Minimum Spanning Trees

MA428, Dr Katerina Papadaki

I. Introduction

In this lecture we will cover the following:

1) What is an MST?
2) Characteristics of MST Solutions.
3) Kruskal's Greedy algorithm.
4) LP formulation
5) Greedy algorithms and matroids
6) Applications

This lecture covers material from chapter 13 of the book "Network Flows" by R.K. Ahuja, T.L. Magnanti, J.B. Orlin (we call the book AMO).

II. What is an Minimum Spanning Tree (MST)?

Define $G = (N, E)$ to be a *graph* made up of $n = |N|$ nodes; $m = |E|$ edges, each denoted by a pair of nodes (i, j) connected by the edge, where $i, j \in N$; and cost c_{ij} for each edge. For this lecture we assume that G is a connected graph without self-loops or multiple edges. A *sub-graph of* G, is a graph that has some or all nodes and edges of G.

Definition An acyclic connected sub-graph of G is called a *tree*. In general a *tree* is an acyclic connected graph.

Definition An acyclic sub-graph of G is called a *forest*. In general a *forest* is an acyclic graph. A forest is composed of many trees.

K. Papadaki is with the Department of Mathematics at London School of Economics, United Kingdom
E-mail: k.p.papadaki@lse.ac.uk

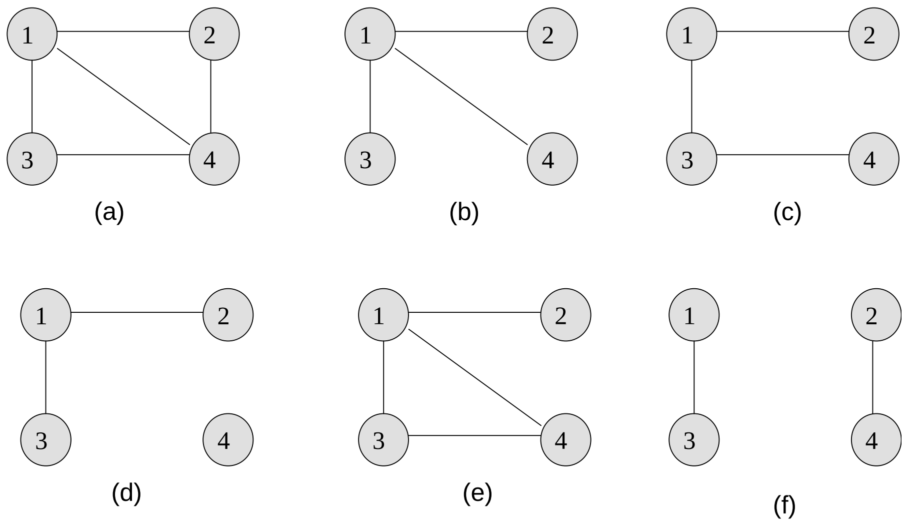

Fig. 1. (a) underlying graph G with n=4 and m=5; (b) and (c) spanning trees; (d) a non-spanning tree (doesn't span all nodes); (e) a non-tree (cyclic) graph (f) a forest with two trees.

Definition A spanning tree T of G is a connected acyclic sub-graph of G containing all n nodes.

Figure 1 shows examples of spanning trees, non-spanning trees, non-trees and forests. Let the edges of a spanning tree be called *tree edges* and the rest be called *non-tree edges*.

Observe that any spanning tree T has the following four properties:

1) Every spanning tree has exactly $n - 1$ edges.
2) There is a unique path in T between any two nodes of G.
3) Every non-tree edge (k, l), together with the unique path in T between nodes k and l, defines a cycle. [See Figure 2].
4) If we delete any tree edge (i, j) from T, the resulting graph partitions the node set N into two subsets: say sets S and $\bar{S}$. Edges of G with one end point in S, and the other in $\bar{S}$, constitute a cut, which is defined below (see Figure 2).

Definition Let T be a spanning tree of $G = (N, E)$. For every tree edge $(i, j) \in T$, removing (i, j) from T partitions the node set N into two subsets S, $\bar{S}$, with $i \in S$, $j \in \bar{S}$. We define a

