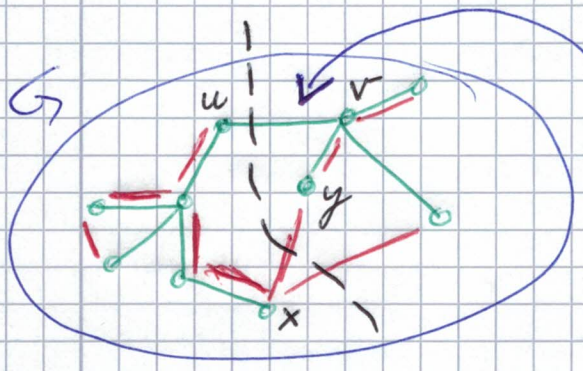
23.1-6

If for every cut there is a unique light edge crossing the cut, then MST is unique

What is an MST ?

Prove by contradiction



1st MST

2nd MST

(for contradiction)

Suppose 2 MSTs GREEN and RED

Trees are not the same ⇒ exist green edge not in the red tree

Maybe many! Fix ONE EDGE (u,v)

Remove it.

Defines cut of green tree in two parts

In green tree, only u,v crosses this cut

Note that we always get spanning tree by

a) removing (u,v)

b) adding any other edge across cut

(WHY ?)

CLAIM 1 (u,v) lightest edge

Otherwise remove (u,v) add ~~add~~ lightest edge across cut $\Rightarrow$ tree with smaller weight

CLAIM 2 Red tree has at least one edge crossing cut

Otherwise tree not connected

Consider path in red tree between u and v – crosses cut in some edge, say (x,y)

Replace (x,y) by (u,v) in red tree

still spanning tree

WHY?

- all vertices connected
- minimal # edges

This give a spanning tree cheaper than the red tree.

Contradiction.

23.1-9

23.1-9 I

T MST for $G = (V, E)$

$V' \subseteq V$

Let T' subgraph of T induced by V'

Let G' subgraph of G induced by V'

What does "SUBGRAPH OF G INDUCED BY V'" mean?

Assume T' connected

Prove that then T' is MST of G'

Proof by contradiction

Suppose not, and let T^* be MST of G'

$$w(T^*) < w(T')$$

Then we can use T^* to get cheaper spanning tree for G

Namely $(T \setminus T') \cup T^*$

WEIGHT

$$w((T \setminus T') \cup T^*) = w(T) - \underbrace{w(T') + w(T^*)}_{w(T') > w(T^*)}$$
$$< w(T)$$

NUMBER OF EDGES same in T' and T^* so no change

BUT DOES

$T^{new} = (T \setminus T') \cup T^*$

SPAN G? I.e., does it connect all vertices u & v

Prove by induction over length of path from u to v in original tree T

<u>Base case</u> $(u,v) \in E(T)$

(a) $(u,v) \in E(T) \setminus E(T')$

Then same edge in T^{new}

so u & v connected

(b) $(u,v) \in E(T')$

NOTE T' connected ⇒ G' connected

Then both u,v in V', connected in G'

T^* spanning tree for G'

So exist path in T^* between u and v

(maybe long path, but we don't care

<u>Induction step</u> Suppose u & v connected in T by path of length $t+1$

Look at last edge before v

$P: u \rightsquigarrow w \to v$

IH Vertices u & w connected by path of length $\leq t$ ⟹ connected also in T^{new}

23.1-9 III

$w \to w$ is edge

so w & v connected in T^{new} by same argument as in base case.

Hence u & v connected in T^{new}

It follows by the induction principle that T^{new} spans G

So T^{new} spanning tree of smaller weight $\Rightarrow$ contradiction

24.3-10

Given weighted, directed graph $G = (V, E)$
Edges leaving source s can have <u>negative</u> weight
All other edges non-negative weight
No negative weight-cycles What is this

<u>Argue that Dijkstra correct for this graph</u>

Why does Dijkstra fail on graphs with negative-weight edges?

<u>CLAIM 1</u> No shortest path $s \rightsquigarrow v$ need return to the source.

<u>Proof</u> By contradiction Suppose $P : s \underset{P_1}{\rightsquigarrow} s \underset{P_2}{\rightsquigarrow} v$

Note $P_2 : s \rightsquigarrow v$ also path, but apparently longer

$$w(P_2) > w(P) = w(P_1) + w(P_2)$$

$$0 > w(P_1)$$

But P_1 cycle. Contradiction — no negative weight cycles

HENCE: Any shortest path uses exactly one edge emanating from source w.l.o.g.

24.3-10

II

THOUGHT EXPERIMENT

Take graph G with negative-weight edges out of s

Add large number W to all edges (s, v) so that all weights positive $\boxed{G^*}$

How do distance estimates change?

If at time t before $v.d = \infty$ then still ∞

For $s.d$ still have $s.d = 0$

For $v \in V \setminus \{s\}$ if

$v.d < \infty$ then is original value $+ W$

Because extra weight W got added only for edge (s, u)

At all times $v.d^{new} = v.d^{prev} + W$

(or $v.d^{new} = v.d^{prev} = \infty$)

Exactly same relax operations.

Prove formally by induction over the algorithm execution.

Final values: Correct values for new G^* graph, because all weights non-negative

So... Shortest paths in

G^* computed correctly

Difference from shortest paths in G:
: subtract W for every path

What happens when Dijkstra run on G?
Same result as for G^*
except subtract W for
every path

Hence, Dijkstra for G computes
correct values