Lecturer: Andrew Parkes
http://www.cs.nott.ac.uk/~pszajp/

COMP2054 ADE
Minimum Spanning Trees

Spanning Tree

- Input: connected, undirected graph
- Output: a tree which connects all vertices in the graph using only the edges present in the graph

Minimum Spanning Tree

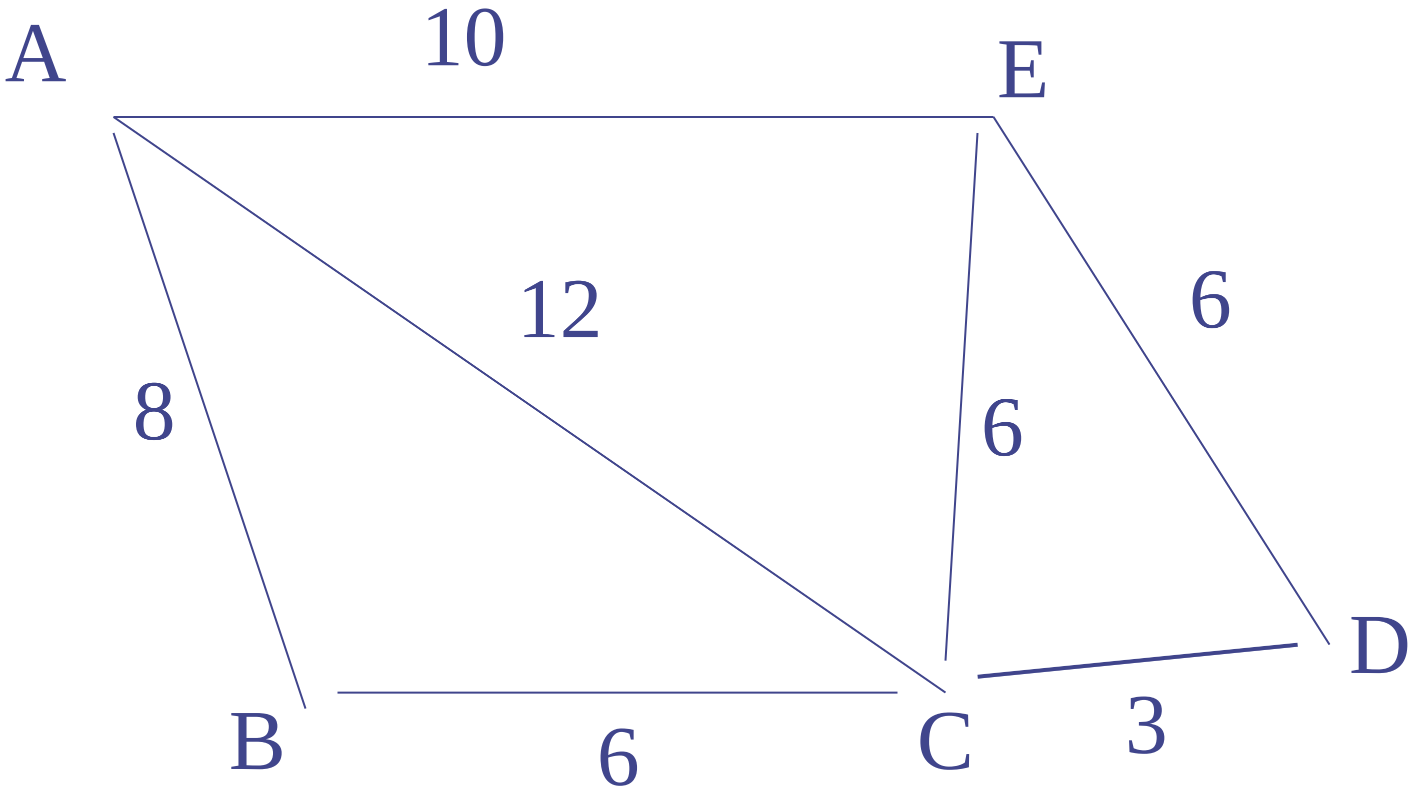
- Input: connected, undirected, weighted graph
- Output: a spanning tree
  - (connects all vertices in the graph using only the edges present in the graph)
  - and is minimum in the sense that the sum of weights of the edges is the smallest possible for any spanning tree