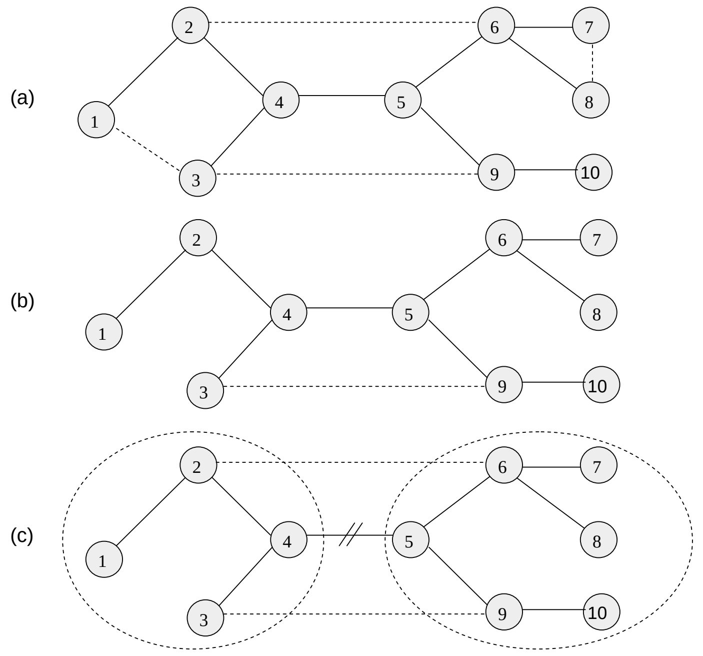

Fig. 2. (a) Graph G with edges of T in unbroken lines. (b) Adding edge $(3,9)$ to T forms the unique cycle $3-4-5-9-3$. (c) Deleting edge $(4,5)$ forms the cut $[S,\bar{S}]$, where $S=[1,2,3,4]$ and $\bar{S}=[5,6,7,8,9,10]$, which is the set of edges $[S,\bar{S}]=\{(2,6),(4,5),(3,9)\}$.

cut $[S,\bar{S}]$ as the set of edges in E, that connect a node in S to a node in $\bar{S}$.

We wish to find a spanning tree, that has the smallest total cost.

Definition A Minimum Spanning Tree (MST), denoted T^*, of G is a spanning tree with the minimum total cost.

Note that T^* may not be unique. Further, note that the edge weights c_{ij} might not be costs but also a desirable quantity that we want to maximize.

Definition A Maximum Spanning Tree (MXST) of G is a spanning tree with the maximum total edge weights.

The results on the MST that we derive in this lecture also translate to the MXST.

III. Characteristics of Solutions

Given that T is a spanning tree, the following conditions are equivalent to T being a MST.

Proposition 3.1: Cut Optimality: T^* is a minimum spanning tree if and only if it satisfies the following Cut Optimality Conditions (COC):

For every tree edge $(i,j) \in T^*$, we have $c_{ij} \leq c_{kl}$ for every edge (k,l) contained in the cut formed by deleting edge (i,j) from T^*.

Proof: Let T^* be a MST and suppose that the COC do not hold: there exists edge $(i,j) \in T^*$ such that $c_{ij} > c_{kl}$ for some non-tree edge (k,l) contained in the cut of (i,j). Replacing, (k,l) with (i,j) in T^* forms a tree with lesser cost, which contradicts the assumption that T^* is a MST.

Suppose T^* satisfies COC. Let T^0 be a MST such that $T^0 \neq T^*$. Then T^* contains at least one edge (i,j) not in T^0. Removing (i,j) from T^* creates a cut $[S,\bar{S}]$. Now, if we added (i,j) to T^0 this will create a cycle $C \subseteq T^0$, and this cycle C will contain at least one edge from the cut $[S,\bar{S}]$. In fact C will contain (i,j) from T^* and an odd number of edges from $C \cap [S,\bar{S}] \subseteq T^0$. Further, these odd number of edges from $C \cap [S,\bar{S}]$ cannot be in T^* since (i,j) is the unique T^* edge of the cut $[S,\bar{S}]$. Pick one of these edges in $C \cap [S,\bar{S}]$ and call it (k,l). When we introduced (i,j) into T^0 we created the cycle C, so now if we remove the edge (k,l) of the cycle, T^0 remains a tree, which spans all the nodes and thus it remains a spanning tree. Further, from COC of T^* we know that $c(i,j) \leq c(k,l)$ so the cost of T^0 is not increa2sed and thus T^0 is still a MST. Thus, we have edited T^0 to have one more edge in common with T^* (since $(i,j) \notin T^0$ and $(k,l) \notin T^*$) and still remain a MST. Continuing this way we can turn T^0 into T^*. ■

Proposition 3.2: Path Optimality: T^* is a minimum spanning tree if and only if it satisfies the following Path Optimality Conditions (POC):

For every non-tree edge $(k,l) \in G$, we have $c_{ij} \leq c_{kl}$ for every tree edge (i,j) contained in the unique path in T^* connecting nodes k and l.

