

Minimum Spanning Tree

①

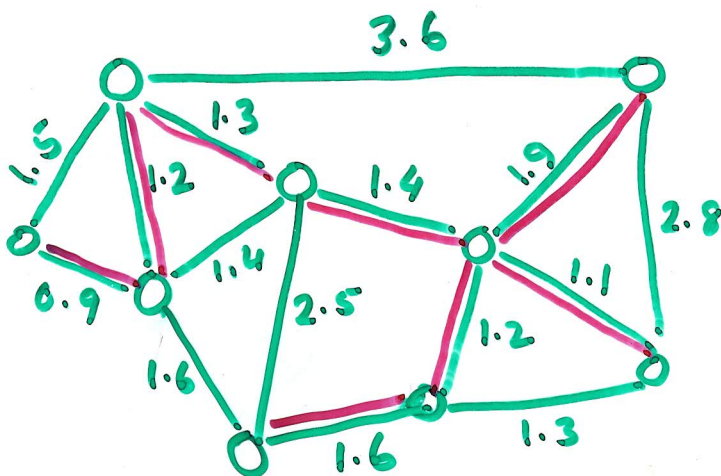
(V, E) is a connected, undirected, weighted graph. A spanning tree is a subgraph (V, T) , $T \subseteq E$ that is connected and has no cycle.

- Recall that a tree with n vertices has $n-1$ edges.

- A minimum spanning tree (MST) is a spanning tree (V, T) that minimizes

$$W(T) = \sum_{\{u,v\} \in T} w(\{u,v\}).$$

Ex.



MST is denoted with red-green edges

We will study two algorithms Prim's and Kruskal's algorithms for computing MST. Both of these algorithms can be viewed as a special case of a generic process which we study first. (2)

Growing an MST

Invariant $A \subseteq E$ is always a subset of some MST of (V, E) .

An edge $uv \in E$ is safe for A if $uv \notin A$ and $A \cup \{uv\}$ also satisfies the invariant.

Generic Method

$A := \emptyset$

while A is not a spanning tree of V yet do

find a safe edge uv ;

$A := A \cup \{uv\}$

endwhile

So far the method is trivial. The main part is how to choose safe edges.

A cut is a partition $V = W \cup (V - W)$;

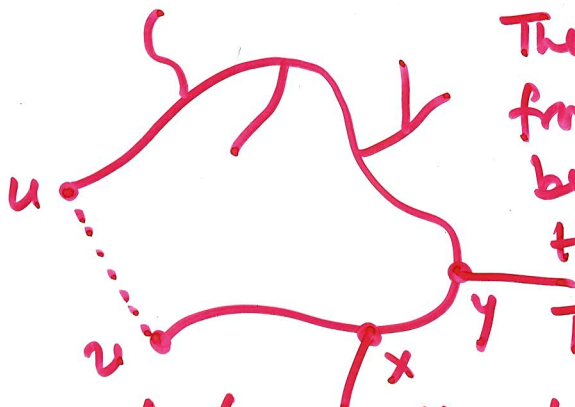
it respects $A \subseteq E$ if $A \subseteq \binom{W}{2} \cup \binom{V-W}{2}$,
i.e. all edges of A are either connecting two vertices in W or in $(V - W)$.

An edge uv crosses the cut if one vertex belongs to W and the other in $V - W$.

Claim. Let A be a subset of some MST of (V, E) . Let $(W, V - W)$ be a cut that respects A . Let uv be a crossing edge that minimizes $w(uv)$. Then uv is safe for A .

Proof. Consider an MST (V, T) with $A \subseteq T$.
If $uv \in T$ we are done.

So, assume $uv \notin T$ and $T' = T \cup \{uv\}$.



There is a unique path from u to v in T . Let xy be an edge on this path that crosses $(W, V - W)$.

Thus, $w(uv) \leq w(xy)$.

Now define $T'' = T' - \{xy\}$. T'' is again a spanning tree of V and $w(T'') \leq w(T)$. So, (V, T'') is an MST.

Prim's Algorithm

(4)

For each vertex i we assume a field p ($V[i].p$) that can be used to store a real number which is the priority of i .

We first add all vertices to a priority queue PQ , and the tree growing process starts. Here, the vertices which are extracted from PQ forms the cut with the rest of the vertices in PQ . Then we update the priorities of vertices with respect to new cross edges each time we include a vertex from PQ to our current set.

Initialization

```
PQ := ∅;  
for i := 1 to n do  
  if  $i \neq k$  then  $V[i].p := \infty$   
  else  $V[i].p := 0$ ;  
        $V[i].\pi := \text{nil}$ ;  
endif  
  add  $i$  to  $PQ$  with  
  priority  $V[i].p$   
endfor
```

