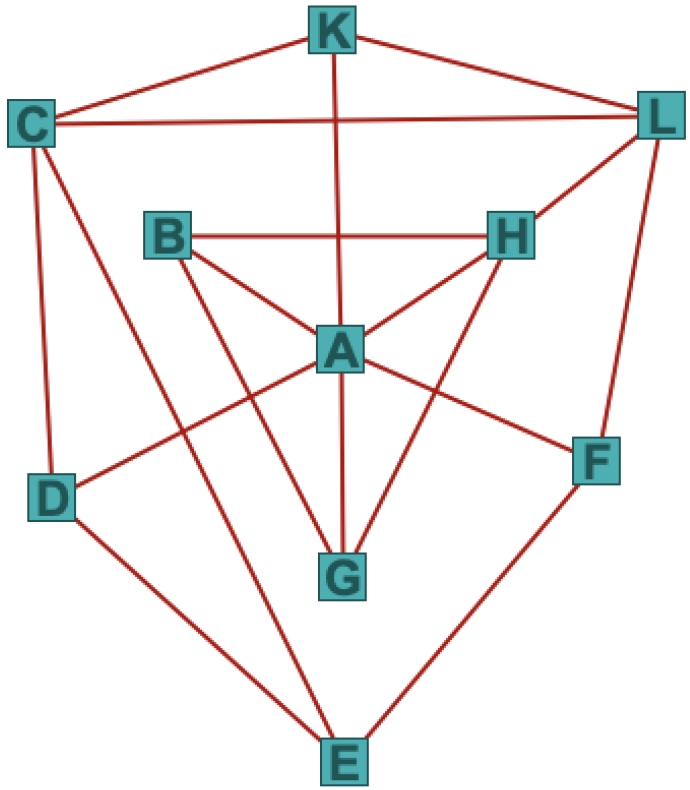
MATH 322 – Graph Theory
Fall Term 2021

Notes for Lecture 13

Tuesday, October 19

Past Exam Problem



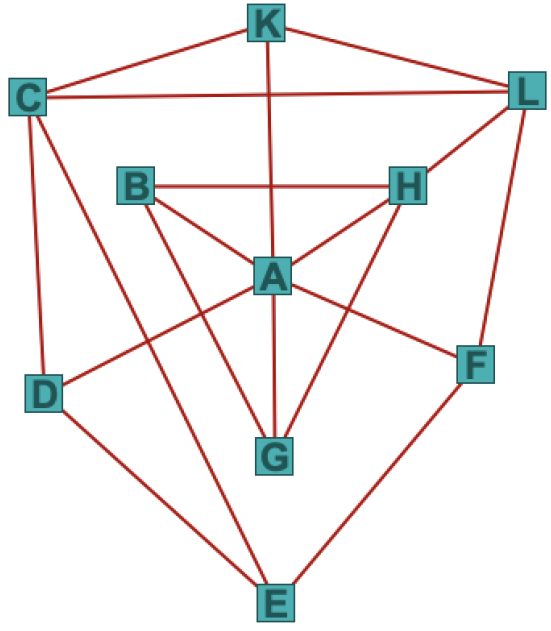
Graph G_0

(a) Show that $\kappa(G_0) = 2$. Give a full justification.

(b) What is $\lambda(G_0)$? Determine it precisely, and justify your answer fully.

(c) Determine $\kappa(C, F)$ precisely, and justify your answer fully.

Past Exam Problem



Next Main Topic:

What (other) ways/notions can we come up with to capture/describe how 'efficiently' connected a given connected graph is?

Spanning tree of a graph

Definition

Let G be a connected graph, and let T be a subgraph of G.
T is called a *spanning tree* of G if

- T is a tree

Spanning tree of a graph

Definition

Let G be a connected graph, and let T be a subgraph of G.
T is called a *spanning tree* of G if

- T is a tree
- and T contains all vertices of G.

Spanning tree of a graph

Definition

Let G be a connected graph, and let T be a subgraph of G.
T is called a *spanning tree* of G if

- T is a tree
- and T contains all vertices of G.

Observation 1

A spanning tree of G is a *minimal* connected subgraph of G that contains all the vertices of G ("minimal" here means that, if we remove any more edges from T, then we will instead end up with a disconnected subgraph of G).

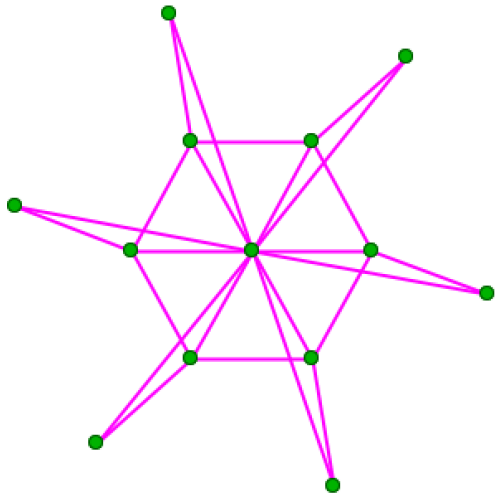
We could think of a spanning tree of a connected graph G as a very simple 'skeletal frame' for the graph.

Observation 2

In general, a connected graph G may have several spanning trees *(in fact, you could check that this is true for all connected graphs which are NOT trees themselves; left as practice; see also next theorem and the way to justify it)*.

Observation 2

In general, a connected graph G may have several spanning trees *(in fact, you could check that this is true for all connected graphs which are NOT trees themselves; left as practice; see also next theorem and the way to justify it)*.