MST Usages

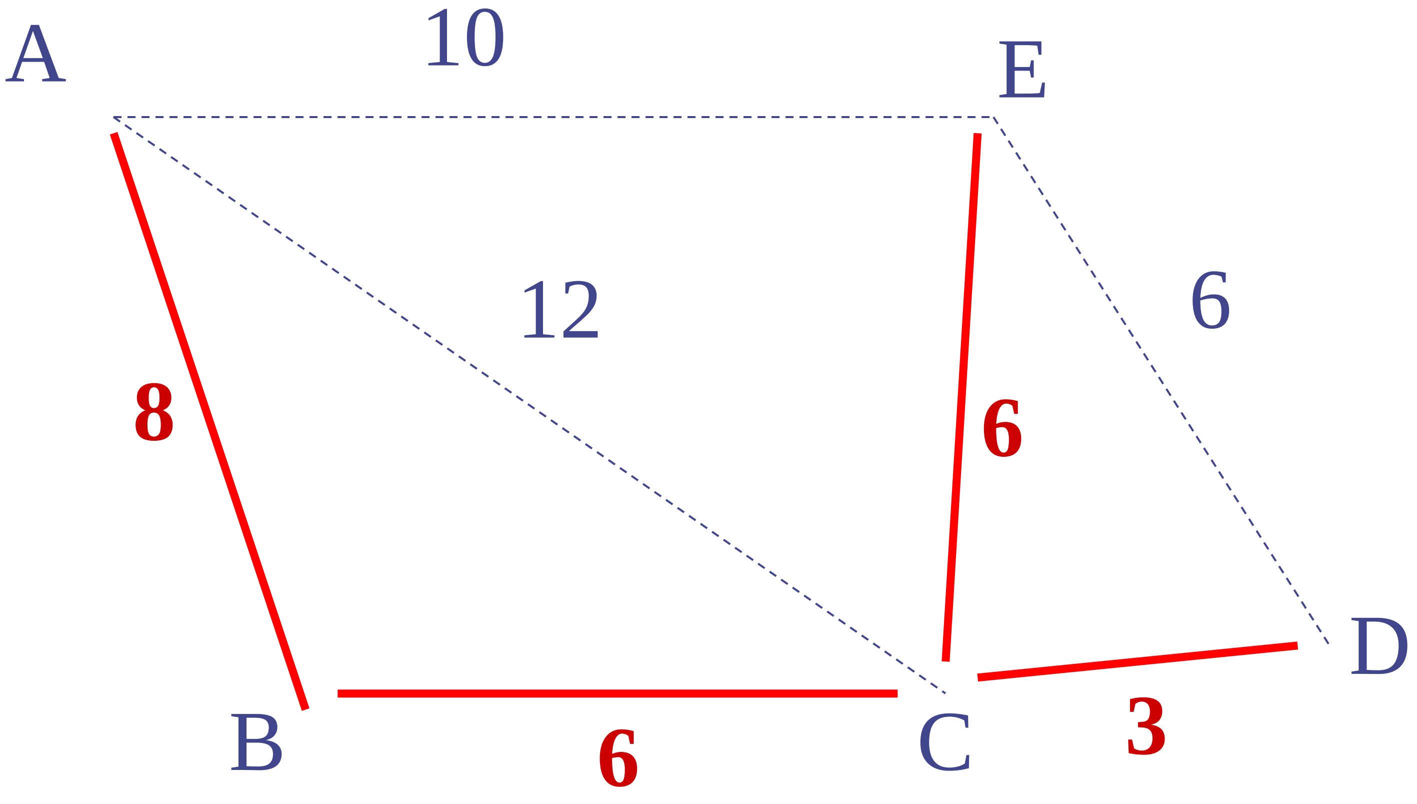
- Gas and water pipelines, optic fibres networks
  - Simply need to build a network to reach every house but minimizing total cost of construction
    - Nodes are supply/delivery locations
    - Edges are “pipes” or “cables” that we can build, and the weight is their cost
    - Aim is to build the cheapest system that meets the supply requirements
      - (In real-world might want to modify to allow redundancy in case of breakages)

- As part of other graph algorithms
  - E.g. a “Christofides algorithm” to approximately solve the TSP exploits constructing an MST and then converting to a tour
    https://en.wikipedia.org/wiki/Christofides_algorithm#Algorithm

Example: graph

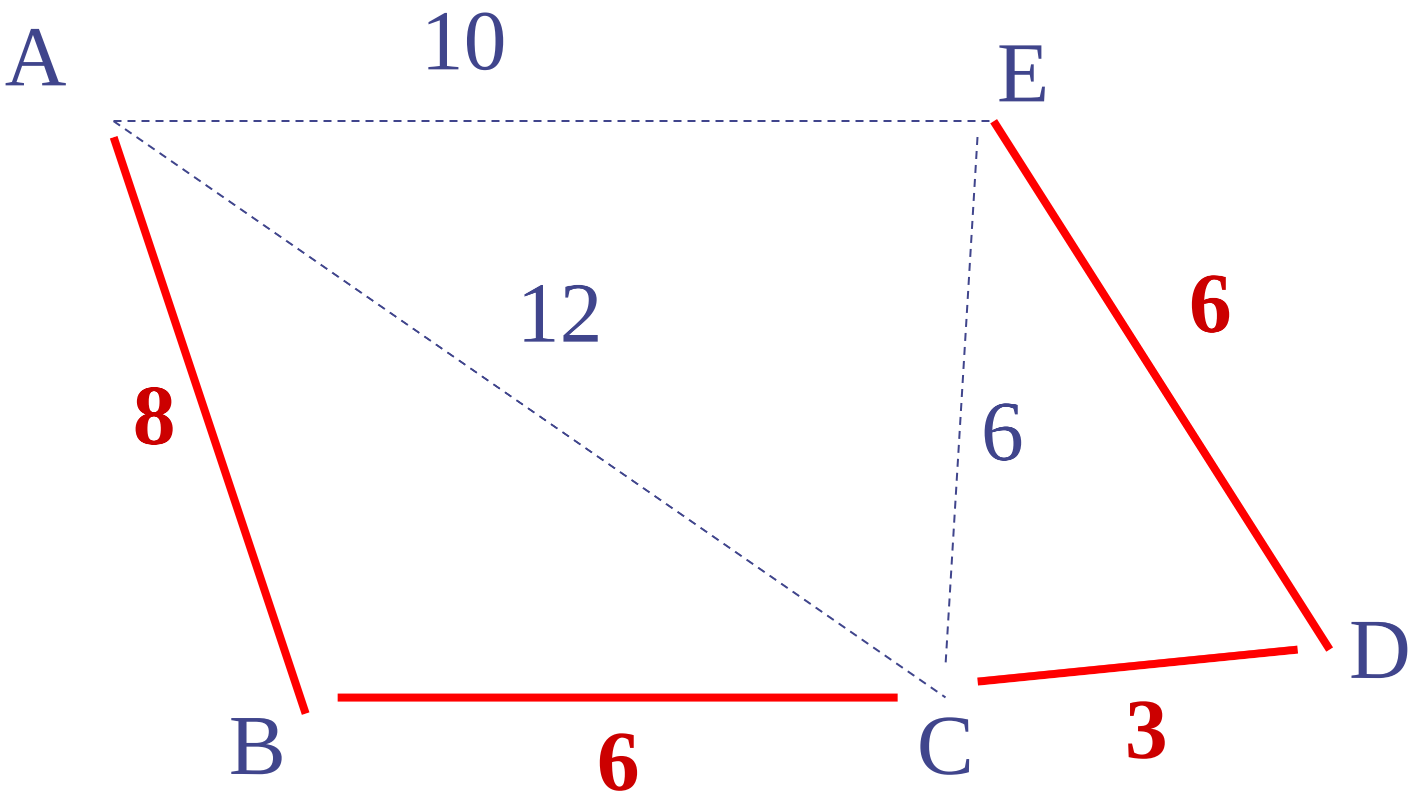


Example: a MST (cost 23)



