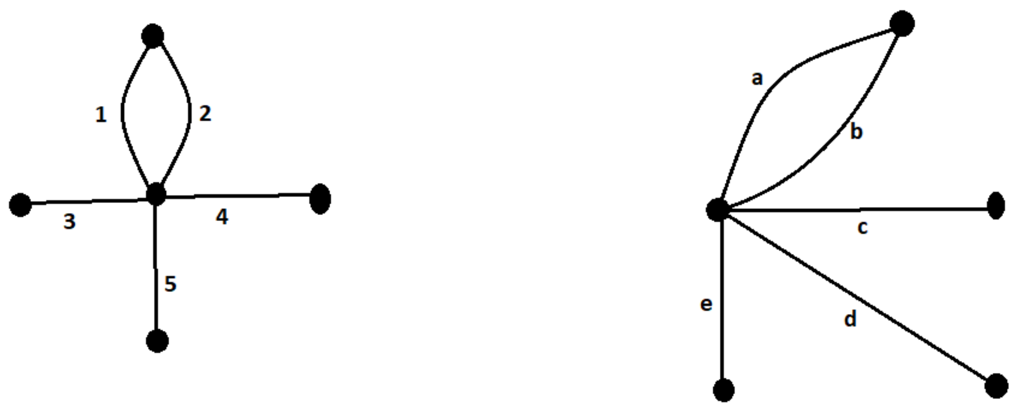
1.5.4 UNIFORM MATROID

The trivial matroid and discrete matroid are special cases of the k-uniform matroid on E, whose bases are those subsets of E with exactly k-elements. The trivial matroid on E is 0-uniform and the discrete matroid is $|E|$-uniform. Note that the independent sets are those subsets of E with not more than k elements, and the rank of any subset A is either $|A|$ or k, whichever is smaller.
For example; consider the complete graph k_3, then $C(k_3) = U_{2,3}$ where $C(k_3)$ is the cycle matroid and $U_{2,3}$ is the uniform matroid

ISOMORPHISM

Here we define two matroids M_1 & M_2 to be isomorphic if there is a one-one correspondence between their underlying sets E_1 & E_2 that preserves independence. Thus, a set of elements of E_1 is independent in M_1 if & only if the corresponding set of elements of E_2 is independent in M_2.

FIG 1.1

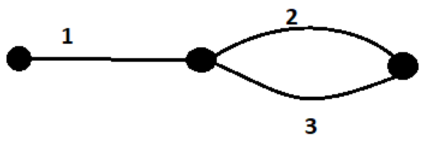


Note that, although matroid isomorphism preserves cycles, cut-sets and the number of edges in a graph, it does not necessarily preserve connectedness, the number of vertices, or their degrees.

1.5.5 GRAPHIC MATROID

Let G be a graph and M is a matroid of G and C(G) is the cycle matroid. If M is isomorphic to C(G) then such matroids are called Graphic Matroids.
For example; the matroid M on the set {1, 2, 3} whose bases are {1, 2} and {1, 3} is a graphic matroid isomorphic to the cycle matroid of the graph G.

FIG 1.2



1.5.6 COGRAPHIC MATROIDS

G is a graph, the cycle matroid C(G) is not the only matroid that can be defined on the set of edges of G. Because of this similarity between the properties of cycles and of cut-sets in a graph, we can construct a matroid by taking the cut-sets of G as cycles of the matroid. A matroid M is said to be Cographic if M is isomorphic to M*(G), where M*(G) is the cut-set matroid.

1.5.7 TRANSVERSAL MATROID

This matroid gives us a link between matroids and transversal theory. If E is a non-empty finite set and if F=(F_1, F_2,. . . . . . ., F_m) is a family of non-empty subsets of E, then the partial transversal of F can be taken as the independent sets of a matroid on E, denoted by M(F) or M(F_1, F_2,. . . . . .,F_m). The matroid obtained in this way is called Transversal matroid.
For example; the graphic matroid M in the previous example is a transversal matroid on the set {1, 2, 3}. Since its independent sets are the partial transversal of the family F=(F_1, F_2), where F_1={1} & F_2={2,3}.

Chapter 2

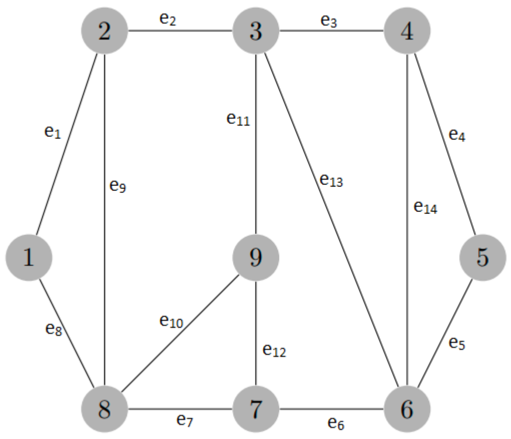
PROPERTIES OF MATROIDS

2.1 INTRODUCTION

In this chapter we will see that a hereditary system is a matroid by verifying any of the property discussed here. We will also discuss how the powerful tool - the greedy algorithm which yields the optimal independent set and the concept of graph theory & independence.

2.2 PROPERTIES

Consider the graph G as given below.
FIG 2.1

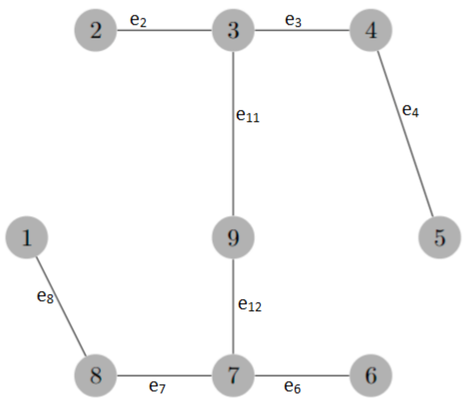


1. **A Base is a maximal independent set.**

For, X= {e_2, e_3, e_4, e_7, e_8, e_6, e_{11}, e_{12} } It is clear that, these edges form a spanning tree of the graph. Hence a maximal independent sets of a graph is its spanning tree (For edge weighted graph is its minimum spanning tree). Hence it is termed as a

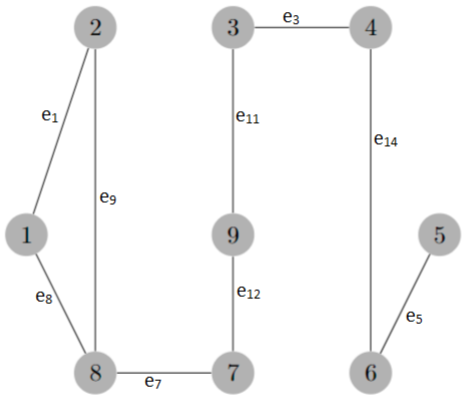
Base.