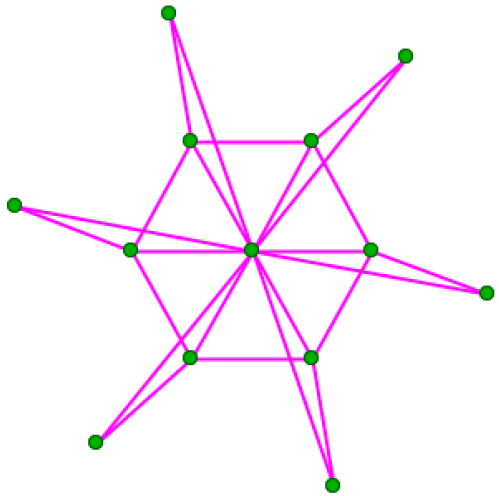


A 'flower' graph

and three spanning trees for it:

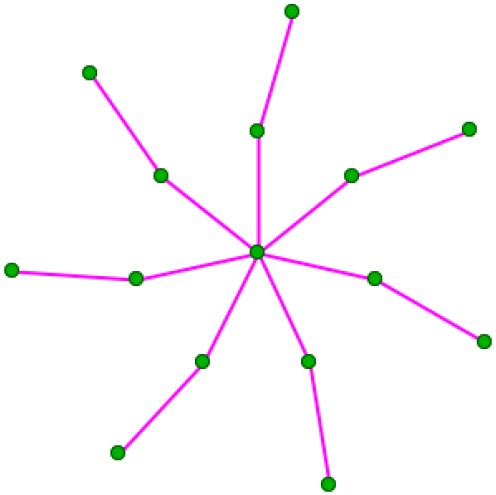
Observation 2

In general, a connected graph G may have several spanning trees *(in fact, you could check that this is true for all connected graphs which are NOT trees themselves; left as practice; see also next theorem and the way to justify it)*.



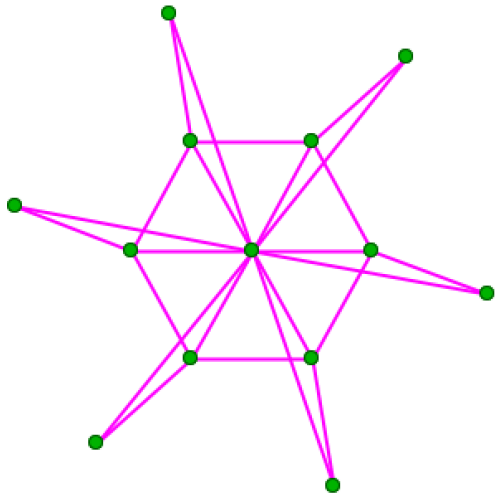
A 'flower' graph

and three spanning trees for it:



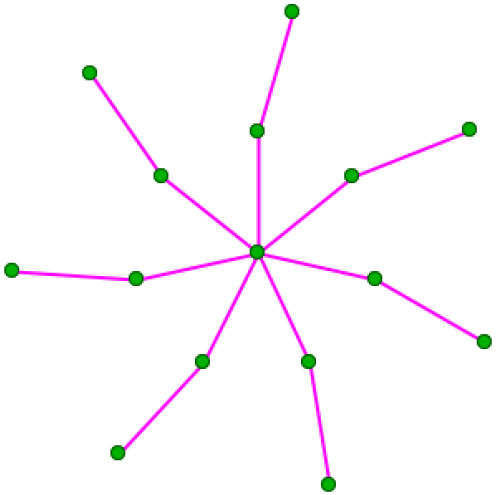
Observation 2

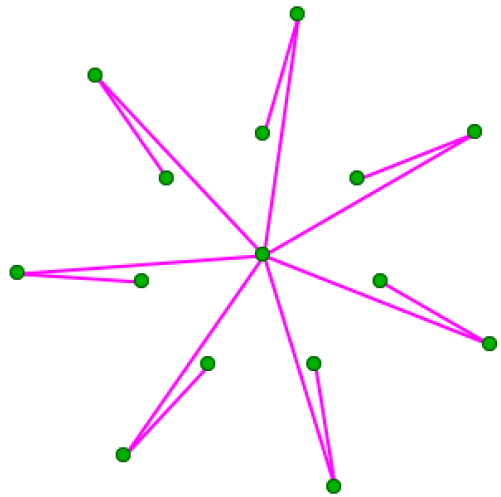
In general, a connected graph G may have several spanning trees *(in fact, you could check that this is true for all connected graphs which are NOT trees themselves; left as practice; see also next theorem and the way to justify it)*.



A 'flower' graph

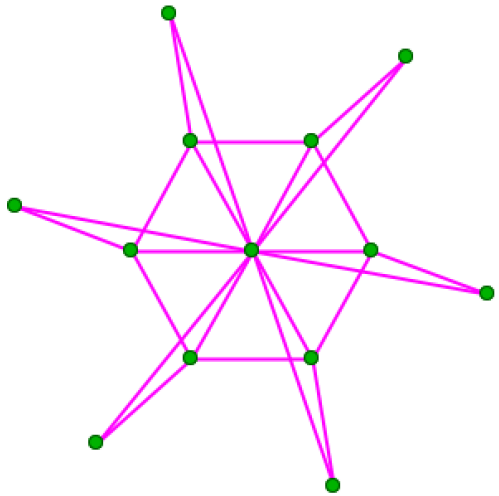
and three spanning trees for it:





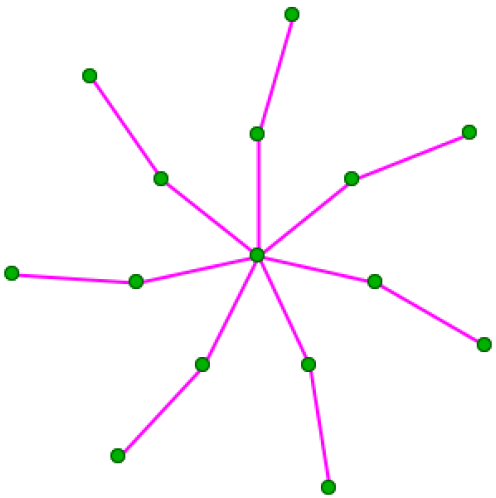
Observation 2

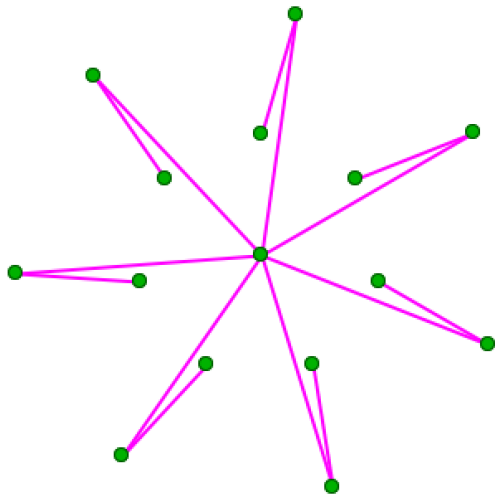
In general, a connected graph G may have several spanning trees *(in fact, you could check that this is true for all connected graphs which are NOT trees themselves; left as practice; see also next theorem and the way to justify it)*.