Proof: Let T^* be a MST and suppose that the POC do not hold: for a non-tree edge (k,l) there exist a tree edge (i,j) on the unique path from k to l such that $c_{ij} > c_{kl}$. Substituting (k,l) instead of (i,j) in T^* gives a spanning tree with strictly less cost, which is a contradiction.

We show that the POC imply the COC. Let $(i,j) \in T^*$ which defines the cut $[S,\bar{S}]$. Pick any other $(k,l) \in [S,\bar{S}]$. The edge (i,j) must belong to the unique path that joins k and l. By the

POC, we have $c_{ij} \leq c_{kl}$. Since this is valid for any non-tree edge in the cut formed by (i, j), the COC hold in T^*. Thus, T^* is a MST. ■

Note that similar conditions hold for Maximum Spanning Trees (MXST), only the $\leq$ are replaced with $\geq$.

IV. Kruskal's Greedy algorithm

A. An example of Kruskal's algorithm

A simple algorithm by Kruskal for finding T^* builds it by adding one by one the cheapest available edge that does not form a cycle with the previously selected edges. Consider the example in table I.

Edges	$(1,2)$	$(1,3)$	$(1,5)$	$(2,3)$	$(2,4)$	$(2,6)$	$(3,5)$	$(3,6)$	$(4,6)$	$(5,6)$
Costs	5	7	10	8	14	12	10	15	15	9

TABLE I
The edges of graph G with their respective costs.

One way to implement Kruskal's algorithm is to start by sorting all the edges in a non-decreasing order of their costs - a process requiring $O(mlogm)$ operations for arbitrarily large costs. When sorting, a tie-breaking rule will be needed to deal with edges that have the same cost.

Tie-breaking rule: We shall use the rule that each edge (i, j) is denoted in such a way that $i < j$ and when choosing between two edges with the same cost, the edge with the smaller first index gets priority. If the first index is the same for two edges, the one with the smaller second index gets priority.

After sorting the edges of G (given in table I) we get the sorted version in table II:

Edges	$(1,2)$	$(1,3)$	$(2,3)$	$(5,6)$	$(1,5)$	$(3,5)$	$(2,6)$	$(2,4)$	$(3,6)$	$(4,6)$
Costs	5	7	8	9	10	10	12	14	15	15

TABLE II
The edges of graph G sorted with respect to their costs (using the tie-breaking rule described above).

Next we prepare a $LIST$ containing edges in T^*. Initially the list is empty. We examine the edges in the sorted order one by one and check if adding the edge will create a cycle with the edges already in the $LIST$. If not, we add the edge to the $LIST$. We stop when the list contains $n-1$ edges. At all intermediate steps the $LIST$ will consist of forests (no cycles and some connected and some isolated nodes).

STEP	$LIST$	FORESTS	NO. OF EDGES	CONTINUE?
0	Empty	$\{1\};\{2\};\{3\};\{4\};\{5\};\{6\}$	0	yes
1	Add $(1,2)$	$\{1,2\};\{3\};\{4\};\{5\};\{6\}$	1	yes
2	Add $(1,3)$	$\{1,2,3\};\{4\};\{5\};\{6\}$	2	yes
3	Not $(2,3)$	No change	2	yes
4	Add $(5,6)$	$\{1,2,3\};\{5,6\};\{4\}$	3	yes
5	Add $(1,5)$	$\{1,2,3,5,6\};\{4\}$	4	yes
6	Not $(3,5)$	No change	4	yes
7	Not $(2,6)$	No change	4	yes
8	Add $(2,4)$	$\{1,2,3,4,5,6\}$	5	Stop

TABLE III

KRUSKAL'S ALGORITHM APPLIED ON THE SORTED EDGES OF TABLE II

B. *Kruskal's algorithm*

A formal description of the algorithm is given in Algorithm 1.

Algorithm 1 Kruskal's Algorithm

```
begin
    order the edges in $E = \{e_1, e_2, \ldots, e_m\}$ so that $c_1 \leq c_2 \leq \ldots \leq c_m$;
    (where $c_k$ is the cost of edge $e_k$)
    set $LIST := \emptyset$;
    while $|LIST| < n-1$ do
        if $LIST \bigcup \{e_j\}$ does not form a cycle then $LIST := LIST \bigcup \{e_j\}$;
    end of while-loop;
    LIST is a minimum spanning tree;
end;
```

There are only two major operations in this algorithm:

1) sorting edges by cost, and
2) checking the $LIST$ for a cycle.