FIG 2.1(a)



2. A Circuit is a minimal dependent set.

For, X= {e_1, e_3, e_5, e_7, e_8, e_9, e_{11}, e_{12}, e_{14} } It is clear that, these edges form a cycle in the graph. Hence it is minimal dependent set. Hence it is termed as Circuit.



3. Circuits are the subsets X with r(X) = $|X|$-1.

Here X = {e_1, e_3, e_5, e_7, e_8, e_9, e_{11}, e_{12}, e_{14}},

Then r(X) = $|9|$ -1 = 8.

4. For an independent set X, the rank is equal to its cardinality r(X) =$|X|$.

(Rank is nothing but, it is the maximum size of an independent set of subset of finite set E)

If we consider the property 1

X = {e_2, e_3, e_4, e_6, e_7, e_8, e_{11}, e_{12}} Here the maximum size of this independent set is 8.

That is, r(X) = $|X|$ = 8.

5. All bases have the same cardinality which is called the rank of matroid, r(M). From FIG 2.1 we can consider some of the bases as
X = {e_2, e_3, e_4, e_6, e_7, e_8, e_{11}, e_{12}}
Y = {e_2, e_3, e_4, e_5, e_7, e_8, e_{11}, e_{12}}
Z = {e_1, e_5, e_9, e_6, e_{10}, e_{13}, e_{14}, e_{12}}
Here we see that all maximal independent sets have same cardinality, which is rank of matroid r(M)

2.3 DEFINITION

A hereditary system M on E is a Matroid if it satisfies the following additional properties;
where I is the Independent sets, B is the Bases, C is the Circuits and r is the rank function of M.

(i) Augmentation – if I_1 and $I_2 \in$ I with $|I_2| > |I_1|$, then I_1 + e $\in$ I for some e $\in$ I_2-I_1.
(ii) Uniformity – for every X $\subseteq$ E, the maximal subsets of X belonging to I have the same size.
(iii) Base Exchange – if B_1 and $B_2 \in$ B, then for all e $\in$ B_1-B_2 there exists f $\in$ B_2-B_1 such that B_1-e+f $\in$ B.
(iv) Submodularity – for X, Y $\subseteq$ E, r(X$\cap$Y) + r(X$\cup$Y) $\leq$ r(X) + r(Y).
(v) Weak absorption – r(X) =r(X +e) = r(X +f) implies r(X +e + f) = r(X) whenever X $\subseteq$ E & e, f $\in$ E.
(vi) Strong absorption – if X, Y $\subseteq$ E & r(X +e) = r(X) $\forall$ e $\in$ Y, then r(X $\cup$ Y) = r(X).
(vii) Weak elimination – for c_1, c_2 ($c_1 \neq c_2$) and x $\in$ $c_1 \cap c_2$ there is another element of C contained in ($c_1 \cup c_2$) – x.
(viii) Induced circuits – if H $\in$ I, then H +e contains at most one circuit.
(ix) Greedy algorithm – for each non-negative weight function on E, the greedy algorithm selects an independent set of maximum total weight.

2.4 THEOREM

For a hereditary system M, the conditions defining matroids in Definition 2.3 are equivalent.

PROOF -

a) We have to, prove that Uniformity implies Base Exchange. From uniformity we can take X=E, and the cardinality of all bases are same. Now we consider the set (B_1-e) $\cup$ B_2, where B_1 & B_2 are bases & e is an edge. Apply the uniformity property to the set (B_1-e) $\cup$ B_2. Since all bases are independent sets and consider (B_1-e) $\cup$ B_2=Y. Then this gives us the augmentation property of the independent set B_1-e from B_2, that implies $|B_2| > |B_1|$.

b) We have to, prove that Base Exchange implies augmentation.
Consider two independent sets I_1 and I_2 from I, with $|I_2| > |I_1|$. Now take two bases B_1, $B_2 \in$ B such that $I_1 \subseteq B_1$ & $I_2 \subseteq B_2$. Consider the set $B_1 - I_1$, apply the Base Exchange property to this set. It will replace elements of $B_1 - I_1$ outside B_2 with the elements of B_2.
Assume that B_1-$I_1 \subseteq B_2$. But if, $B_1 - I_1 \subseteq B_2 - I_2$ then $|B_1| < |B_2|$ Therefore it is a contradiction, since the Base Exchange property conveys that all bases have same cardinality.
If $|B_1| < |B_2|$ where B_1, $B_2 \in$ B, then we can replace elements in B_1-B_2 by B_2-B_1 step by step, which will obtain a base of cardinality $B_1 \subseteq B_2$. But no base is contained in another. Hence I_2 has an element contained in B_1-I_1. Therefore, we will use that element to augment I_1.

c) We have to, prove that Augmentation implies weak absorption.
We prove this by contrapositive method. Let us assume that
r(X) = r(X +e) = r(X +f) and r(X +e + f) = r(X).
Consider I_1 and I_2 be the maximum independent subsets of X and of X +e + f. If $|I_2| > |I_1|$ then we can expand I_1 from I_2.
Then augmentation can only add e or f. since I_1 is a maximum independent subset of X. Therefore it is a contradiction to our assumption that r(X) = r(X +e) = r(X +f).

d) We have to, prove that weak absorption implies strong absorption.
Let X, Y $\subseteq$ E. consider $|Y - X|$. We proceed the proof by mathematical induction on $|Y - X|$. When $|Y - X| = 1$, r(X) = r(X +e) implies, r(X $\cup$ Y) = r(X). Hence the result is true for $|Y - X| = 1$. Assume Y' = Y − e- f, for e, f $\in$ Y-X.
Now by our induction hypothesis, on the proper subset of Y r(X) = r(X $\cup$ Y' +e) = r(X $\cup$ Y' +f) = r(X $\cup$ Y'). Hence r(X) = r(X $\cup$ Y), i.e. weak absorption yields

strong absorption.

e) We have to, prove that strong absorption implies uniformity.
Let Y $\subseteq$ X, i.e. Y is a maximal independent subset. r(Y +e) = r(Y) where e $\in$ X-Y. Applying strong absorption property implies that r(X) = r(Y) = $|Y|$.
Since Y is arbitrary, all maximal independent subset Y have same cardinality.