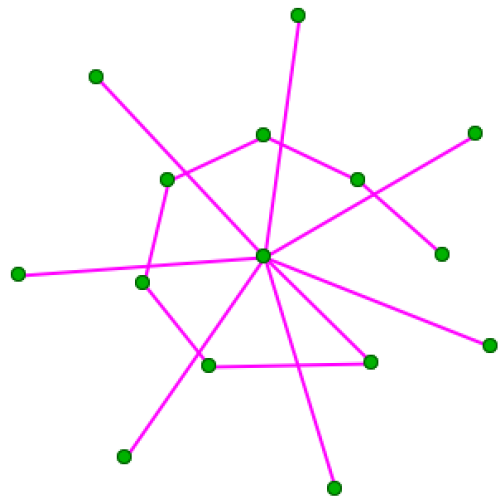


A 'flower' graph

and three spanning trees for it:







Important Observation

Theorem

Every connected graph G has at least one spanning tree.

Important Observation

Theorem

Every connected graph G has at least one spanning tree.

Proof of the theorem. Assume that G has order n with $n \geqslant 3$ (since otherwise G is already a tree and the conclusion is obvious).

Important Observation

Theorem

Every connected graph G has at least one spanning tree.

Proof of the theorem. Assume that G has order n with $n \geqslant 3$ (since otherwise G is already a tree and the conclusion is obvious).

We can prove the theorem using induction in the size of G.

Important Observation

Theorem

Every connected graph G has at least one spanning tree.

Proof of the theorem. Assume that G has order n with $n \geqslant 3$ (since otherwise G is already a tree and the conclusion is obvious).

We can prove the theorem using induction in the size of G.

- **Base Case:** *What is the minimum size to consider here?*

Important Observation

Theorem

Every connected graph G has at least one spanning tree.

Proof of the theorem. Assume that G has order n with $n \geqslant 3$ (since otherwise G is already a tree and the conclusion is obvious).

We can prove the theorem using induction in the size of G.

• **Base Case:** *What is the minimum size to consider here?* Recall that the minimum size of a connected graph on n vertices is $n - 1$. But then, as we saw in Theorem 1a of Lecture 7, if we consider a connected graph G of order n with precisely $n - 1$ edges, this graph will be a tree, and hence the entire graph G will be a spanning tree of itself.

Important Observation

Theorem

Every connected graph G has at least one spanning tree.

Proof of the theorem. Assume that G has order n with $n \geqslant 3$ (since otherwise G is already a tree and the conclusion is obvious).

We can prove the theorem using induction in the size of G.

• **Base Case:** *What is the minimum size to consider here?* Recall that the minimum size of a connected graph on n vertices is $n-1$. But then, as we saw in Theorem 1a of Lecture 7, if we consider a connected graph G of order n with precisely $n-1$ edges, this graph will be a tree, and hence the entire graph G will be a spanning tree of itself.

• **Induction Step:** Assume now that, for some s with $n-1 \leqslant s < \binom{n}{2}$, we have already shown that

every connected graph H on n vertices
which has size s has at least one spanning tree.

Important Observation

Theorem

Every connected graph G has at least one spanning tree.

Proof of the theorem. Assume that G has order n with $n \geqslant 3$ (since otherwise G is already a tree and the conclusion is obvious).

We can prove the theorem using induction in the size of G.

• **Base Case:** *What is the minimum size to consider here?* Recall that the minimum size of a connected graph on n vertices is $n-1$. But then, as we saw in Theorem 1a of Lecture 7, if we consider a connected graph G of order n with precisely $n-1$ edges, this graph will be a tree, and hence the entire graph G will be a spanning tree of itself.

• **Induction Step:** Assume now that, for some s with $n-1 \leqslant s < \binom{n}{2}$, we have already shown that

every connected graph H on n vertices
which has size s has at least one spanning tree.

Consider now a connected graph G on the same set of vertices which has size $s+1$. Then $s+1 > n-1$, and hence G cannot be a tree (recall again Thm 1a of Lec 7).

But this implies that there exists at least one edge of G (say edge e_0) which is NOT a bridge (recall also Thm 2a of Lec 10).

Important Observation

Theorem

Every connected graph G has at least one spanning tree.

Proof of the theorem. Assume that G has order n with $n \geqslant 3$ (since otherwise G is already a tree and the conclusion is obvious).

We can prove the theorem using induction in the size of G.

• **Base Case:** *What is the minimum size to consider here?* Recall that the minimum size of a connected graph on n vertices is $n-1$. But then, as we saw in Theorem 1a of Lecture 7, if we consider a connected graph G of order n with precisely $n-1$ edges, this graph will be a tree, and hence the entire graph G will be a spanning tree of itself.

• **Induction Step:** Assume now that, for some s with $n-1 \leqslant s < \binom{n}{2}$, we have already shown that

every connected graph H on n vertices
which has size s has at least one spanning tree.

Consider now a connected graph G on the same set of vertices which has size $s+1$. Then $s+1 > n-1$, and hence G cannot be a tree (recall again Thm 1a of Lec 7).

But this implies that there exists at least one edge of G (say edge e_0) which is NOT a bridge (recall also Thm 2a of Lec 10). Then the subgraph $G - e_0$ will be connected, and will have size equal to $e(G) - 1 = (s+1) - 1 = s$. Thus by the Inductive Hypothesis, $G - e_0$ will have at least one spanning tree T_1.

Important Observation

Theorem

Every connected graph G has at least one spanning tree.

Proof of the theorem. Assume that G has order n with $n \geqslant 3$ (since otherwise G is already a tree and the conclusion is obvious).

We can prove the theorem using induction in the size of G.

• **Base Case:** *What is the minimum size to consider here?* Recall that the minimum size of a connected graph on n vertices is $n-1$. But then, as we saw in Theorem 1a of Lecture 7, if we consider a connected graph G of order n with precisely $n-1$ edges, this graph will be a tree, and hence the entire graph G will be a spanning tree of itself.

• **Induction Step:** Assume now that, for some s with $n-1 \leqslant s < \binom{n}{2}$, we have already shown that

every connected graph H on n vertices
which has size s has at least one spanning tree.

Consider now a connected graph G on the same set of vertices which has size $s+1$. Then $s+1 > n-1$, and hence G cannot be a tree (recall again Thm 1a of Lec 7).