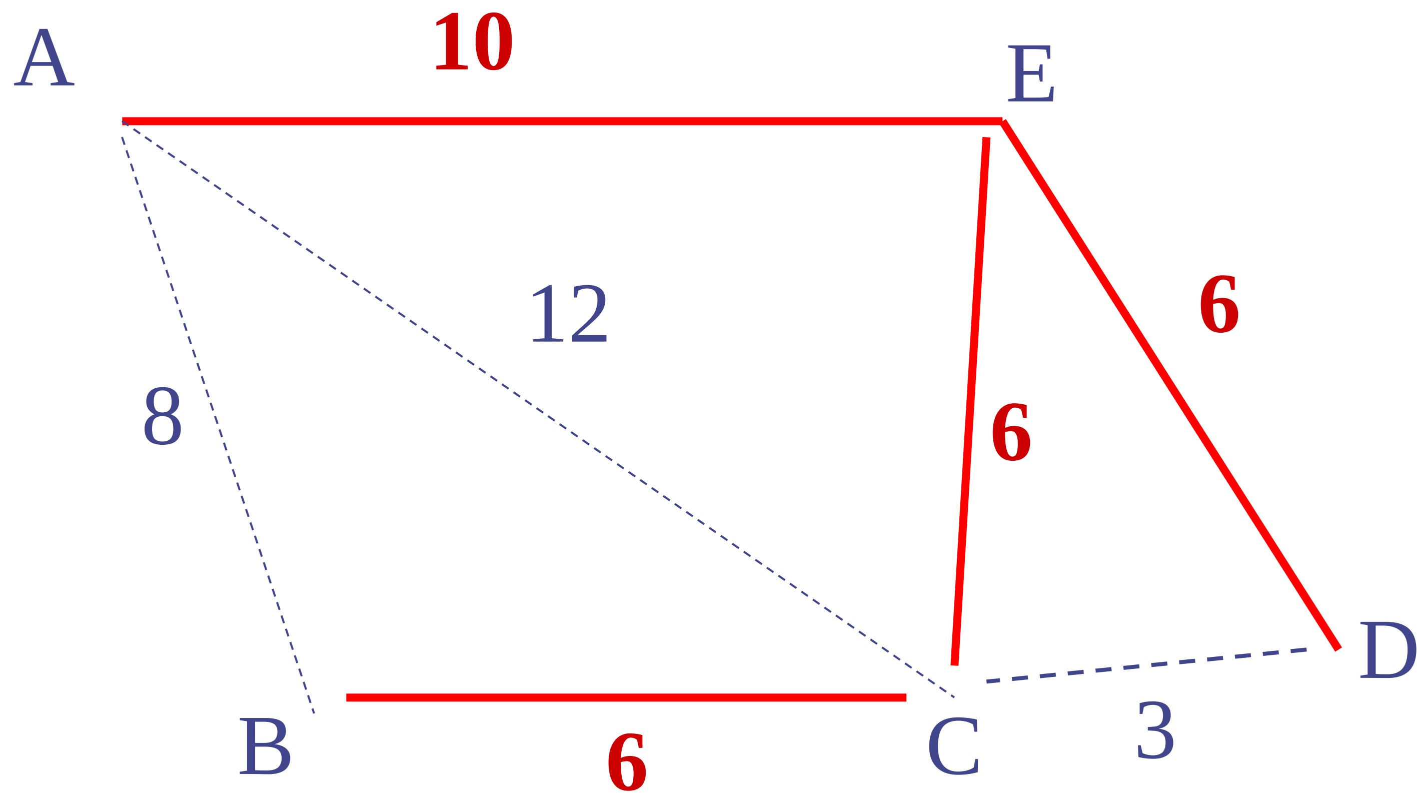
Note: This MST is NOT a path !!!!

Example: another MST (cost 23)



Note: An MST can (sometimes) be a path, but it is still not necessarily the shortest path between the endpoints

Example: not MST (cost 28)



Note: The above is a “spanning tree”, but it is not “minimum”

Why MST is a tree

- We really want a minimum spanning sub-graph
  - that is, a subset of the edges that is connected and that contains every node
- (Assuming all weights are non-negative)
  If the graph has a cycle then we can remove an edge of the cycle, and the graph will still be connected, and will have a smaller weight
- If a graph is connected and acyclic then it is a tree

An MST is (generally) a TREE

- Do not confuse a minimum **TREE** with a "minimum" (shortest) **PATH**
  - Finding the shortest path that goes through all the nodes is a different problem (roughly "TSP" / "Hamiltonian cycle") from the MST (and much harder)
  - It is also different from shortest path between two nodes
- (Many have people confused these on many exams).

Usages?