f) We have to, prove that uniformity implies submodularity.
For submodularity we have to consider two maximal independent subsets X, Y $\subseteq$ E. Consider X $\cap$ Y, now choose a maximal independent set I$_1$ $\in$ X $\cap$ Y.
Now applying the uniformity property to the set I$_1$ we can augment another maximal independent subset of X $\cup$ Y, let it be I$_2$.
Take I$_2$ $\cap$ X & I$_2$ $\cap$ Y then both includes I$_1$ also & they are independent subsets of X and Y.
Hence, r(X $\cap$ Y) + r(X $\cup$ Y) = $|I_1|$ + $|I_2|$ = $|I_2\cap$ X $|$ + $|I_2\cap$ Y $|$ $\leq$ r(X) + r(Y) I.e. r(X $\cap$ Y) + r(X $\cup$ Y) $\leq$ r(X) + r(Y).
Hence we get the submodularity property.

g) We have to, prove that submodualrity implies weak elimination.
Now we can consider the circuits C$_1$ and C$_2$ $\in$ C & x $\in$ C$_1$ $\cap$ C$_2$ We know that r(C$_1$)=$|C_1|$-1 and r(C$_2$)=$|C_2|$-1
It is known that every proper subset of a circuit is independent,
r(C$_1$ $\cap$ C$_2$) = $|C_1\cap$ C$_2|$
Assume $(C_1\cup$ C$_2) - x$ does not contain a circuit,
then we have r$(C_1\cup$ C$_2)$–x =$|C_1\cup$ C$_2|$ -1.
Now applying the submodularity property toC_1 & C$_2$ implies that
$|C_1\cap$ C$_2|$ + $|C_1\cup$ C$_2|$ -1 $\leq$ $|C_1|$ + $|C_2|$ -2 . Which is a contradiction, to our assumption.
Hence our assumption is wrong, therefore submodularity implies weak elimination.

h) We have to prove that weak elimination implies induced circuits. Suppose C$_1$, C$_2$ $\in$ H+ e (C$_1$, C$_2$ $\in$ C) for some H $\in$ I then e belongs to both C$_1$ and C$_2$. Now applying weak elimination we get a circuit contained in (C$_1$ $\cup$ C$_2$) $-$ e. but we know that (C$_1$ $\cup$ C$_2$) $-$ e is independent which is contained in H.

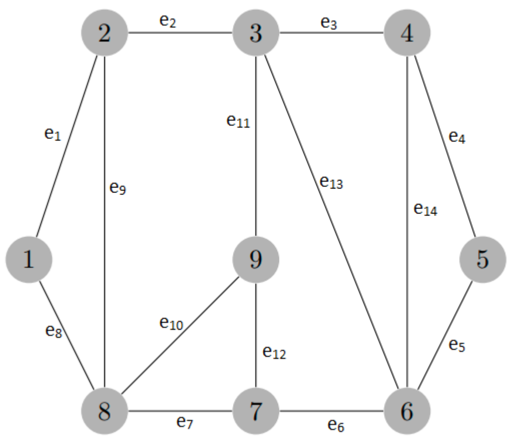
i) We have to prove that induced circuits implies greedy algorithm.
Let w be the weight function. Let the output of the greedy algorithm be H.
Take H* be a maximum weight independent set such that, H $\cap$ H* is the largest one. Since H is the output of the greedy algorithm, then H $\not\subseteq$ H* is the largest one. Assume H $\neq$ H*. Let a $\in$ H-H*, such that a be the first element chosen by the

greedy algorithm. Since H* is a maximum weight independent set, then H*+e is dependent therefore it has a unique circuit C. We know that H does not contains C. Choose b such that b $\in$ C-H & H*+e has at most one circuit, H*+a-b $\in$ I. Therefore the optimum of H* obtains w(b) $\geq$ w(a) ——(1)
Since b and the elements of H chosen earlier than a all contained in H*, b does not complete a circuit with them. Therefore b was available when the algorithm selected a, which implies w(b) $\leq$ w(a) ——(2)
From (1) and (2) it implies that w(b) = w(a) and w(H*+a-b)=w(H*)
But $|(H^* + a - b)\cap$ H $| \geq |H^*\cap$ H $|$, this is a contradiction to our assumption, that is the choice of H*. Hence H=H*.

j) We have to prove that greedy algorithm implies augmentation.
Consider two elements of I, I_1 & I_2 such that $|I_1| < |I_2|$.
We represent a weight function for which the iterative steps of the greedy algorithm obtains the desired augmentation.
Consider $|I_1|$ = m. Let w(a) = m+2 for a $\in$ I_1 Let w(b) = m+1 for b $\in$ I_2 - I_1 and let w(e) =0 for e $\notin$ $I_1 \cup I_2$.
Implies w(I_2) (m+1)2 > m(m+2) = w(I_1) Therefore I_1 is not a maximum weighted independent set. But the algorithm chooses all elements of I_1 and then chooses the element of I_2-I_1.
Since the greedy algorithm chooses a maximum weighted independent set, the iteration continues after taking in I_1 and adds an element e $\in$ I_2-I_1 such that I_1+e $\in$ I. Hence the greedy algorithm implies augmentation property.

2.5 GRAPH THEORY & INDEPENDENCE

Here we focuses only on connected graphs. There are two common ways to define independence in a graph, on the vertices or on the edges. We focus on the edges. What might it mean for a set of edges to be independent?
When would edges be necessary in a connected graph? Edges exists to connect vertices. Put another way, edges are how we move from vertex to vertex in a graph. So some set of edges should be considered independent if, for each edge the removal of that edge makes some vertices inaccessible to a previously accessible.
Consider the graph G in figure 2.6 with edge set E = {e_1, e_2,. . . . . . ,e_{14}}
Fig



Now consider the subset of edges X= {e_1, e_3, e_5, e_7, e_8, e_9, e_{11}, e_{12}, e_{14}} is this an independent set of edges? No because, the same set of vertices are connected to one another even if, for example, edges e_9 were removed from S. But here the set X contains a cycle.

Any time some set of edges contains a cycle, it cannot be an independent set of edges. In any connected graph, a set of edges forming a tree or forest (an acyclic sub graph) is independent. This makes sense in two different ways; first, a tree or forest never contains a cycle; second, the removal of any edge from a tree or forest disconnects some vertices from one another, decreasing accessibility, and so every edge is necessary.

A maximal such set is a set of edges containing no cycles, which also makes all vertices accessible to one another. This is called a Spanning Tree. There must be at least one spanning tree for a connected graph.