But this implies that there exists at least one edge of G (say edge e_0) which is NOT a bridge (recall also Thm 2a of Lec 10). Then the subgraph $G - e_0$ will be connected, and will have size equal to $e(G) - 1 = (s+1) - 1 = s$. Thus by the Inductive Hypothesis, $G - e_0$ will have at least one spanning tree T_1.

Remark: T_1 is also a spanning tree of G (since $T_1 \subseteq G - e_0 \subseteq G$ and T_1 contains all the vertices in $V(G - e_0) = V(G)$).

'Algorithmic method' suggested by the proof for finding a spanning tree

Let G be a connected graph on n vertices.

- If G is not already a tree, then G contains cycles, and hence it also contains edges which are not bridges.

'Algorithmic method' suggested by the proof for finding a spanning tree

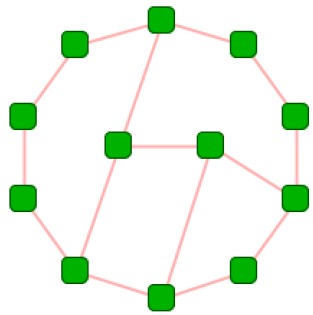
Let G be a connected graph on n vertices.

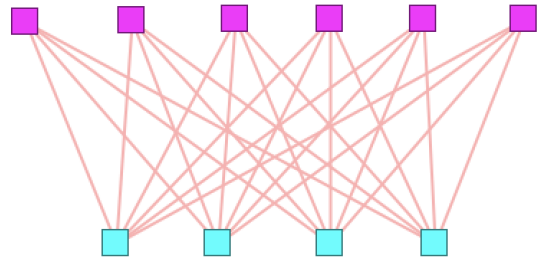
- If G is not already a tree, then G contains cycles, and hence it also contains edges which are not bridges.
- Pick an edge which is not a bridge, and remove it. Check whether the resulting subgraph G_1 is a spanning tree of G. If not, then repeat this step for G_1.

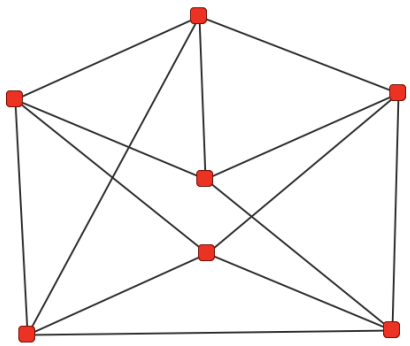
'Algorithmic method' suggested by the proof for finding a spanning tree

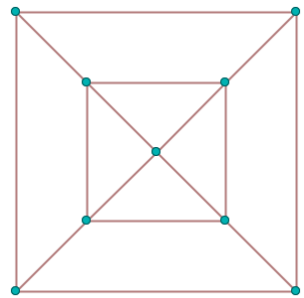
Let G be a connected graph on n vertices.

- If G is not already a tree, then G contains cycles, and hence it also contains edges which are not bridges.
- Pick an edge which is not a bridge, and remove it. Check whether the resulting subgraph G_1 is a spanning tree of G. If not, then repeat this step for G_1.









Practice Question. Find spanning trees for the above graphs.

A variation coming up in applications

Definition

Let $K = (V(K), E(K))$ be a graph. A *weight function* for K is a function

$$w : E(K) \to [0, +\infty)$$

mapping each edge of K to a non-negative real number.

A variation coming up in applications

Definition

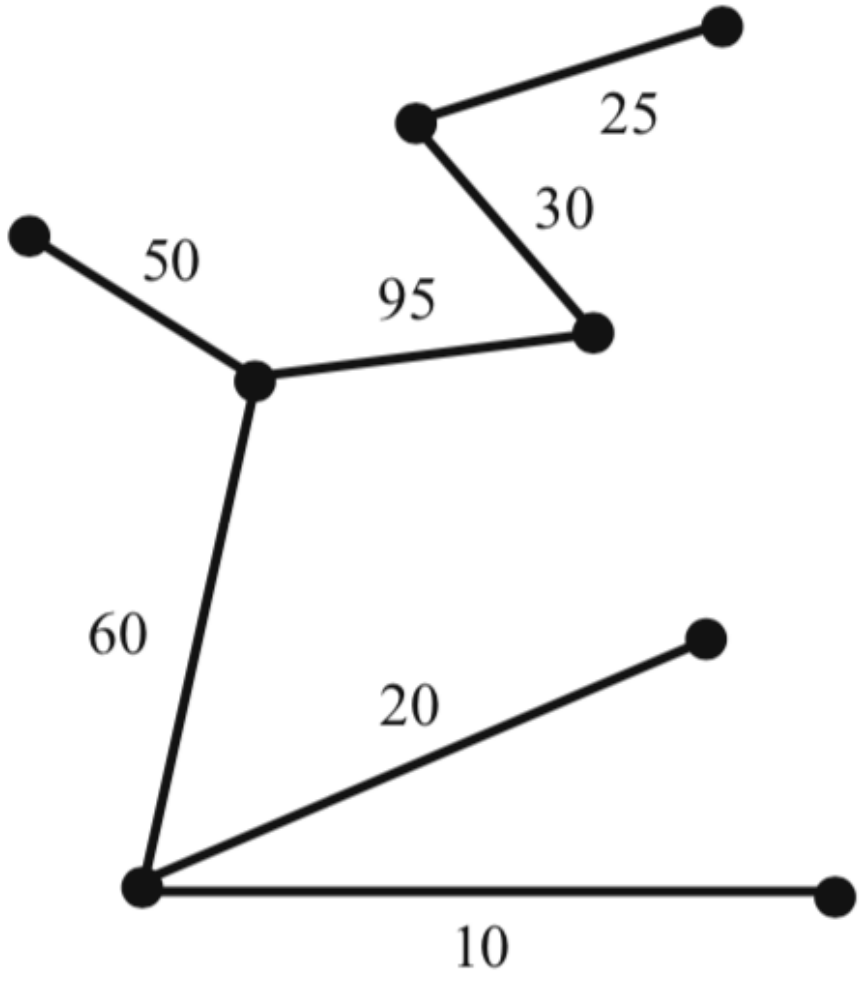
Let $K = (V(K), E(K))$ be a graph. A *weight function* for K is a function

$$w : E(K) \to [0, +\infty)$$

mapping each edge of K to a non-negative real number.

The graph K together with a weight function for it is called a *weighted graph*.

