Spanning trees.

Recall from last time

G connected is a Tree

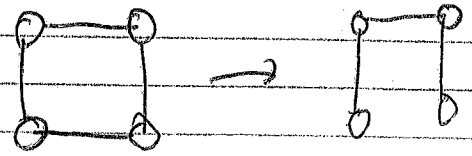
(A) Between any two point, there is only one path ~~joining~~ any circuit

~~Any two paths joining those two~~

(A) Has no (non trivial) circuits (no holes)

(B) #edges = #vertices − 1.

(C) Removing any edge separates the graph into two parts

(C) 

(C) ⇒ (A) If G has a loop, property (C) does not hold

(A) ⟺ (B) Euler characteristic

(A) ⇒ (C)

no other path from B → A.
⇒ AB is a bridge.

- The strongest characterisation?

A B or C.

- The most efficient way to check that a graph is a tree.

⇒ (B) Redondancy R = # edges − (# vertices − 1) R = 0

(B) holds.

Example: Take a list of edges.

AB AE BC DA FB FD.

Tree? • Vertices A B C D E F 6

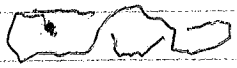
• Connected Yes.

• Edges 6

vertices − 1 = 5 ≠ # edges = 6.

R = 1 edges redondant.

More edges ⇒ More redondancy ⇒ more circuits.



Why tree?

Pb:
- Pass the pen to one of you.
- You have to setup a route between neighboors.

minimizing the number of route you setup.

Constraints: You have to teach the method you employ to someone who cannot figure it out, but who can apply simple directives.

Formalization

Start with graph

G { students = vertices.
neighbor = edges.

Find subgraph of G minimizing the number of edges + connecting all vertices.

Real world example:

students ⟷ computers

edges ⟷ link between computers

Poiss ⇒ Algorithm to setup a network without linking all the computers between them.

Formalization:

G graph Find algorithm to find a good subgraph.

The algorithm constructs a spanning tree