Here is the set T of some spanning trees for G:

T = { {e_1, e_7, e_8, e_{11}, e_{12}, e_3, e_4, e_5}, {e_1, e_3, e_2, e_4, e_6, e_8, e_{10}, e_{12}}, {e_1, e_2, e_{11}, e_{12}, e_7, e_6, e_{14}, e_5}, {e_1, e_2, e_{11}, e_{10}, e_7, e_{13}, e_4, e_5}, {e_1, e_5, e_6, e_9, e_{10}, e_{12}, e_{13}, e_{14}}, {e_1, e_2, e_3, e_5, e_6, e_7, e_{11}, e_{12}}, {e_1, e_9, e_{10}, e_7, e_{11}, e_{13}, e_3, e_4}, {e_2, e_{11}, e_{10}, e_8, e_{13}, e_6, e_{14}, e_4}, {e_2, e_3, e_4, e_5, e_7, e_8, e_{11}, e_{12}}, {e_2, e_3, e_4, e_6, e_8, e_7, e_{11}, e_{12}} }

Here again we see that all maximal independent sets must have the same size.

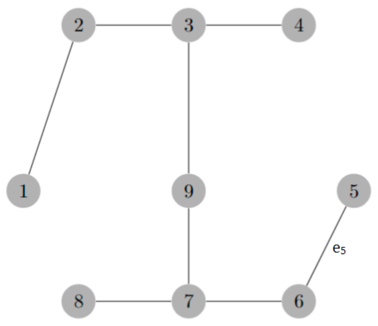
Spanning trees also have two other important traits:

a) No spanning tree properly contains another spanning tree.

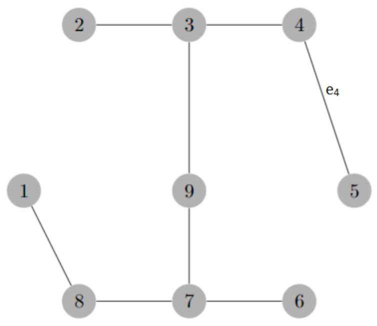
b) Given two spanning trees T_1 and T_2 and an edge e from T_1 we can always find some edge f from T_2 such that {T_1-e} ∪ f will also be a spanning tree.

Now we can demonstrate the second condition.

Fig

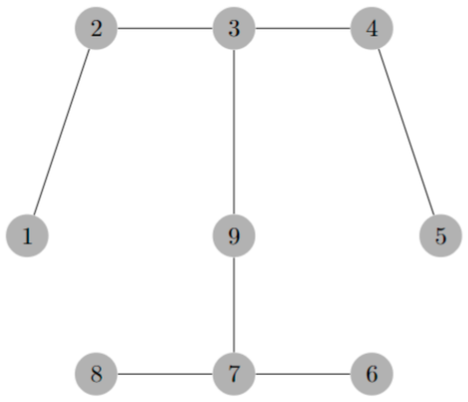


T_1



T_2

$(T_1 - e_5) \cup e_4$



T_3

Chapter 3

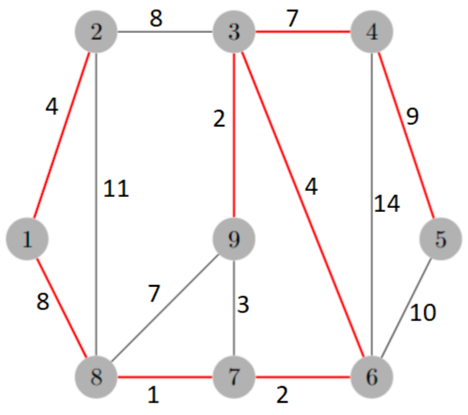
MINIMUM SPANNING TREE

3.1 INTRODUCTION

Minimum spanning trees were first considered by Boruvka shortly after 1920, electricity was to be supplied to the rural area of Southern Moravia; the problem of finding as economical a solution as possible for the proposed network was presented to Boruvka.
He found an algorithm for constructing a minimum spanning tree namely Boruvka's Algorithm. But later, Kruskal and Prim also founded algorithms for constructing a minimum spanning tree.
There are also references to various applications reaching from the obvious examples of constructing traffic, communication networks, image processing.

3.2 DEFINITION

A Spanning tree of a graph G is a sub-graph T that is:
a) Connected: A graph is connected when it has at least one vertex and there is a path between every pair of vertices.
b) Acyclic: containing no cycles.
c) Contains all the vertices.
Hence a Minimum Spanning Tree is a subset of the edges of a connected, edge-weighted un-directed graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight. That is, it is a spanning tree whose sum of edge weights is as small as possible.



The above graph contains 9 vertices and 14 edges. The numbers adjacent to each edge is its weight. The edges in red color represent the minimum spanning tree of this graph.

3.3 MINIMUM SPANNING TREE-PROBLEM

The minimum spanning tree problem (MSTP) is an exceeding popular problem of combinatorial optimization. Methods for its solution have generated important ideas of modern combinatorics and have played a crucial role in the design of Computer Algorithms.

A typical MSTP is stated as below,

Given a weighted graph G whose nodes represent cities, whose edges represent possible communication links and edge weights represent the cost of construction or length of edge/link. One would then wish to construct a communication network that connects all cities to have minimum cost or of minimum total length.

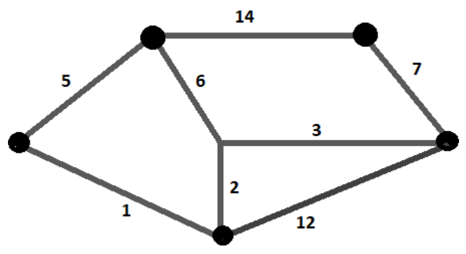
The importance of MSTP step from several facts

1) Gives an efficient solution, which makes it practical to optimize truly huge graphs of thousands of vertices.

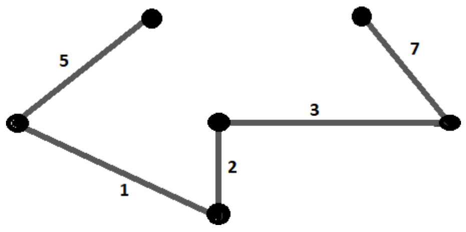
2) Obvious and direct application in design of computed and communication networks.

3) Indirect applications in other problems such as, Network Reliability, Image Segmentation, Travelling Salesman Problem.

EXAMPLE: Find the minimum spanning tree of the graph given below.