Examples of weighted graphs



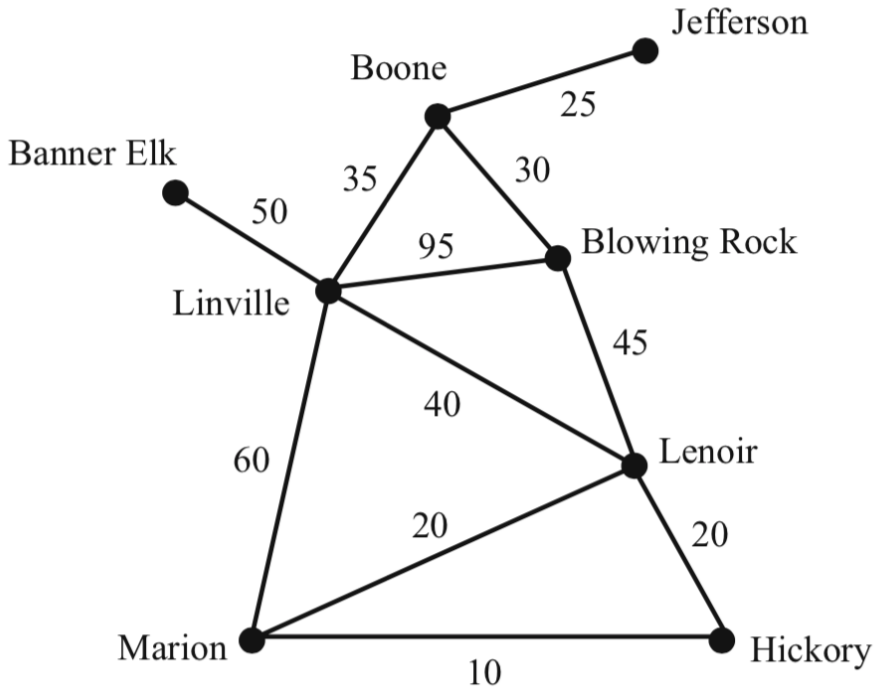
Examples of weighted graphs in applications

Problem that can lead to such an example *(from the Harris-Hirst-Mossinghoff book)*
The North Carolina Department of Transportation (NCDOT) has decided to implement a rapid rail system to serve eight cities in the western part of the state.
— Some of the cities are currently joined by roads or highways, and the state plans to lay the tracks right along (some of) these roads.
— Due to mountainous terrain, some of the roads are steep and curvy, and so laying track along these roads would be difficult and expensive.

Examples of weighted graphs in applications

Problem that can lead to such an example *(from the Harris-Hirst-Mossinghoff book)*
The North Carolina Department of Transportation (NCDOT) has decided to implement a rapid rail system to serve eight cities in the western part of the state.
— Some of the cities are currently joined by roads or highways, and the state plans to lay the tracks right along (some of) these roads.
— Due to mountainous terrain, some of the roads are steep and curvy, and so laying track along these roads would be difficult and expensive.

Due to the above, the NCDOT has hired a consultant to study the roads and to assign a difficulty rating to each one. Here is the graph the consultant has returned:



Examples of weighted graphs in applications

What would be an 'efficient' plan for the NCDOT to follow:
find a spanning tree of this graph.

Examples of weighted graphs in applications

What would be an 'efficient' plan for the NCDOT to follow:
find a spanning tree of this graph.

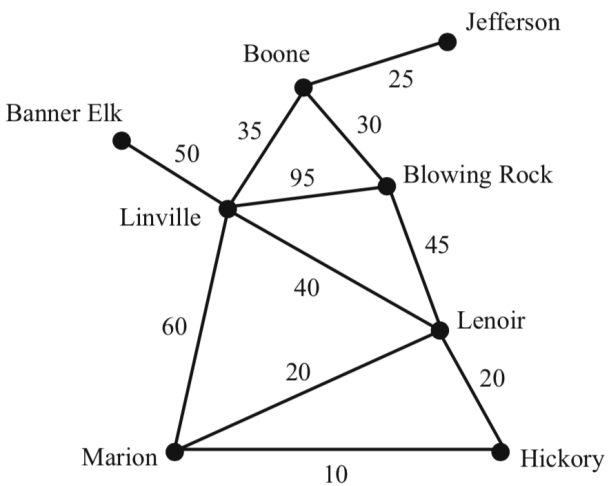
In fact, now it also seems to make sense to 'take even more factors into consideration', and look for a *minimum weight spanning tree*.

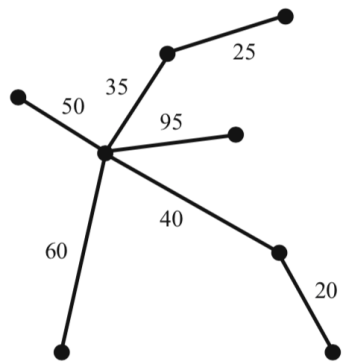
Examples of weighted graphs in applications

What would be an 'efficient' plan for the NCDOT to follow:
find a spanning tree of this graph.

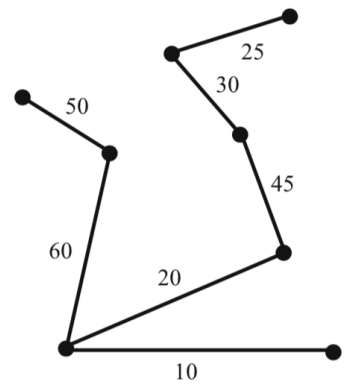
In fact, now it also seems to make sense to 'take even more factors into consideration', and look for a *minimum weight spanning tree*.

Some spanning trees of the above graph:

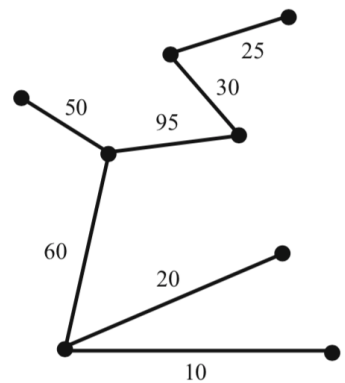




Total Weight = 325



Total Weight = 240



Total Weight = 290

Most ‘efficient’ solution

Most ‘efficient’ plan for the NCDOT to follow:

find a spanning tree of this graph

Most ‘efficient’ solution

Most ‘efficient’ plan for the NCDOT to follow:
find a **minimum weight** spanning tree of this graph

Most ‘efficient’ solution

Most ‘efficient’ plan for the NCDOT to follow:
find a **minimum weight** spanning tree of this graph

(in other words, try to minimise the total cost (*total weight*) of laying the necessary infrastructure).