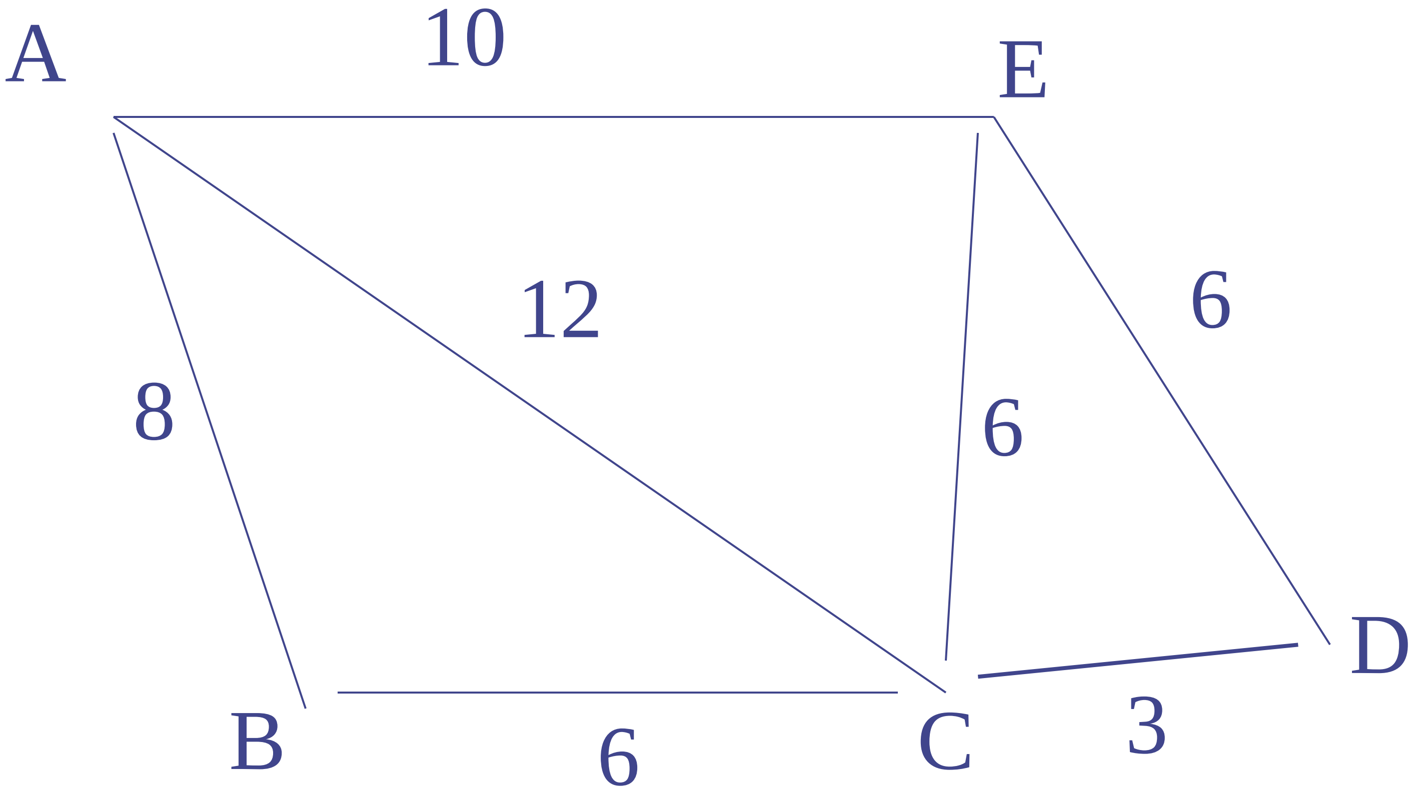
- Gas/water (main) pipelines
  - Connect every house
  - Minimise the total length of all the pipes
  - Follow roads
- Other network problems
- As part of some other algorithm
  - "first find an MST, and then …"

Prim’s algorithm

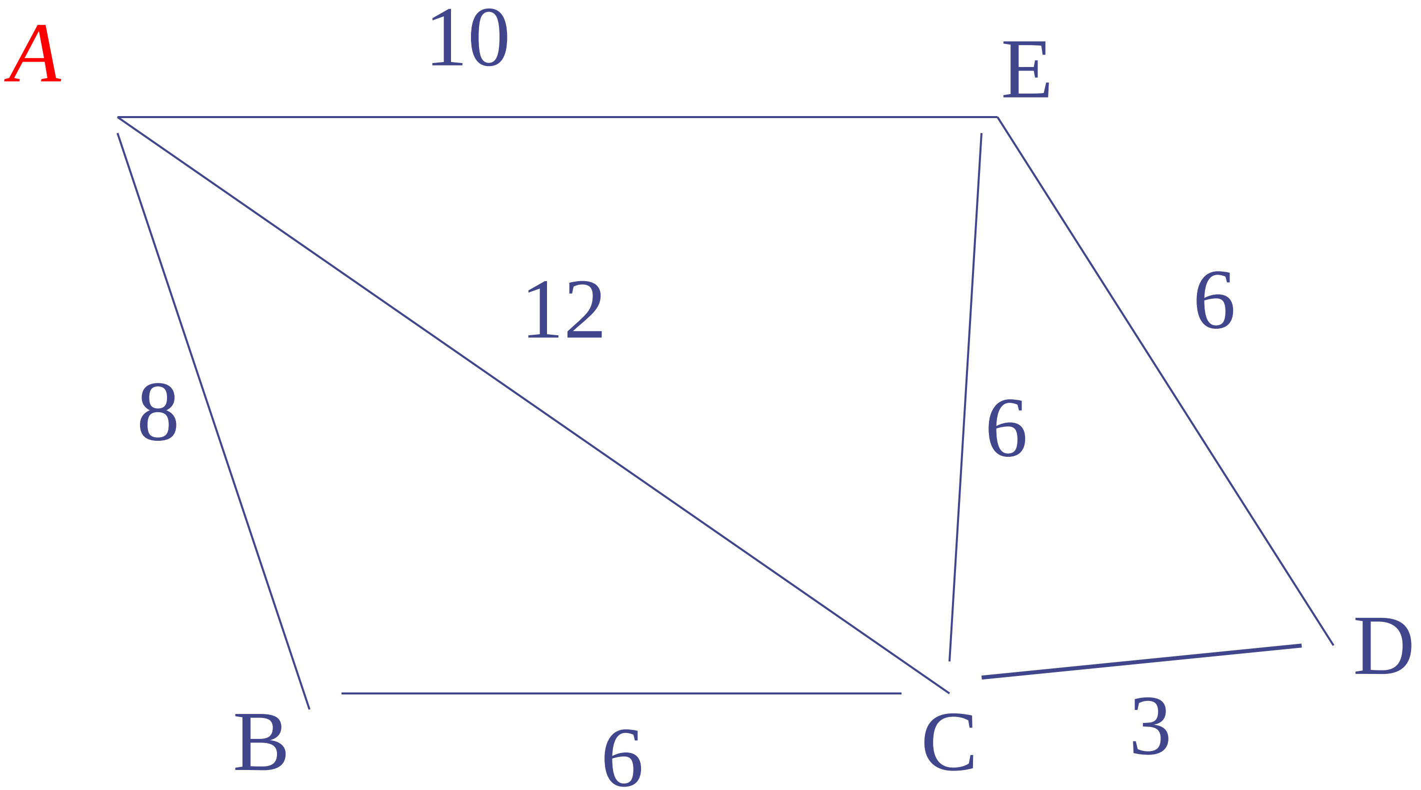
To construct an MST:

- Start by picking any vertex M
- Choose the shortest edge from M to any other vertex N
- Add edge (M,N) to the MST
- Loop:
  - Continue to add at every step a shortest edge from a vertex in MST to a vertex outside, until all vertices are in MST
  - (If there are multiple shortest edges, then can take any arbitrary one)

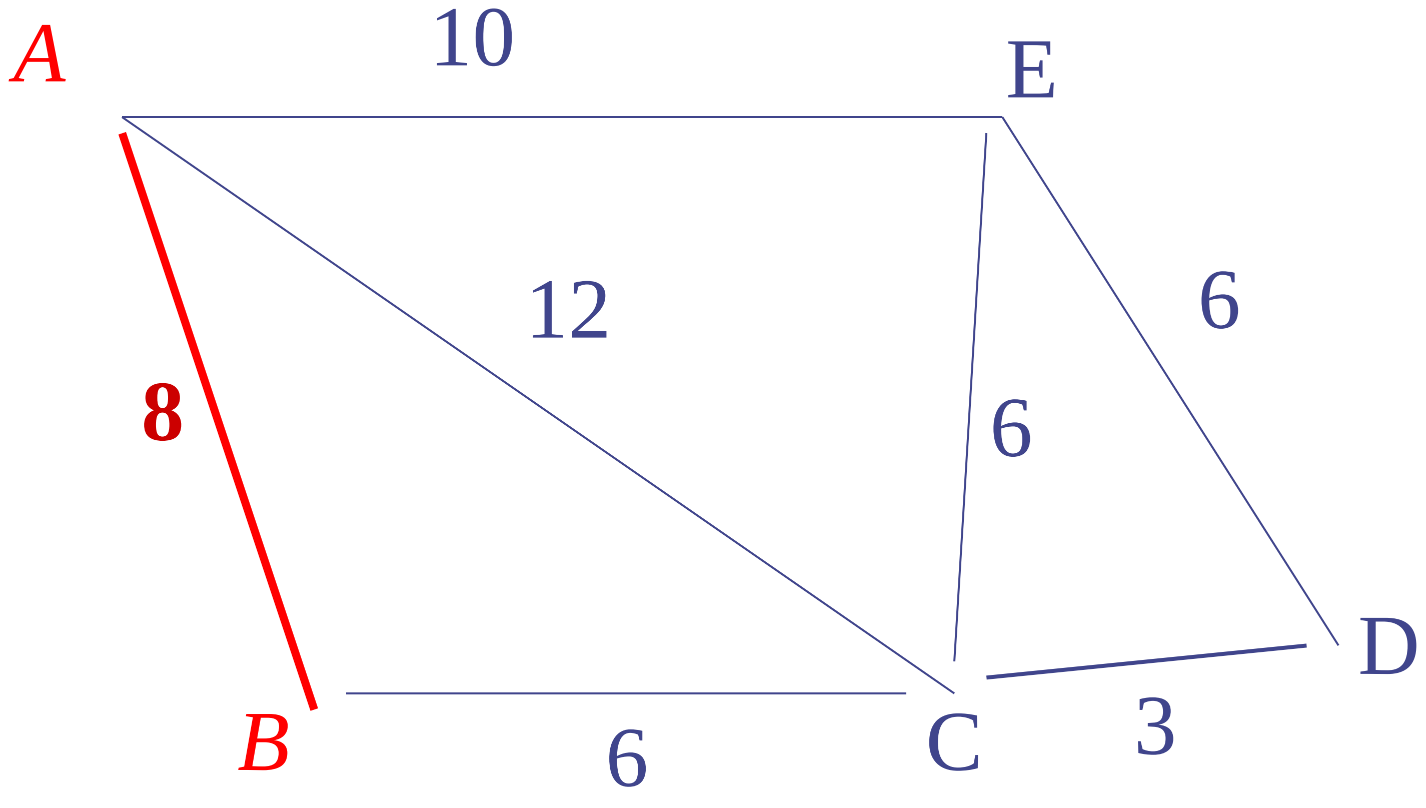
Example



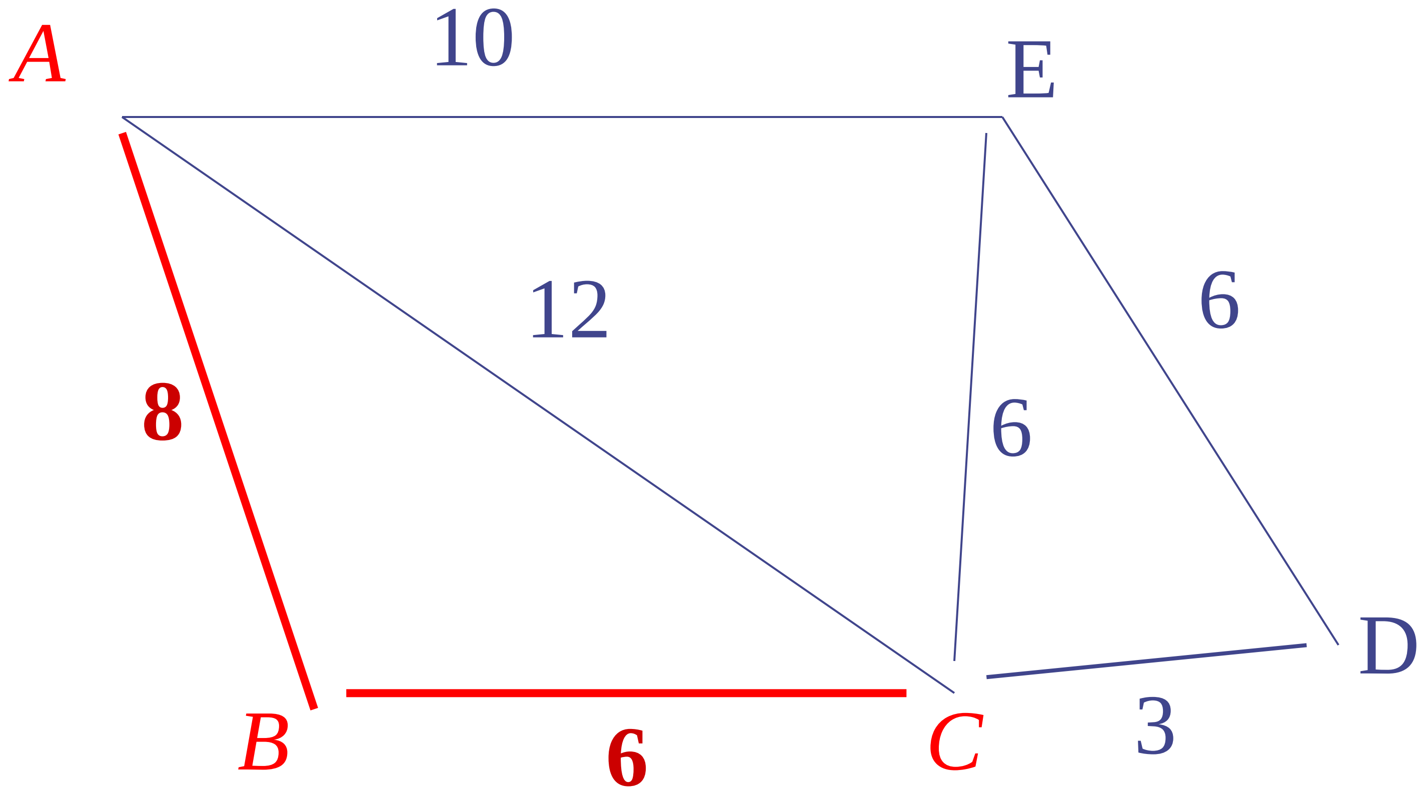
Example



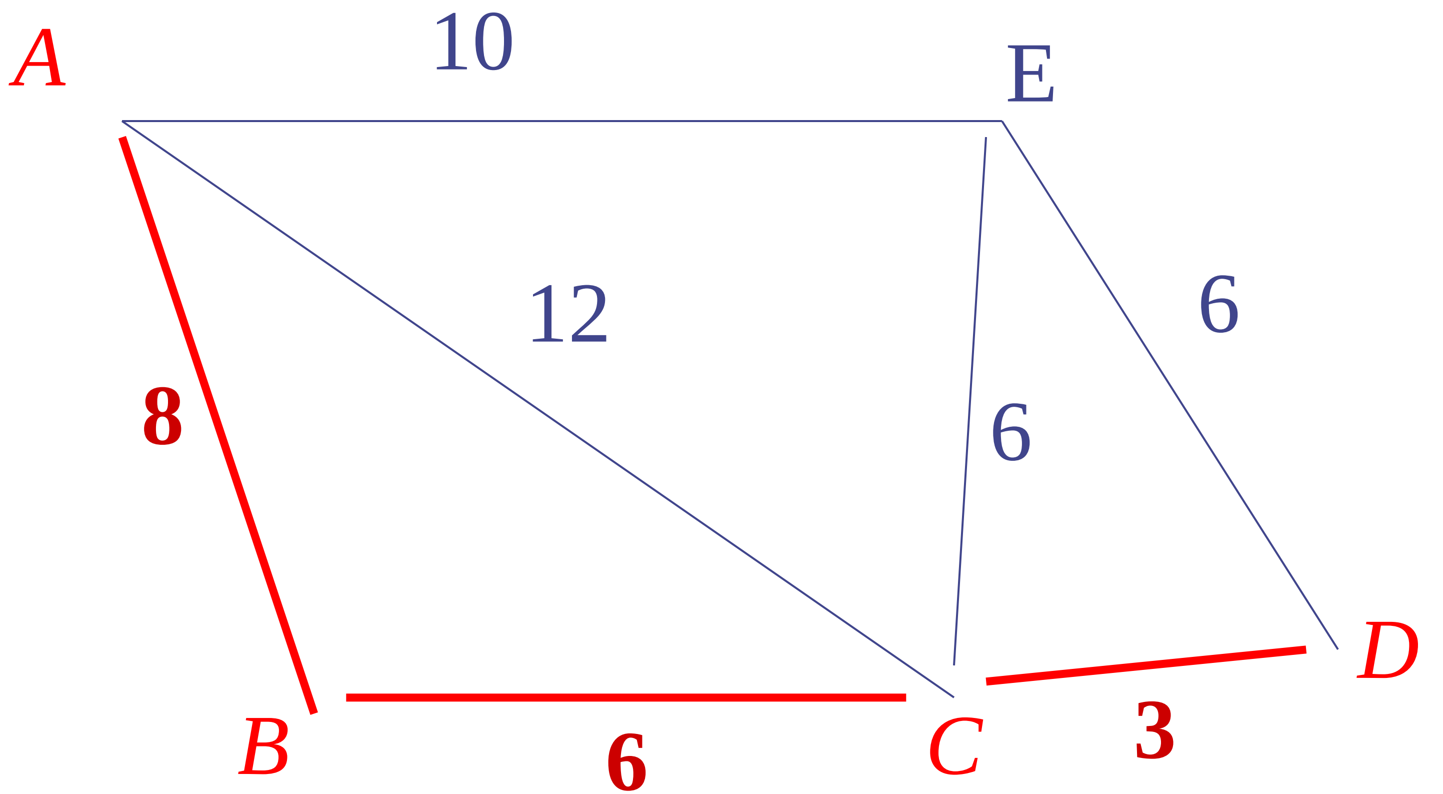
Example



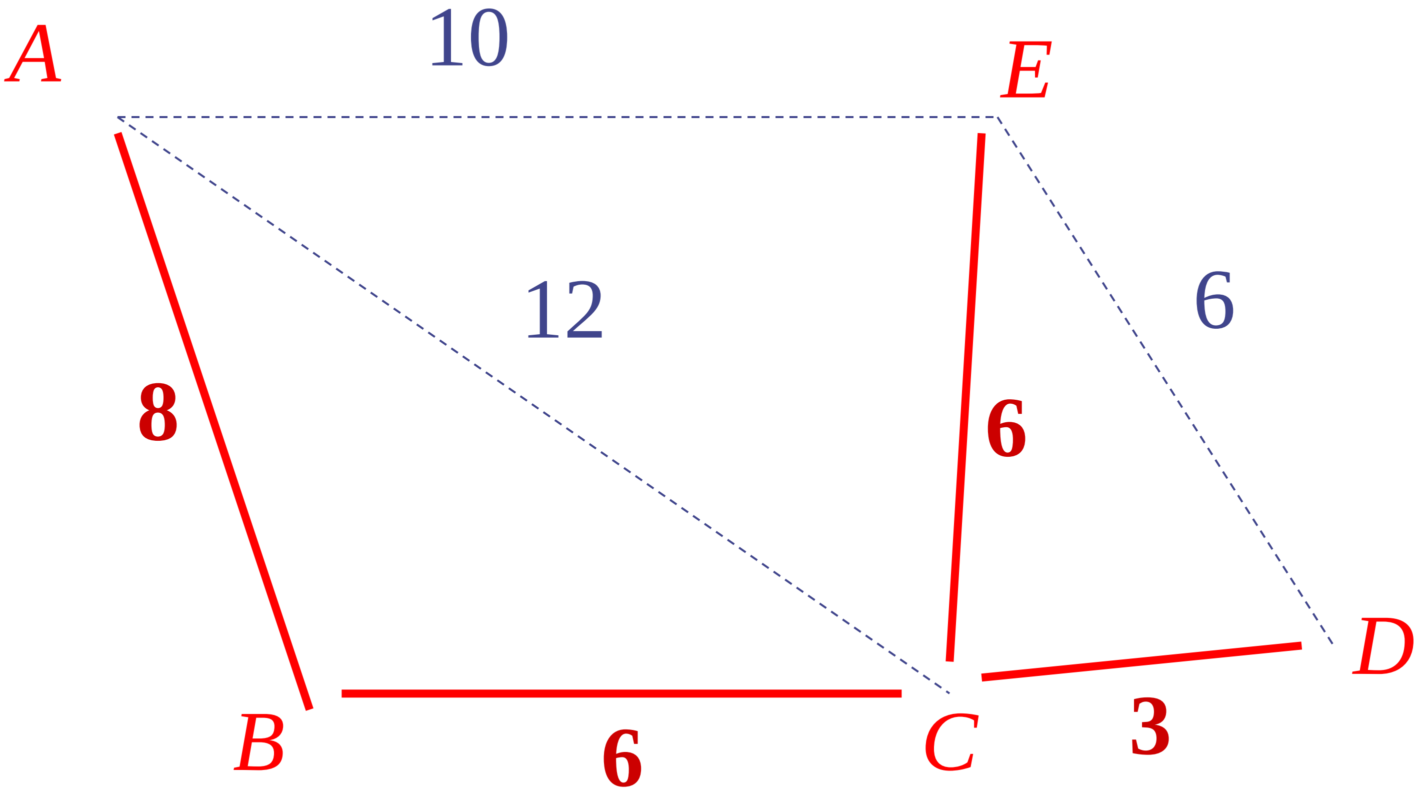
Example



Example



Example

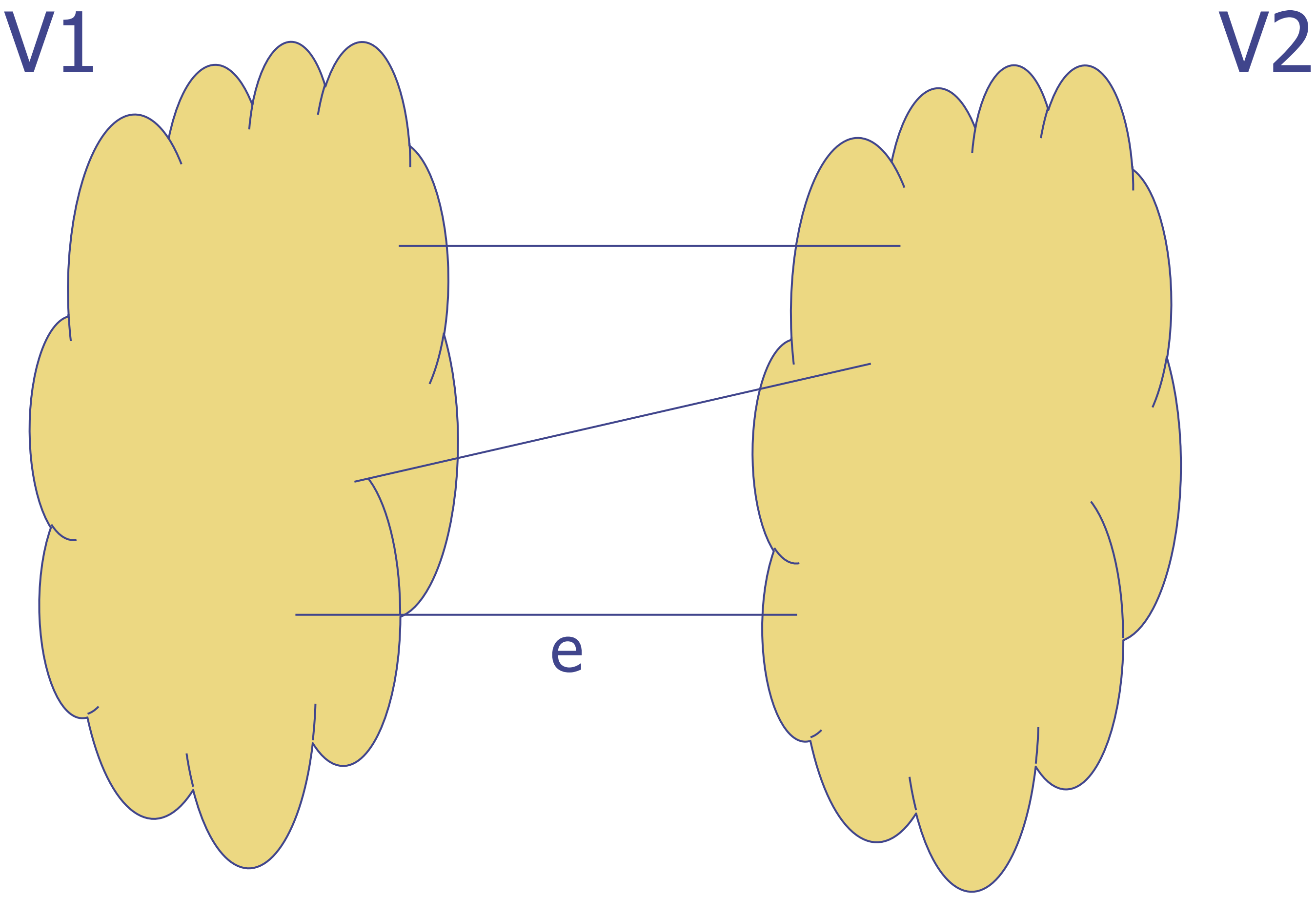


Why is this optimal!?