The graph has 8 vertices. Therefore the minimum spanning tree has only 5 edges. The minimum spanning tree is as given below.



3.4 PROPERTIES

1. Possible multiplicity: If there are n vertices in the graph, then each spanning tree has n-1 edges.

Proof: First we will prove that if T is a tree with n vertices then it has precisely n-1 edges. We use induction on n when n=1 i.e. T has only one vertex, then since it has no loops, T cannot have any edge. That is, it has n-1=0 edges. This establishes that the result is true for n=1.

Now we suppose that the result is true for n=k, where k is some positive integer. We use this to show that the result is true for n=k+1.

Let T be a tree with k+1 vertices and let n be a vertex of degree 1 in T. Let e = uv denote the unique edge of T which has u as an end. Then if x and y are vertices in T both different from u, any path P joining x to y does not go through the vertex u.

Since if it did it would involve the edge e twice. Then the subgraph T-u obtained from T by deleting the vertex u is connected.

Moreover if C is a cycle in T-u then C would be a cycle in T-impossible. Since T is a tree, Thus the subgraph T-u is acyclic. Hence T-u is a tree.

However T-u has k vertices and so by our induction assumption, T-u has k-1 edges. Since T-u has exactly 1 edge less than T. It follows that T has k edges, as required. In other words, assuming the result is true for k, we have shown that it is true for k+1. Thus by the principle of mathematical induction, it is true for all positive integers k.

A minimum spanning tree of a graph G is a spanning subgraph of G that is a true. Therefore, a minimum spanning tree with n vertices has precisely n-1 edges.

2.Uniqueness

If each edge have a distinct weight then there will be only one, unique minimum spanning tree. If any two edges have same weight then the MST need not be unique.

Proof:

1. Assume the contrary, that there are two different MSTs A and B.

2. Since A and B differ despite containing the same nodes, there is at least one edge that belongs to one but not the other. Among such edges, let e_1 be the one with least weight; this choice is unique because the edge weights are all distinct. Without loss of generality, assume e_1 is in A.

3. As B is an MST, $e_1 \cup B$ must contain a cycle C with e_1.

4. As a tree, A contains no cycles, therefore C must have an edge e_2 that is not in A.

5. Since e_1 was chosen as the unique lowest-weight edge among those belonging to exactly one of A and B, the weight of e_2 must be greater than the weight of e_1.

As e_1 and e_2 are part of the cycle C, replacing e_2 with e_1 in B. Therefore, yields a spanning tree with a smaller weight. This contradicts the assumption that B is a MST.

Hence the proof.

3. Minimum Cost Subgraph

Minimum spanning tree is a minimum cost subgraph. i.e. the total cost of the MST is the least compared to all other subgraph of the original graph.

4. Cycle Property

For any cycle C in the graph, if the weight of an edge e of C is larger than the individual weights of all other edges of C, then this edge cannot belong to an MST.

Proof: Assume the contrary, i.e. that e belongs to an MST T_1. Then deleting e will break T_1 into two sub trees with the two ends of e in different subtrees.

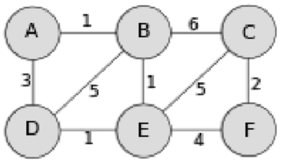
The remainder of C reconnects the subtrees, hence there is an edge f of C with ends in different subtrees, i.e., it reconnects the subtrees into a tree T_2 with weight less than that of T_1, because the weight of f is less than the weight of e.

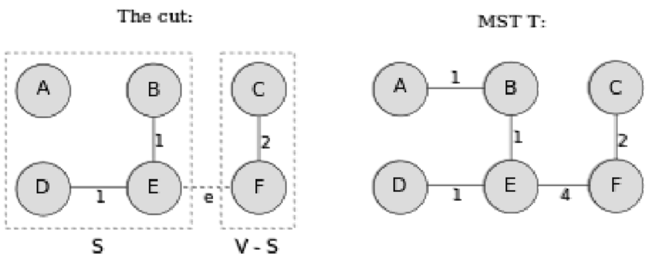
5. Minimum Cost Edge

If the minimum cost edge e of a graph is unique, then this edge is included in any MST.

Proof: If e was not included in the MST, removing any of the (larger cost) edges in the cycle formed after adding e to the MST would yield a spanning tree of smaller weight.

6. Cut Property





This figure shows the cut property of MSTs. T is the only MST of the given graph. If S = {A,B,D,E}, thus V-S = {C,F}, then there are 3 possibilities of the edge across the cut(S,V-S), they are edges BC, EC, EF of the original graph. Then, e is one of the minimum-weight-edge for the cut, therefore S ∪ {e} is part of the MST T.

For any cut C of the graph, if the weight of an edge e in the cut- set of C is strictly smaller than the weights of all other edges of the cut-set of C, then this edge belongs to all MSTs of the graph.

Proof:

Assume that there is an MST T that does not contain e. Adding e to T will produce a cycle, that crosses the cut once at e and crosses back at another edge e'. Deleting e' we get a spanning tree T-{e'} ∪ {e} of strictly smaller weight than T. This contradicts the assumption that T was a MST. Therefore the MST T contains the edge e'.

Chapter 4

APPLICATIONS

4.1 INTRODUCTION

In this chapter, we consider their applications. Beyond unifying distinct areas of discrete mathematics, matroids are essential in combinatorial optimization. The greedy algorithm, a powerful optimization technique, can be recognized as a matroid optimization technique. In fact, the greedy algorithm guarantees an optimal solution only if the fundamental structure is a matroid.

4.2 GREEDY ALGORITHM

4.2.1 DEFINITION

A Weighted Matroid is (E, I, w) where w: E $\to$ R the weight of any A $\in$ I is given by Σ x $\in$ A w(x). Next, we would like to compute a maximum weight maximal independent set.

4.2.2 ALGORITHM

```
1. Sort the elements of E in non-increasing order of their weights.
2. A=$\emptyset$
3. For i=1 to $|E|$ do
If (A $\cup$ $x_i$ $\in$ I)
Then A = A $\cup$ $x_i$;
4. Return A.
```

4.2.3 THEOREM

The solution obtained from the greedy algorithm is optimal.

Proof –

To prove this theorem we will prove some claims.

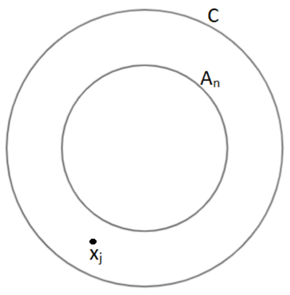
Claim- 1 An is a maximal independent set, where An is the nth iterative set.