Sorting:

Even for this simple procedure there are important and interesting issues of implementation that affect the number of operations required in the worst case. For example, a crude method of sorting will require $O(m^2)$ comparisons, but a good one uses only $O(mlogm)$. [See D. E. Knuth, The Art of Computer Programming, Vol. III: Sorting and Searching. Addison-Wesley, 1973.]

Checking for a cycle:

Also what data structure is best suited for checking for a cycle? We can start with the trivial spanning forest in G consisting of n isolated nodes. At each step we select an edge which leads to the merging of forest components. For example, selecting edge $(1, 2)$ at step 1 links nodes 1 and 2 into one component. At step 2, the selection of edge $(2, 3)$ leads to a new merged component with nodes $1, 2, 3$. Thus during the progress of the algorithm we have several trees that can be stored as linked lists. For example, after adding edge $(5, 6)$ we have three trees, one with nodes $1, 2, 3$ another with nodes $5, 6$ and the smallest one with the (as yet) isolated node 4.

When checking the next edge we can check if both nodes of the edge belong to the same tree. If they do, the edge has to be rejected. If not, they belong to two different trees. In this case we add the edge to the $LIST$ and merge the two trees into one. For each edge we need to go through all the nodes of the forests in $LIST$ to check if they belong to the same tree. This requires $O(nm)$ operations in the worst case. There are improvements to this step that are described in AMO's book.

C. *Optimality of Kruskal's algorithm*

Kruskal's algorithm gives us a spanning tree solution. But how do we know that this is the MST solution? Let T be the tree given by Kruskal. We will show that T satisfies the *path optimality conditions*. In Kruskal's algorithm each non-tree edge (k, l) was rejected because it created a cycle with edges of lesser cost. Thus each non-tree edge (k, l) has the highest cost from all the tree edges that form the unique path from k to l. The latter are the path optimality conditions. Since T satisfies the path optimality conditions, it is a MST.

V. Modelling problems as MST problems

A. Optimal Message Passing

An intelligence service has n agents in a non-friendly country. Each agent knows some of the other agents and has in place procedures for arranging a rendezvous with anyone s/he knows. For each such possible rendezvous, say between agent i and agent j, any message passed between these agents will fall into hostile hands with a certain probability p_{ij}. The group leader wants to transmit a confidential message among all the agents while minimizing the probability that the message is intercepted.

We assume that the events of pairs of agents being intercepted are independent of each other (for different pairs of agents). Thus, the events of pairs of agents not being intercepted are also independent of each other. The probability that the message is not intercepted by any pair of agents is equal to $\prod_{(i,j)\in T}(1-p_{ij})$, and thus the probability that the message is intercepted is equal to $1-\prod_{(i,j)\in T}(1-p_{ij})$.

If we represent the agents by nodes, and each possible rendezvous by an edge, then in the resulting graph G we would like to identify a spanning tree T that minimizes the probability of interception $1-\prod_{(i,j)\in T}(1-p_{ij})$. Since this expression does not decompose into edge costs of each pair of agents, we consider the problem of maximizing the probability of avoiding interception given by $\prod_{(i,j)\in T}(1-p_{ij})$, which decomposes as a product. So we would like to find a spanning tree T that maximizes $\prod_{(i,j)\in T}(1-p_{ij})$.

When finding a minimum or maximum spanning tree we minimize or maximize the *sum of the edges*. Here we are faced with maximizing a *product*. To solve this problem we note that maximizing $\prod_{(i,j)\in T}(1-p_{ij})$ is the same as maximizing $log\left(\prod_{(i,j)\in T}(1-p_{ij})\right)$, since the log function is a monotonically increasing function. Then,

$$log\left(\prod_{(i,j)\in T}(1-p_{ij})\right) = \sum_{(i,j)\in T} log(1-p_{ij})$$

Now our objective function decomposes as a sum. Thus we can define the length of an edge (i,j) as: $c_{ij}=log(1-p_{ij})$ and solve a maximum spanning tree (MXST) problem. This will give the group leader the spanning tree that minimizes the probability that the message is intercepted.

What is more interesting is the fact that finding an MST for $c_{ij}=p_{ij}$ also solves the problem. To see why, note that: if $log(1-a)\geq log(1-b)$, then $(1-a)\geq(1-b)$ and so $b\geq a$. This implies that arranging $log(1-p_{ij})$ in a non-increasing order to maximize the product of $(1-p_{ij})$ will produce the same ordering as arranging p_{ij} in a non-decreasing order for minimizing the sum of probabilities. It follows that:

$$\text{MXST for } \max \prod_{(i,j)\in T} (1-p_{ij}) = \text{ MST for } \min \sum_{(i,j)\in T} p_{ij},$$

even though $\sum_{(i,j)\in T} p_{ij}$ does not equal to the probability of an interception. Remember from the sum rule of probability that only if events A and B are mutually exclusive (disjoint) do we have $P(A \text{ or } B) = P(A) + P(B)$; normally we have $P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$.