- [Goodrich/Tamassia Textbook]. Proposition 13.25 (Section 13.7)

- "Let G be a weighted connected graph, and
  - let V1 and V2 be a partition of the vertices of G into two disjoint non-empty sets.
  - Furthermore, let e be an edge with minimum weight from among those with one endpoint in V1 and the other in V2.
  - There is an MST that has e as one of its edges."

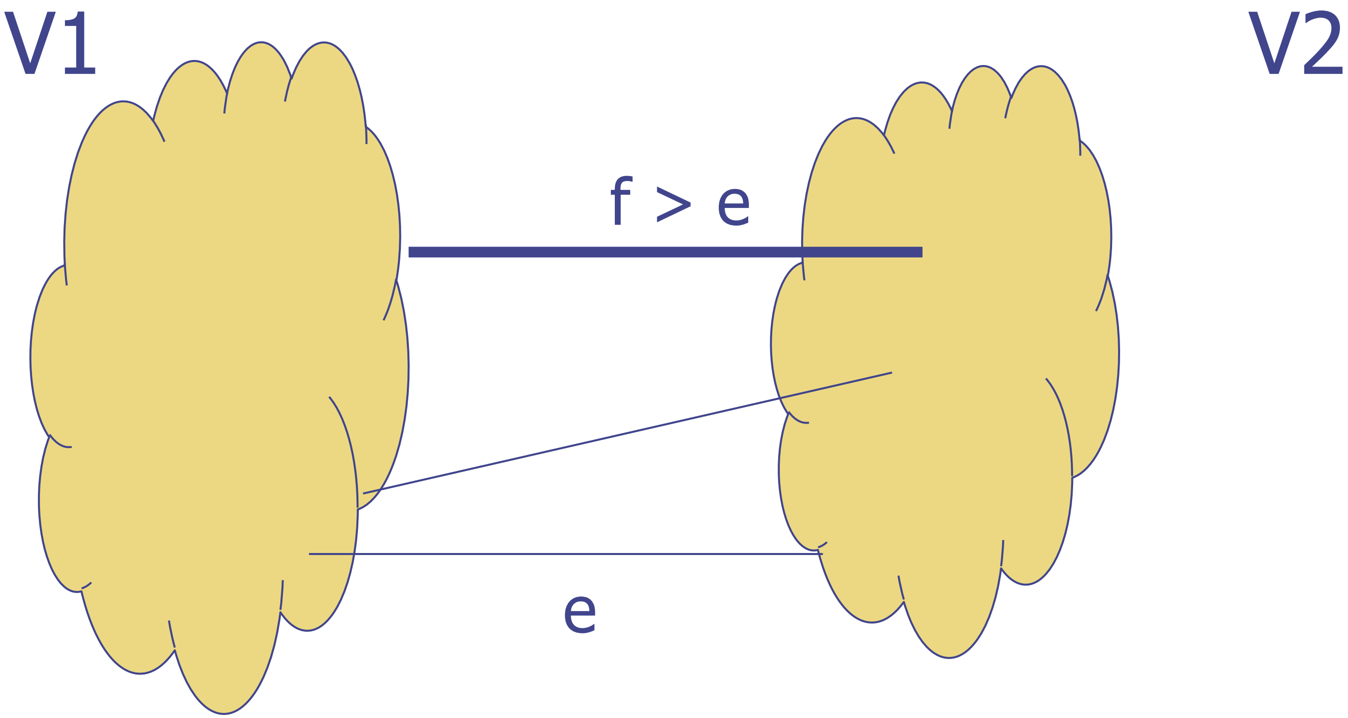
Why is this optimal!?



Justification of Prop. 13.25

- Argument by contradiction.
- Suppose that some minimum spanning tree T that is better than all trees containing e.
- Then can add edge e to T and remove some other edge between V1 and V2 and obtain a better MST