Most 'efficient' solution

Most 'efficient' plan for the NCDOT to follow:
find a **minimum weight** spanning tree of this graph

(in other words, try to minimise the total cost (*total weight*) of laying the necessary infrastructure).

A problem of this type is usually referred to as

the connector problem.

Algorithms for finding minimum weight spanning trees?

One such algorithm is

Kruskal's algorithm

Suppose you are given a weighted connected graph G, and you are looking for a minimum weight spanning tree of it.

Algorithms for finding minimum weight spanning trees?

One such algorithm is

Kruskal's algorithm

Suppose you are given a weighted connected graph G, and you are looking for a minimum weight spanning tree of it. Then:

1. Begin by finding an edge of minimum weight, and mark it.

Algorithms for finding minimum weight spanning trees?

One such algorithm is

Kruskal's algorithm

Suppose you are given a weighted connected graph G, and you are looking for a minimum weight spanning tree of it. Then:

1. Begin by finding an edge of minimum weight, and mark it.
2. Out of all the edges that remain unmarked **and which do not form a cycle with any of the already marked edges**, pick an edge of minimum weight and mark it.

Algorithms for finding minimum weight spanning trees?

One such algorithm is

Kruskal's algorithm

Suppose you are given a weighted connected graph G, and you are looking for a minimum weight spanning tree of it. Then:

1. Begin by finding an edge of minimum weight, and mark it.
2. Out of all the edges that remain unmarked **and which do not form a cycle with any of the already marked edges**, pick an edge of minimum weight and mark it.
3. If the set of the already marked edges gives a spanning tree of G, then terminate the process. Otherwise, return to Step 2.

Algorithms for finding minimum weight spanning trees?

One such algorithm is

Kruskal's algorithm

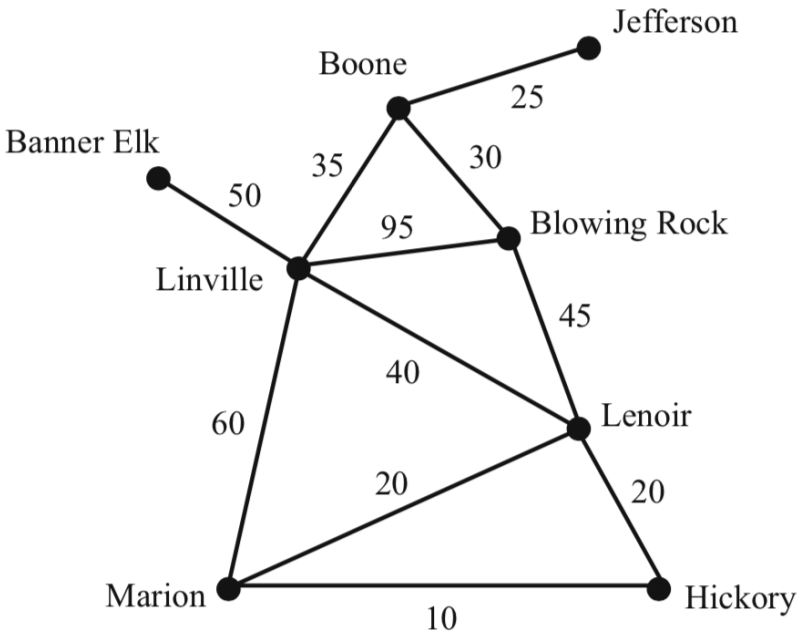
Suppose you are given a weighted connected graph G, and you are looking for a minimum weight spanning tree of it. Then:

1. Begin by finding an edge of minimum weight, and mark it.
2. Out of all the edges that remain unmarked **and which do not form a cycle with any of the already marked edges**, pick an edge of minimum weight and mark it.
3. If the set of the already marked edges gives a spanning tree of G, then terminate the process. Otherwise, return to Step 2.

Theorem (analogous to above)

For every weighted connected graph G, Kruskal's algorithm gives a minimum weight spanning tree.

Applying Kruskal's algorithm to the above example



Different instances of the connector problem

Different instances of the connector problem

from the Balakrishnan-Ranganathan book

Problem 1. Various cities in a country are to be linked by means of roads. Given the various possibilities of connecting the cities and the costs involved, what is the most economical way of laying roads so that in the resulting road network any two cities are connected by a chain of roads?

Similar problems involve designing railroad networks, or water-line transports.

Different instances of the connector problem

from the Balakrishnan-Ranganathan book

Problem 1. Various cities in a country are to be linked by means of roads. Given the various possibilities of connecting the cities and the costs involved, what is the most economical way of laying roads so that in the resulting road network any two cities are connected by a chain of roads?

Similar problems involve designing railroad networks, or water-line transports.

Problem 2. A layout for a housing colony in a city is to be prepared. Various locations of the colony are to be linked by roads. Given the various possibilities of linking the locations and their costs, what is the minimum-cost layout so that any two locations are connected by a chain of roads?

Different instances of the connector problem

from the Balakrishnan-Ranganathan book

Problem 1. Various cities in a country are to be linked by means of roads. Given the various possibilities of connecting the cities and the costs involved, what is the most economical way of laying roads so that in the resulting road network any two cities are connected by a chain of roads?

Similar problems involve designing railroad networks, or water-line transports.

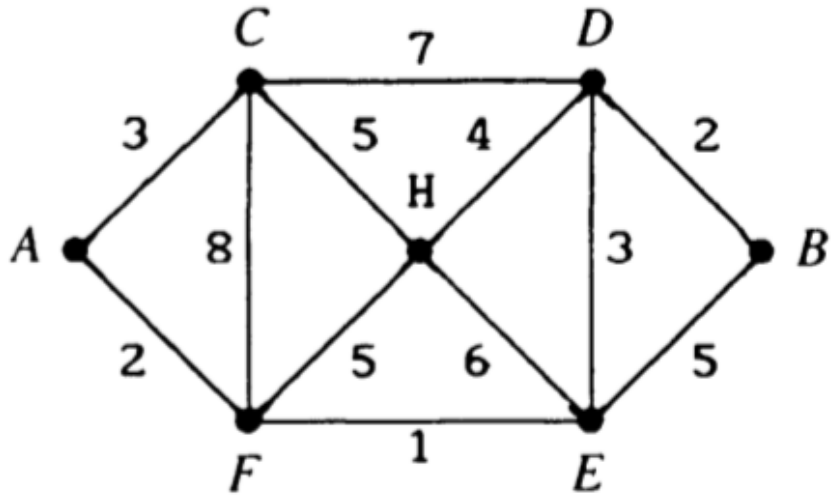
Problem 2. A layout for a housing colony in a city is to be prepared. Various locations of the colony are to be linked by roads. Given the various possibilities of linking the locations and their costs, what is the minimum-cost layout so that any two locations are connected by a chain of roads?