B. Minimax Problems on a Graph

Minimax Paths

In a network $G = (N, E)$ with edge costs c_{ij}, we define the *value of a path* P from node p to node q as the maximum cost edge of all edges in P. The *all-pairs minimax path problem* requires that we determine, for every pair $[p, q]$ of nodes, a minimum value path from node p to node q. We show how to solve the all-pairs minimax path problem on G by solving a single minimum spanning tree problem.

Let T^* be a minimum spanning tree of G. Let P denote the unique path in T^* between a node pair $[p, q]$ and let (i, j) denote the maximum cost edge in P. So the value of the path P is c_{ij} . By deleting edge (i, j) from T^*, we partition the node set N into two subsets and therefore define a cut $[S, \bar{S}]$ with $i \in S$ and $j \in \bar{S}$. We know from the Cut Optimality conditions that this cut satisfies the following property:

$$c_{ij} \leq c_{kl} \text{ for each edge } (k,l) \in [S, \bar{S}].$$

Now, consider any path P' in G from node p to node q. This path must contain at least one edge (k, l) in $[S, \bar{S}]$. The Cut Optimality conditions imply that the value of this path P' will be at least c_{ij}. Since c_{ij} is the value of the path P in T^*, P must be a minimum value path from node p to node q. This observation establishes the fact that the unique path between any pair of nodes in T^* is a minimum value path between that pair of nodes.

The problem of finding a minimax path between two specific nodes arises in a variety of situations. To quote from AMO's book: ”As an example, consider a spacecraft that is about to enter the earth's atmosphere. The craft passes through different pressure and temperature zones that we can represent by edges of a network. It needs to fly along a trajectory that will bring the craft to the surface of the earth while keeping the maximum temperature to which the surface of the craft is exposed as low as possible. As an alternative, we might wish to select a path that will minimize the maximum deceleration during the descent.

Other examples of the minimax path problem arise when:

1) in traveling through a desert, we want to minimize the length of the longest stretch between rest areas; and
2) in travelling in a wheelchair, a person might wish to minimize the maximum ascent along the path segments."

Minimax Spanning Trees

Now define the *value of a spanning tree* T to be equal to the cost of its most expensive edge and denote it by $V(T)$. We call a *minimax spanning tree*, a spanning tree whose value is minimized. Let T^* be a minimax spanning tree, and let T be a MST of G. From the definition of the minimax spanning tree we have:

$$V(T^*) \leq V(T). \tag{1}$$

Let (k,l) be the most expensive edge on T and thus $V(T) = c_{kl}$. Removing edge (k,l) defines the cut $[S, \bar{S}]$, where $k \in S$ and $l \in \bar{S}$. We know from the Cut Optimality Conditions that edge (k,l) has the smallest cost from all the edges in the cut. T^* uses one of the edges in the cut. Thus, its value is greater than or equal to c_{kl}:

$$V(T^*) \geq c_{kl} = V(T). \tag{2}$$

From equations (1) and (2) we conclude that $V(T) = V(T^*)$ and thus T is a minimax spanning tree.

However, not all minimax spanning trees are MSTs. For example, consider a complete graph (each node connected to every other) on three nodes, i.e., a triangle and let the costs of the three edges be 1, 2 and 2. Any two edges define a minimax spanning tree but the one with the two most expensive edges (with cost 2 and 2) is not an MST.

C. Reducing Data Storage

In several different application contexts, it is more efficient (in terms of memory space) to store data in the form of a two-dimensional array of differences in the elements of data items rather than storing all the data items individually in rows. This is usually true when data elements in different data items have many similar entries and differ only at a few places. One such situation arises in the sequence of amino acids in a protein found in the mitochondria of different animals and higher plants. DNA sequencing of many life forms also share this characteristic of having many (over 95% in primates) similar entries.

Since the entities in the rows of data items are similar, one approach for saving memory is to store one row, called the reference row, completely, and to store only the differences between some of the rows so that we can derive each row from these differences and the reference row. Let c_{ij} denote the number of different entries in rows i and j; that is, if we are given row i, then by making c_{ij} changes to the entries in this row we can obtain row j, and vice versa. Suppose that the array contains four rows, represented by $R1$, $R2$, $R3$, and $R4$, and we decide to treat $R1$ as the reference row. Then one plausible solution is to store the differences between $R1\&R2$, $R2\&R4$, and $R1\&R3$. Clearly, from this solution, we can obtain rows $R2$ and $R3$ by making c_{12} and c_{13} changes to the elements in row $R1$. Having obtained row $R2$, we can make c_{24} changes to the elements of this row to obtain $R4$.