For, suppose A_n is not maximal. Then there exists C $\in$ I such that A_n C. then there must be some element $x_j \in$ C-A_n. Let B= {$x_i \in A_n$ / i$\leq$j } $\subset$ A. From our notation B=A_j-1. Then in the jth iteration we must test whether A_j-1 $\cup$ {x_i} $\in$ I. since $x_j \in C - A_n$, x_j A_n,

hence we must have found that A_j-1 $\cup \{x_i\} \notin I.ButA_j - 1\cup$ {x_i} $\subset$ C. From the third condition of matroid A_j-1 $\cup$ {x_i} $\in$ I.

That is a contradiction to our assumption. Hence our assumption is wrong. A_n must be itself a maximal independent set.

Fig (claim 1)



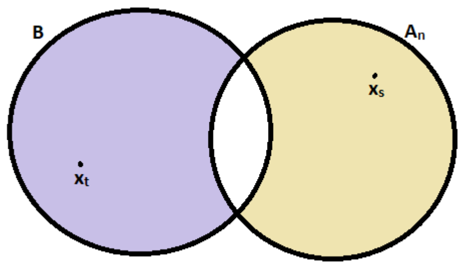
Claim – 2

Let B($\neq A_n$) be a maximal independent set and s be the smallest index in A_n – B and t be the smallest index in B – A_n. Then s<t.

For, assume that t<s. Let C = {$x_i \in A_n$ / i<t}. Again observe that C=A_t-1 and C $\subset$ B. We must have found that A_t -1 $\cup$ {x_t} $\notin$ I. But A_t-1 $\cup$ {x_t} $\subseteq$ B.

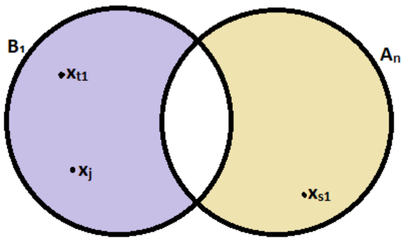
From the third condition of matroid A_t-1 $\cup$ {x_t} $\in$ I. Therefore it is a contradiction. Hence our assumption is wrong. Hence s<t.

Fig (claim 2)

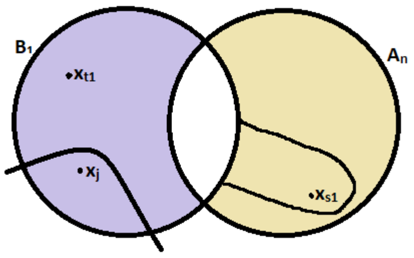


Claim – 3

A_n is the maximum weight maximal independent set.
For, assume that A_n is not a maximum weight maximal independent set. Suppose $B_1, B_2, B_3, \ldots$ B_r be maximum weight maximal independent set where $r \geq 1$. Corresponding to each B_i let us consider the inequality $s_i < t_i$, from claim 2 it is clear that $s_1 \in A_n - B_1$ and $t_1 \in B_1 - A_n$ (where s_i & t_i being the smallest index among them). Without loss of generality assume that s_1 is the largest index.
Fig



From condition two and three of matroid B'= (B $\cup$ $\{x_{s1}\}$)$-\{x_j\}$ $\in$ I. Since $j \geq t_1 > s_1$ and the weights are in non-increasing order, so $w(x_j) \leq w(x_{s1})$ Therefore wt(B') $\geq$ wt(B). since B is a maximum weight maximal independent set so B' is also a maximum weight maximal independent set.
Fig



Since A_n $-$ B' $=$ A_n $-$ B $-$ $\{x_s\}$. The smallest index in A_n $-$ B' must be greater than s_1. Let B'=B_p. Therefore $s_p > s_1$ i.e. a contradiction. Hence our assumption is wrong. Hence A_n must be a maximum weight maximal independent set.

Therefore this algorithm establishes it computes the desired object namely the maximum weight maximal independent set. If we want to compute the minimum weight maximal independent set, then we can consider the negation of the weights in this case, i.e. w'(a)= - w(a) $\forall$ a $\in$ E. The edges are taken in the non- decreasing order.

Time Complexity

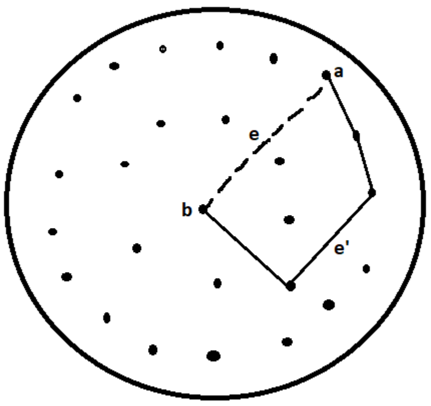
If m is not independent, then m(r) where r is the size of the set being tested. Time complexity is given by O(n log(n) + $\sum_{i=1}^{n}$ m(i)). Suppose it takes m time to deter-

mine the membership of I. if m is independent of the size of the matroid, then the time complexity is given by O(n log(n) + n.m) where n=$|E|$.

4.3 AN APPLICATION OF MATROID ALGORITHM

Let us consider the case where we have to compute the spanning tree in a given edge weighted graph (V, E, w) such that the total edge weight is minimum. Consider M= (E, τ, w) for each T $\in \tau$ (V, T) is a spanning tree.
Now we claim that M is a matroid.
$\emptyset \in$ M this is trivial.
If T $\in \tau$, then (V, T) is a spanning tree. Suppose T' $\subset$ T then (V, T') also cannot have a cycle. Hence (V, T') is also a spanning tree. Thus T' $\in \tau$.
Let us consider two spanning trees T$_1$ and T$_2$ such that $|T_1| \leq |T_2|$.
Then we need to show that there exists e' $\in$ T$_1$ $-$ T$_2$
such that (T$_1$ $-$ {e'}) $\cup$ {e} $\in \tau$.

Fig



If we consider the edge e $\notin$ T$_1$ and add in T$_1$, then there creates a cycle. If we take any other paths of the edge ab we will obtain a cycle. Hence (T$_1$ $-$ e') $\cup$ {e} $\in \tau$. Hence M= (E, τ, w) is a matroid.

Observe that every spanning tree of the given graph (V, E, w) is a member of τ.

4.3.1 THEOREM

If (V, T) is a spanning tree of (V, E, w) if and only if T is a maximal independent set.

PROOF –

In necessary condition assume the contrary.