Problem 3. A layout for the electrical wiring of a building is to be prepared. Given the costs of the various possibilities, what is the minimum-cost layout?

Other similar problems which have motivated interesting questions in Graph Theory (which in turn has led to further understanding of the applications too)

The shortest path problem

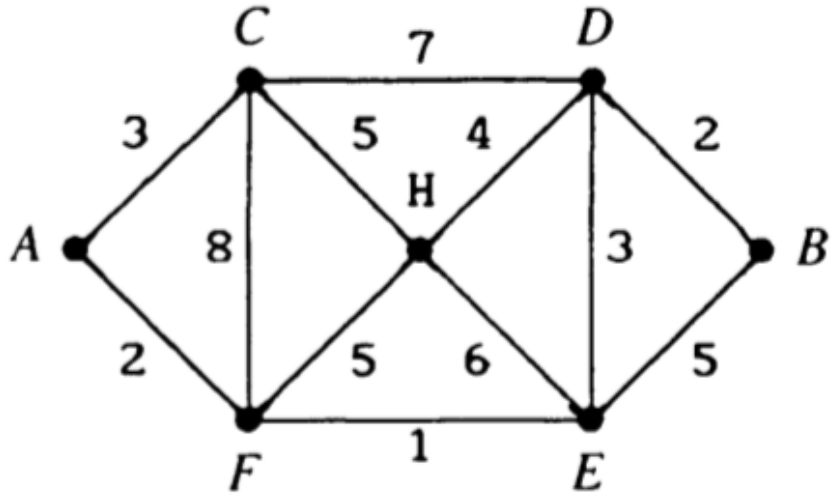
Let G_0 be a weighted connected graph, and let A, B be two different vertices of G_0.



from the Balakrishnan-Ranganathan book

The shortest path problem

Let G_0 be a weighted connected graph, and let A, B be two different vertices of G_0.



from the Balakrishnan-Ranganathan book

Since G_0 is connected, we know that there are paths in G_0 taking us from A to B.

Question. How do we find such a path which also has minimum weight?

The travelling salesman problem

Let G_0 be a weighted connected graph whose vertices represent different cities that a salesman wants to visit, which are connected by, say, roads and highways, or by train routes, or by airline routes, represented by the edges of the graph *(with each edge weight capturing the cost or distance of travel from one city - endvertex to the other city - endvertex which this edge joins).*

The travelling salesman problem

Let G_0 be a weighted connected graph whose vertices represent different cities that a salesman wants to visit, which are connected by, say, roads and highways, or by train routes, or by airline routes, represented by the edges of the graph *(with each edge weight capturing the cost or distance of travel from one city - endvertex to the other city - endvertex which this edge joins).*

Question 1. What is the most cost-efficient (or time-efficient) way for the salesman to visit all the cities and finally return to the city which he is supposed to start from?

The travelling salesman problem

Let G_0 be a weighted connected graph whose vertices represent different cities that a salesman wants to visit, which are connected by, say, roads and highways, or by train routes, or by airline routes, represented by the edges of the graph *(with each edge weight capturing the cost or distance of travel from one city - endvertex to the other city - endvertex which this edge joins).*

Question 1. What is the most cost-efficient (or time-efficient) way for the salesman to visit all the cities and finally return to the city which he is supposed to start from?

Question 2. Is there a way for the salesman to visit all the cities but not pass by any city more than once (except perhaps to return to the city where he starts from at the end of his trip)?

An efficient scenic route...

The city of Königsberg, Prussia was set on the Pregel River, and included two large islands that were connected to each other and the mainland by seven bridges.

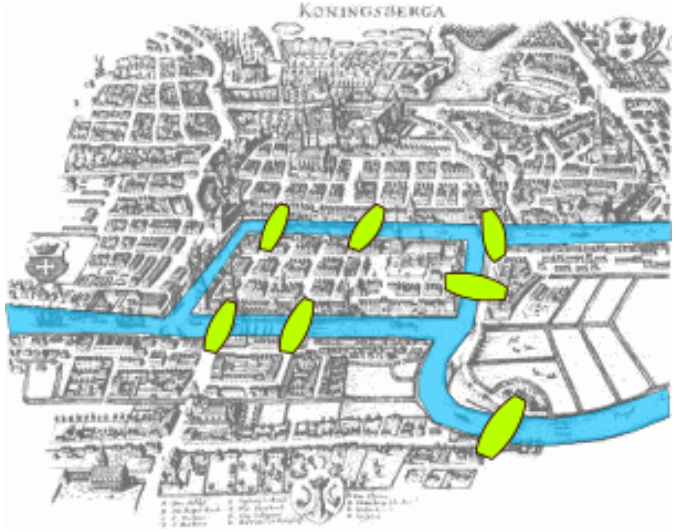


Image from Wikipedia: Map of the city in Leonhard Euler's time showing the actual layout of the seven bridges, and highlighting the river Pregel and the bridges.

An efficient scenic route...

The city of Königsberg, Prussia was set on the Pregel River, and included two large islands that were connected to each other and the mainland by seven bridges.