It is easy to see that it is sufficient to store differences between those rows that correspond to edges of a spanning tree, where the edge costs are the c_{ij} the number of difference between the rows. These differences permit us to obtain each row from the reference row. The total storage requirement for a particular storage scheme will be the length of the reference row (which we can take as the row with the least amount of data) plus the sum of the differences between the rows. Therefore, a minimum spanning tree would provide the least cost storage scheme.

VI. LP FORMULATION

The following formulation for the MST problem has $O(2^n)$ constraints. Let $x_{ij} = 1$ if the edge $(i, j) \in T^*$ and $x_{ij} = 0$ otherwise. Let $E(S)$ denote set of edges of G with both ends in a subset of nodes $S \subseteq N$.

$$\sum_{(i,j)\in E(S)} x_{ij} \leq \|S\| - 1 \quad \text{for any } S \subseteq N, S \neq \emptyset \tag{3}$$

$$\sum_{(i,j)\in E} x_{ij} = n - 1 \tag{4}$$

Conditions (3) forbid any cycle, because k nodes need at least k edges to form a cycle between them. If (3) is observed for all subsets S, then condition (4) ensures a spanning tree, since a graph with no cycles and $n-1$ edges is connected.

So the LP defined by:

$$\min_{x_{ij}} \sum_{(i,j)\in E} c_{ij}x_{ij} \quad \text{subject to (3) and (4),} \tag{5}$$

gives an LP formulation of the MST problem (see AMO chapter 13 for details).

VII. Greedy algorithms and Matroids

A. Preliminaries

A greedy algorithm is often used as a heuristic method to find a good starting solution. For example, in the Assignment problem or the Transportation problem, it is common to find the 'greedy' basic solution as a starting point. But these problems cannot, in general, be solved to optimality by a greedy algorithm. What is it about the MST problem that it can be solved to optimality by a greedy algorithm? Murty (page 477) identifies three features.

(i) The problem solution is a set of elements which can be built up one at a time.

(ii) Once an element is selected, it is never replaced - no backtracking is required.

(iii) Each new element selected for inclusion is the best available at that stage.

How can we identify other problems with the same features? The theory of *matroids* provides the abstraction needed for the purpose. To understand it let us reconsider Kruskal's algorithm. It works with two objects: the set of edges and forests which are subsets of edges. In the following discussion we define matroids and at the same time we make a parallel with the MST problem.

1) The *Ground Set* E that is a finite collection of objects. In the MST case the ground set E is the set of edges.
2) The *Subset Set* Φ is a collection of subsets of E. In the MST case Φ contains subsets of edges that form a forest (i.e. have no cycles); so Φ is the set of forests subgraphs of a graph G.

Given the above notation we proceed to the following definitions:

Definition The pair (E,Φ) is called a *subset system* or *independence system* if it satisfies the following property:

1) *Hereditary property*: If $I \in \Phi$ and $I' \subseteq I$, then $I' \in \Phi$.

   The elements of Φ in a subset (independence) system are called *independent sets*.

   The subset system (E, Φ) is a *matroid* if it also satisfies the following property:

2) *Growth property*: If $I(p), I(p+1) \in \Phi$ are subsets of E containing p and $p+1$ elements respectively, then there exists an element $e \in I(p+1)$ but not in $I(p)$, $e \in I(p+1) \backslash I(p)$, such that $G(p) \bigcup \{e\} \in \Phi$.

Any subset of a forest is a forest. So the pair (E, Φ), where E is the set of edges of G and Φ is the set of forests defined over edges of a graph G, is an independence system. Any forest is an independent set.

We will also show later in proposition 7.1 that forests also satisfy the growth property and thus form a matroid. Note that in Kruskal's algorithm we use the growth property: we start with a forest that contains n trees (and zero edges) and we end with a forest that contains one tree, the MST, (and $n-1$ edges). The growth property implies that the maximum cardinality of an independent set is a fixed number: this is because any set of lower cardinality can grow to the maximum cardinality. For the MST problem this means that the number of edges in every spanning tree is the same.

Note that in any independence system (E, Φ) we must have $\emptyset \in \Phi$.

B. Some other examples

Example 1: Nodes of a graph.