Then there exists T' $\in \tau$ such that T τ. Suppose e $\in$ T' $-$ T and since T is a spanning tree, then any two pair has a path, so the edge e in T' creates a cycle which implies that T' $\notin \tau$. This contradiction establishes that the spanning trees are maximal independent set.
Suppose there are some components in H, then n$_i$ represents the number of vertices in each corresponding component. Then Σn$_i$ $= |V|$. Conversely,
assume H $\in \tau$ which is a maximal independent set and it is not a spanning tree. Consider the $|H|$ = cardinality of edges in a spanning tree = $|V|$ - 1. Then the number of edges, $|H| = \sum_{i=1}^{k}$ n$_i$ $-$ k $< |V|$ - 1. So there must be at least one connected components with n$_i$ vertices and at least n$_i$ edges. Then this component must have cycle. Hence H is a spanning tree.

Therefore the problem of computing a minimum spanning tree in terms of matroid is to compute a minimum weight maximal independent set.

4.4 KRUSKLA'S ALGORITHM

Kruskal's greedy algorithm for minimum spanning tree can be abstracted to a wider class of problems when one realises that tree edges T $\subseteq$ E in a graph determine an independent sets in matroids. In a connected graph, the edges in a spanning tree is the maximal independent set.
This standard greedy algorithm of ordering the elements by least weight is an optimal algorithm for optimally computing a linear function over any matroid.

4.4.1 ALGORITHM

```
1. Sort the elements of E in non-increasing order of their weights.
2. A=$\emptyset$
3. For i=1 to $|E|$ do
If (A $\cup$ x$_i$ $\in$ I)
Then A = A $\cup$ x$_i$;
4. Return A.
```

4.5 PROBLEM

A group of students wants to find the minimum distance between twenty places in Ernakulam. The students considered it as a graph G= (V, E) in which nodes

represent the place that they want to visit.

Each edge e is the road link that connects these places. For each pair of nodes u, v $\in$ V they want to select the minimum distance. So that one can find a spanning tree T of G so that for every pair of nodes u, v the unique path in the tree actually attains the optimum solution. Therefore, give an algorithm constructing a spanning tree T which optimises the given problem.

Solution -

We can solve this problem with the help of Kruskal's algorithm which we have discussed above.

For a general study of this problem the Python code of Kruskal's algorithm is used to solve this problem. Let us consider the 20 places, that student wants to visit.

The places are-

1.Aluva

2.Angamaly

3.Chellanam

4.Chendamangalam

5.Cheranallur

6.Elanji

7.Kadamakkudy

8.Kochi

9.Kodanad

10.Kothamangalam

11.Kumbalam

12.Kuttampuzha

13.Malayattoor

14.Mulavukad

15.Muvattupuzha

16.Nedumbassery

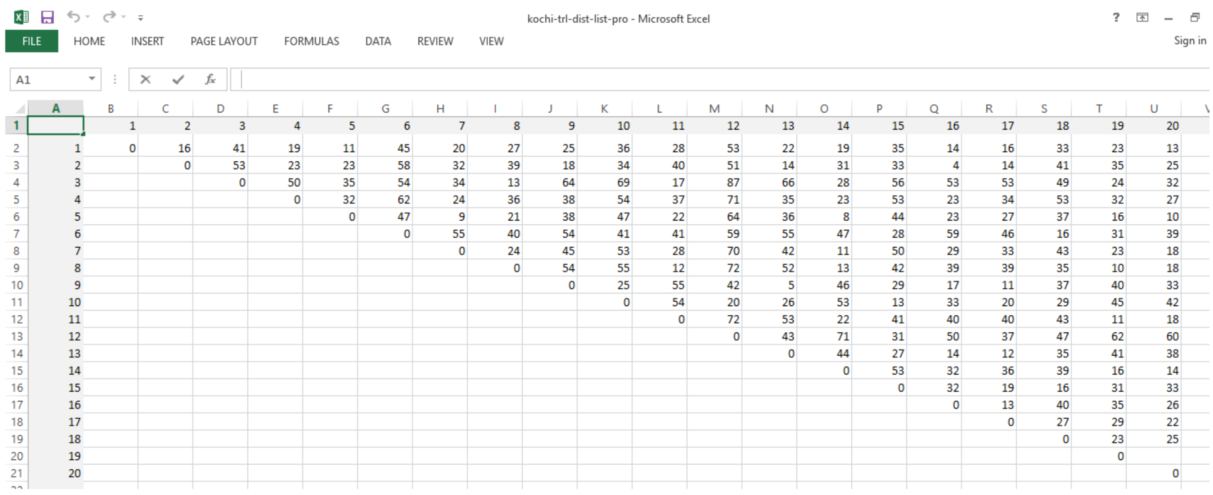
17.Perumbavoor

18.Ramamangalam

19.Thripunithura

20.Vazhakala

The distances are as follows;