Justification of Prop. 13.25



Remove f and replace with e :

- Still gives a spanning tree.
- Gives a better spanning tree.

Prop 13.25 and Prims

- At each stage:
  - V1 = vertices within the current MST
  - V2 = “the rest” (vertices not in the MST)
  - The algorithm adds a minimum weight edge between V1 and V2, and so this edge must be part of some MST
  - Hence, the construction cannot make a “fatal mistake” – at no point can it add an edge not part of an MST

Greedy algorithm

- Prim’s algorithm for constructing a Minimal Spanning Tree is a ***greedy algorithm***:
  - it just adds a minimum weight edge
  - without worrying about the overall structure, without looking ahead.
  - It makes a locally optimal choice at each step.
- However, it still gives a globally optimal answer
  - This is fairly unusual
  - Usually greedy methods give answers that are “good but possibly sub-optimal” (e.g. change-giving)

Other MST Algorithms (not assessed)

- Kruskal
  https://en.wikipedia.org/wiki/Kruskal%27s_algorithm

- Reverse-delete
  https://en.wikipedia.org/wiki/Reverse-delete_algorithm

Minimal Expectations

- Clearly know, understand and be able to use
  - Definition of an MST
  - Algorithm to create one
  - Why it gives an optimal (minimum weight) spanning tree