Consider the problem in which the ground set is the set of nodes in a graph $G = (N, E)$ and a set I of nodes is independent if each pair of nodes $p, q \in I$ are connected to each other, i.e. $(p.q)$ is an edge of G. Note that this forms an independence system since any subset of an independent set is also independent (satisfies the hereditary property). However, it does not satisfy the growth property. Thus, the maximum cardinality independent sets are no longer of the same size.

Example 2: Students from the same nationality.

Let the ground set be all the currently registered students at LSE. Suppose we define a set of LSE students as independent if all the students in the set have the same nationality. The hereditary property obviously holds, so this definition of independence is valid. But the growth property is not valid. If I take a set of five Greek students and six Chinese students, I cannot augment the size of my first independent set by adding any member of the larger set to it. So this definition of independence does not lead to a matroid.

Example 3: Students from different nationalities.

What if I defined a set of LSE students to be independent if all of them are from a different country! It is obvious that the hereditary property still holds. But now I have the growth property as well. Given an independent set with five students and another with six, even if some students in the two sets overlap, I can always find one in the larger set who can be added to the smaller one to give a larger independent set. So this notion of independence has a matroidal structure or it is a matroid.

Why is this important? Simply because if the growth property holds for an independent set, the size of the maximal independent set is fixed. The largest independent set of LSE students from different countries has size r where r is the total number of countries from which students come to LSE. Similarly, the largest no. of edges in a forest is $r = n - 1$ where n is the no. of nodes in G. This is the key to the applicability of the Greedy algorithm to the MST problem.

C. Growth property for forests.

Now we go back to the proof of the growth property for independent sets defined as forests on the ground set of edges.

Proposition 7.1: Let E be the set of all edges of G, and Φ be a collection of forest subgraphs of G. If $G(p)$ and $G(p+1)$ are forest subgraphs of G containing p and $p+1$ edges respectively, there exists an edge $e \in G(p+1)$ but not in $G(p)$, $e \in G(p+1)\backslash G(p)$, such that we can add e to $G(p)$ and produce another forest subgraph, i.e. $G(p) \bigcup \{e\} \in \Phi$.

Proof:

Suppose $G(p)$ is a forest of K trees: $T_1, T_2, \dots T_K$. We want to find an edge $e \in G(p+1)$ whose nodes do not belong to the same T_k, for any $k = 1, \dots K$. Because if the endpoints of e belonged to the same tree then it will form a cycle with the edges of that tree. We call an edge e where both its endpoints belong to the same tree T_k for all k, a forbidden edge (because it creates a cycle with edges of $G(p)$. We count the number of forbidden edges that forest $G(p+1)$ could have. Let's look at tree T_k and suppose it has m nodes and thus $m-1$ edges.

The maximum number of edges that $G(p+1)$ could have, i.e. with both endpoints in T_k is $m-1$ since $G(p+1)$ is a forest and with more than $m-1$ edges in m nodes we would have a cycle. Thus, the maximum number of forbidden edges that $G(p+1)$ can have in trees $T_1, \ldots, T_K$ is $|T_1| + \ldots + |T_K| = p$.

Therefore, since $G(p+1)$ has $p+1$ edges, it must have at least one edge e that does not have both ends in one of the T_k's (that is forbidden). Thus, $e \in G(p+1) \backslash G(p)$, and $e \bigcup G(p)$ is a forest. ■

D. Why the name independent set?

The term independence comes from the following matroid. Consider a real-valued matrix A. Let the collection of columns of A be called E - our basic set of objects or ground set. We say a set of columns of A is independent if the columns are linearly independent as vectors. If a set I (which is a subset of E) of columns is linearly independent, so is any subset I' of I. In fact the hereditary property of subsets I of objects E in matroids is referred to as independent sets because of this connection with linear independence in the columns of a matrix. The growth property for this set of objects E and the independent sets I, defined as linearly independent subsets of E, follows from elementary linear algebra results. The fact that the maximum number of linearly independent columns of A (i.e., the rank of A) is a constant is common knowledge.

Further borrowing further notation from linear algebra we refer to a *maximal independent set* of a matroid as a *basis* of a matroid.

E. Matroid Optimization Problem and the Greedy Algorithm

We now describe the MST problem as a matroid optimization problem.

Let (E, Φ) be a matroid. Suppose we associate a weight w_e to each element e of E and define the weight $w(S)$ for any subset S of E as the sum of weights of its elements:

$$w(S) = \sum_{e \in S} w_e.$$

Then the *matroid optimization problem* consists of finding a maximal independent set of the subset system (E, Φ) with the minimum weight. This is the generalization of the MST problem in matroid terms.