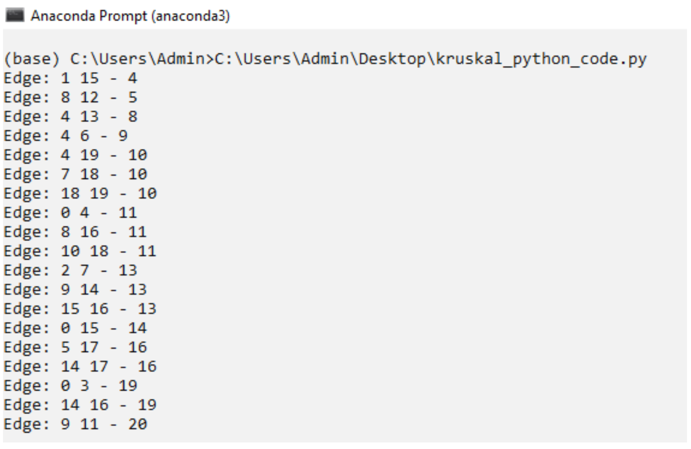


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	16	41	19	11	45	20	27	25	36	28	53	22	19	35	14	16	33	23	13
2		0	53	23	23	58	32	39	18	34	40	51	14	31	33	4	14	41	35	25
3			0	50	35	54	34	13	64	69	17	87	66	28	56	53	53	49	24	32
4				0	32	62	24	36	38	54	37	71	35	23	53	23	34	53	32	27
5					0	47	9	21	38	47	22	64	36	8	44	23	27	37	16	10
6						0	55	40	54	41	41	59	55	47	28	59	46	16	31	39
7							0	24	45	53	28	70	42	11	50	29	33	43	23	18
8								0	54	55	12	72	52	13	42	39	39	35	10	18
9									0	25	55	42	5	46	29	17	11	37	40	33
10										0	54	20	26	53	13	33	20	29	45	42
11											0	72	53	22	41	40	40	43	11	18
12												0	43	71	31	50	37	47	62	60
13													0	44	27	14	12	35	41	38
14														0	53	32	36	39	16	14
15															0	32	19	16	31	33
16																0	13	40	35	26
17																	0	27	29	22
18																		0	23	25
19																			0	
20																				0

```
' ' ' kruskal- python code' ' '

class Graph:
    def __init__(self, vertex):
        self.V = vertex
        self.graph = []

    def join_edge(self, p, q, r):
        self.graph.append([p, q, r])


    def search(self, parent, j):
        if parent[j] == j:
            return j
        return self.search(parent, parent[j])

    def merge(self, parent, rank, u, v):
        uroot = self.search(parent, u)
        vroot = self.search(parent, v)
        if rank[uroot] < rank[vroot]:
            parent[uroot] = vroot
        elif rank[uroot] > rank[vroot]:
            parent[vroot] = uroot
        else:
            parent[vroot] = uroot
            rank[uroot] += 1
    def kruskal(self):
        result = []
        i, e = 0, 0
        self.graph = sorted(self.graph, key=lambda item: item[2])
        parent = []
        rank = []
        for vertex in range(self.V):
            parent.append(vertex)
            rank.append(0)
        while e < self.V - 1:
            p, q, r = self.graph[i]
            i = i + 1
            u = self.search(parent, p)
            v = self.search(parent, q)
            if u != v:
                e = e + 1
                result.append([p, q, r])
                self.merge(parent, rank, u, v)
        for p, q, weight in result:
            print("Edge:",p, q,end =" ")
            print("-",weight)

g=Graph(20)
g.join_edge(0, 1, 16)
g.join_edge(0, 2, 41)
g.join_edge(0, 3, 19)
g.join_edge(0, 4, 11)
g.join_edge(0, 5, 45)
g.join_edge(0, 6, 20)
g.join_edge(0, 7, 27)
g.join_edge(0, 8, 25)
g.join_edge(0, 9, 36)
g.join_edge(0, 10, 28)
g.join_edge(0, 11, 53)
g.join_edge(0, 12, 22)
g.join_edge(0, 13, 19)
g.join_edge(0, 14, 35)
g.join_edge(0, 15, 14)
g.join_edge(0, 16, 16)
g.join_edge(0, 17, 33)
g.join_edge(0, 18, 23)
g.join_edge(0, 19, 13)
g.join_edge(1, 2, 53)
g.join_edge(1, 3, 23)
g.join_edge(1, 4, 23)
g.join_edge(1, 5, 58)
```

In the above program, first we create a class to represent Graph. Next we create a init function, for the vertices of the graph, where 'self' is a default dictionary to store the graph.
The join_edge function is created to add an edge to the graph. The merge function is build, so that it does the union of the two sets of u and v. Since it is a union-find algorithm, first it determines which subset a particular element belong to.
This can be done to determine if two elements are in same subsets. Next it joins the two subsets into a single subset. But here, it is used to check whether the undirected, connected graph contains cycle or not.
The next step shows that, it attaches the smaller rank tree under root of higher rank tree. If ranks are same then make one as root and add one to its rank.

The function kruskal is the important part of the python program, to construct the MST. The result variable stores the required output & i, e are used as index variable for sorted edges and result[] respectively.
First the edges are sorted in non-decreasing order of their weight. Next a variable V(subsets) is created with single elements. In the output graph, we require V-1 edges.
Secondly, it picks the smallest weighted edge and adds the index for next iteration. If including this edge does not create cycle it is included in the result, then it is incremented in the result for the next iteration. If cycle is created then the edge is eliminated.
The output of the Program is given below.



Anaconda Prompt (anaconda3)

```
(base) C:\Users\Admin>C:\Users\Admin\Desktop\kruskal_python_code.py
Edge: 1 15 - 4
Edge: 8 12 - 5
Edge: 4 13 - 8
Edge: 4 6 - 9
Edge: 4 19 - 10
Edge: 7 18 - 10
Edge: 18 19 - 10
Edge: 0 4 - 11
Edge: 8 16 - 11
Edge: 10 18 - 11
Edge: 2 7 - 13
Edge: 9 14 - 13
Edge: 15 16 - 13
Edge: 0 15 - 14
Edge: 5 17 - 16
Edge: 14 17 - 16
Edge: 0 3 - 19
Edge: 14 16 - 19
Edge: 9 11 - 20
```

Hence the required solution.

Chapter 5

CONCLUSION

This study, points to the highly influential role played by Graph theory in Matroids, since there are similarities with linear independence and dimension in vector spaces.
The study of matroid theory actually simplifies the concept of minimum spanning tree and its applications with the establishment of the proofs.

The representation of Kruskal's algorithm i.e. the greedy algorithm achieves the optimal solution of the problem. It is most convenient and easy to understand and uses the necessary & sufficient conditions of the sets, to be tested.
The Python code performed here can be considered as the general code for the computation of a minimum spanning tree.

Chapter 6

REFERENCE

[1] Douglas B West, “Introduction to Graph Theory”, Second Edition, Pearson Education, 2001.

[2] Deo Narsingh , “Graph Theory with Applications to engineering and Computer Science”Second Edition, Prentice-Hall India Learning Private Limited, 1979

[3] Jaan Kiusalaas, “Numerical Methods in Engineering with Python 3 ”,Third Edition, Cambridge University Press, 2013

[4] David L.Neel, Nancy Ann Neudauer,“Matroids You have known” from Mathematics Magazine.

[5] H. Whitney, “On the Abstract properties of Linear Independence ”, Amer J Math 57(3) (1935) 509-533

[6] Robin J Wilson, “Introduction to Graph Theory”, Fourth Edition, Prentice Hall, 1996.

[7] J.Oxley, “What is a Matroid?”
Preprint http://www.math.lsu.edu/oxley/survey4.pdf

[8] Even, Shimon,“Graph Algorithms”, Second Edition. Cambridge University Press, 2012.

[9] NPTEL, “Matroids”, https://youtu.be/a5W3oQ8gdtw

[10] NPTEL, “Minimum Spanning Trees ”, https://youtu.be/M9FMtiCvvhs

[11] “Python program”, Retreived from, https://www.programiz.com/dsa/kruskal-algorithm