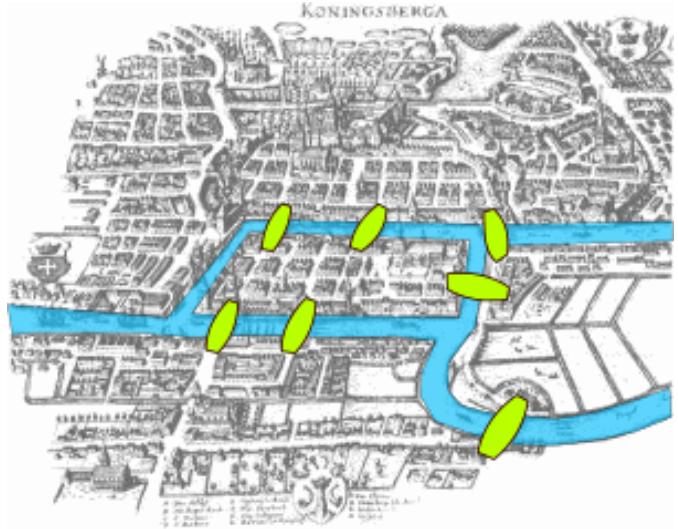


Image from Wikipedia: Map of the city in Leonhard Euler's time showing the actual layout of the seven bridges, and highlighting the river Pregel and the bridges.

People spent time trying to discover a way in which they could cross each bridge exactly once before returning to the point / place in the city that they started from.

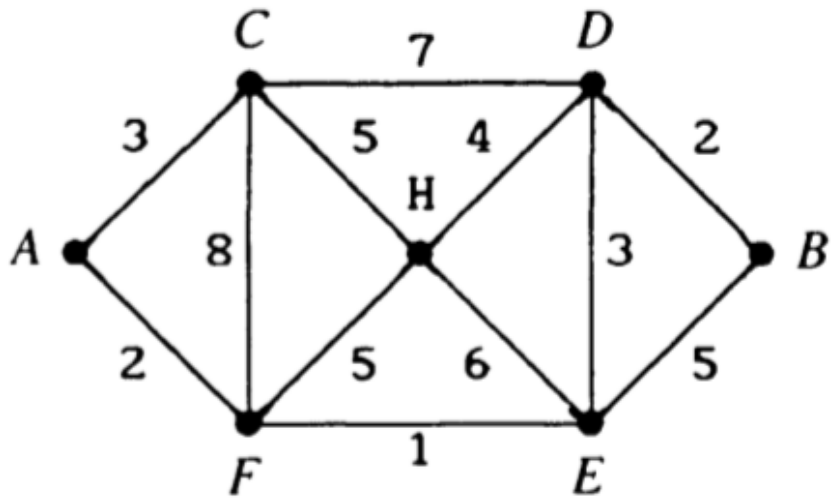
An efficient scenic route...

This problem is now known as the *Königsberg Bridges Problem*, or equivalently the *Euler-Königsberg Bridges Problem*, since it was Leonhard Euler who determined in 1735 that finding such a way is not possible.

This result of his is regarded as essentially giving rise to Graph Theory as an area of Mathematics.

Next time:
we start with the shortest path problem

Recall that we have a weighted connected graph G_0 of order n (here $n = 7$), and want to find a minimum weight path connecting, say, the vertices A and B below.



Weight matrix of a weighted graph

In the process of solving the shortest path problem, we need to consider the *weight matrix* W_{G_0} of G_0.

Assume that the vertex set of G_0 is $\{v_1, v_2, \ldots, v_n\}$; then $W_{G_0} = \left(w_{i,j}\right)_{i,j}$ is an $n \times n$ matrix satisfying

$$w_{i,j} = \begin{cases} \infty & \text{if } i = j, \text{ or the vertices } v_i \text{ and } v_j \text{ are non-adjacent} \\ \text{weight of the edge } \{v_i,\ v_j\} & \text{if } v_i \text{ and } v_j \text{ are adjacent vertices} \end{cases}$$

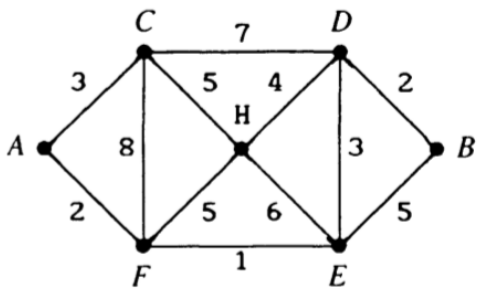
Weight matrix of a weighted graph

In the process of solving the shortest path problem, we need to consider the *weight matrix* W_{G_0} of G_0.

Assume that the vertex set of G_0 is $\{v_1, v_2, \ldots, v_n\}$; then $W_{G_0} = \left(w_{i,j}\right)_{i,j}$ is an $n \times n$ matrix satisfying

$$w_{i,j} = \begin{cases} \infty & \text{if } i = j, \text{ or the vertices } v_i \text{ and } v_j \text{ are non-adjacent} \\ \text{weight of the edge } \{v_i,\ v_j\} & \text{if } v_i \text{ and } v_j \text{ are adjacent vertices} \end{cases}$$

For the example above, we have



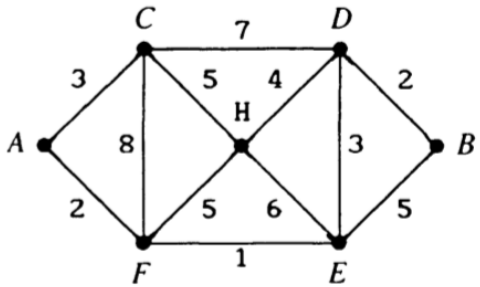
Weight matrix of a weighted graph

In the process of solving the shortest path problem, we need to consider the *weight matrix* W_{G_0} of G_0.

Assume that the vertex set of G_0 is $\{v_1, v_2, \ldots, v_n\}$; then $W_{G_0} = \left(w_{i,j}\right)_{i,j}$ is an $n \times n$ matrix satisfying

$$w_{i,j} = \begin{cases} \infty & \text{if } i = j, \text{ or the vertices } v_i \text{ and } v_j \text{ are non-adjacent} \\ \text{weight of the edge } \{v_i,\ v_j\} & \text{if } v_i \text{ and } v_j \text{ are adjacent vertices} \end{cases}$$

For the example above, we have



$$W_{G_0} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ H \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & E & F & H \end{array} \\ \begin{pmatrix} \infty & \infty & 3 & \infty & \infty & 2 & \infty \\ \infty & \infty & \infty & 2 & 5 & \infty & \infty \\ 3 & \infty & \infty & 7 & \infty & 8 & 5 \\ \infty & 2 & 7 & \infty & 3 & \infty & 4 \\ \infty & 5 & \infty & 3 & \infty & 1 & 6 \\ 2 & \infty & 8 & \infty & 1 & \infty & 5 \\ \infty & \infty & 5 & 4 & 6 & 5 & \infty \end{pmatrix} \end{array}.$$