Algorithm 2 Greedy Algorithm

```
begin
order the elements of $E = \{e_1, e_2, \ldots, e_k\}$ so that $w_1 \leq w_2 \leq \ldots \leq w_k$;
set $LIST := \emptyset$;
for $j = 1$ to $k$ do
if $LIST \bigcup \{ej\}$ is independent then $LIST := LIST \bigcup \{ej\}$;
end of for-loop;
LIST is a minimum weight basis;
end;
```

We can attempt to solve this problem using a greedy algorithm which is a direct generalization of Kruskal's algorithm. The Greedy Algorithm is shown in Algorithm 2.

Theorem 7.2: The greedy algorithm solves the Matroid optimization problem.
For the proof of this theorem see AMO page 530.

VIII. Applications

Modern applications: Connecting villages in Kenya to a water source at minimum distance reported in last year's OR society newsletter.

MST - Applications from: http://www.ics.uci.edu/ eppstein/gina/mst.html

Applications of computational geometry. John Hershberger describes some geometric problems arising in his work at Mentor graphics including interpolation of thermal data, minimum spanning trees, and breakout routing in PC board design.

Cancer imaging. The BC Cancer Research Ctr. uses minimum spanning trees to describe the arrangements of nuclei in skin cells.

Cosmology at the University of Kentucky. This group works on large-scale structure formation, using methods including N-body simulations and minimum spanning trees.

Detecting actin fibers in cell images. A. E. Johnson and R. E. Valdes-Perez use minimum spanning trees for biomedical image analysis.

The Euclidean minimum spanning tree mixing model. S. Subramaniam and S. B. Pope use geometric minimum spanning trees to model locality of particle interactions in turbulent fluid flows. The tree structure of the MST permits a linear-time solution of the resulting particle-interaction matrix.

Extracting features from remotely sensed images. Mark Dobie and co-workers use minimum spanning trees to find road networks in satellite and aerial imagery.

Finding quasar superstructures. M. Graham and co-authors use 2d and 3d minimum spanning trees for finding clusters of quasars and Seyfert galaxies.

Learning salient features for real-time face verification, K. Jonsson, J. Matas, and J. Kittler. Includes a minimum-spanning-tree based algorithm for registering the images in a database of faces.

Minimal spanning tree analysis of fungal spore spatial patterns, C. L. Jones, G. T. Lonergan, and D. E. Mainwaring.

A minimal spanning tree analysis of the CfA redshift survey. Dan Lauer uses minimum spanning trees to understand the large-scale structure of the universe.

A mixing model for turbulent reactive flows based on Euclidean minimum spanning trees, S. Subramaniam and S. B. Pope.

Sausages, proteins, and rho. In the talk announced here, J. MacGregor Smith discusses Euclidean Steiner tree theory and describes potential applications of Steiner trees to protein conformation and molecular modeling.

Weather data interpretation. The Insight group at Ohio State is using geometric techniques such as minimum spanning trees to extract features from large meteorological data sets.

Part of Geometry in Action, a collection of applications of computational geometry. David Eppstein, Theory Group, ICS, UC Irvine.

IX. Exercises

1) Solve problem 13.2 (a) of AMO after replacing the word *minimum* by *maximum* in the question.
   Solve 13.2 (b) as well.
2) Solve 13.6 of AMO.
3) Give an alternative IP formulation for the MST problem, where you use constraint (4) and another set of $2^n - 2$ constraints that rule out any disconnected components.

4) Suppose we wish to choose a maximum weight subset of elements from the table below subject to the constraint that no two elements are from the same row of the matrix.

1	2	6	10
3	4	8	11
5	7	9	12
13	14	15	16

Write the problem as an IP problem and show that a greedy algorithm can be applied to find a solution.

If it was required to find a maximum weight subset such that no two are from the same row or column, how would your formulation and your answer about applicability of a greedy algorithm change?

5) Formulate the following knapsack problem faced by a librarian as a linear programming problem in integer variables and use a greedy heuristic to find a solution:

The librarian has to decide an optimum subset among 8 journals to renew the subscription with only 5000 available in the new budget for journals. The journals should be selected to maximize the number of users per year within the budget constraint. The relevant data is given below.

JOURNAL j	1	2	3	4	5	6	7	8
ANNUAL SUBSCRIPTION	800	950	1150	1650	1250	780	690	990
NO. OF USERS	7840	6715	8510	15015	7375	1794	897	8315

TABLE IV

DATA OF LIBRARY JOURNALS.

What journals would you subscribe using the greedy heuristic if the budget was 2440? Show that this solution is not optimal by finding another feasible solution that gets more users.

Show that the knapsack problem described above cannot be solved using the greedy algorithm by formulating it as an independence system and showing that it is not a matroid.