CS 340 lecture 27 Nov 16

Assignment 7 due Nov 23

one more assignment after that

late assignments usually not accepted

4.5 Minimum Spanning Trees

Definition 18 Let $G = (V, E)$ and $T = (V_T, E_T)$ be undirected graphs.

T is a spanning tree of G if:

- $V_T = V$
- $E_T \subseteq E$
- T is a tree (i.e., T is connected and acyclic)

Let $c : E \to \mathbb{R}^{\geq 0}$ be a weight function for G.

T is a minimum spanning tree (MST) for G w.r.t. c if

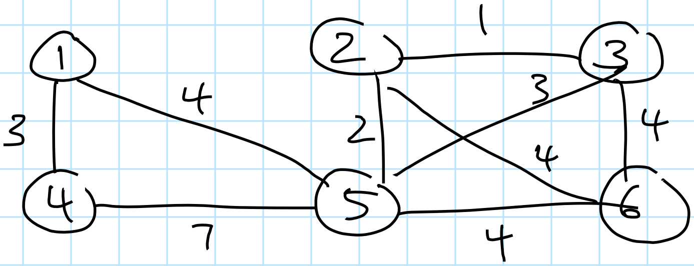
(1) T is a spanning tree for G AND

(2) there is no spanning tree $T' = (V_T', E_T')$ for G with

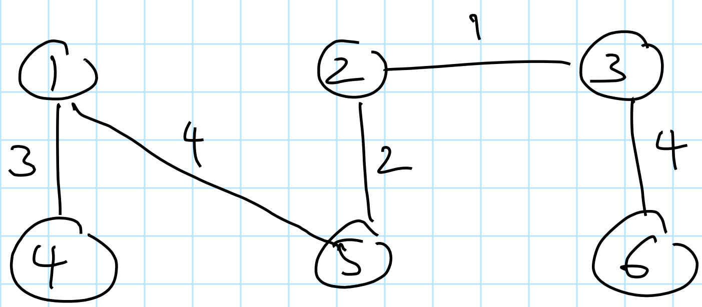
$$\underbrace{\sum_{(v,w) \in E_T'} c(v,w)}_{\text{cost of edges in } T'} < \underbrace{\sum_{(v,w) \in E_T} c(v,w)}_{\text{cost of edges in } T}$$

Example 48

G

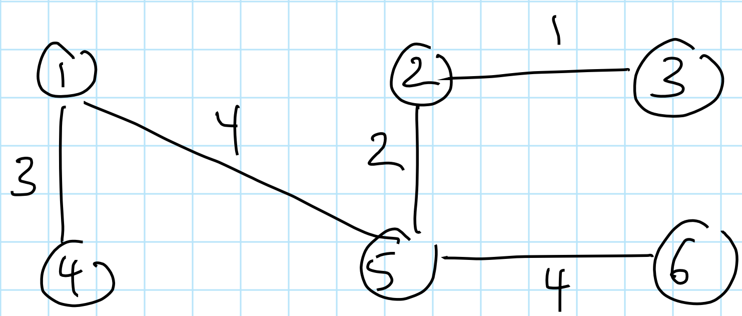


an MST.



need two of the edges
(1, 4)
(1, 5)
(4, 5)

another MST:



MST Problem

given: undirected graph $G = (V, E)$
and weight function $c : E \to \mathbb{R}^{\geq 0}$

task: find an MST for G w.r.t. c

(e.g., connect houses in a neighbourhood with min. amount of cable)

NOTE

- MST exists $\iff$ G connected

  $\leadsto$ we will assume for simplicity that G is connected

4.5.1 Prim's Algorithm

Robert Prim 1957

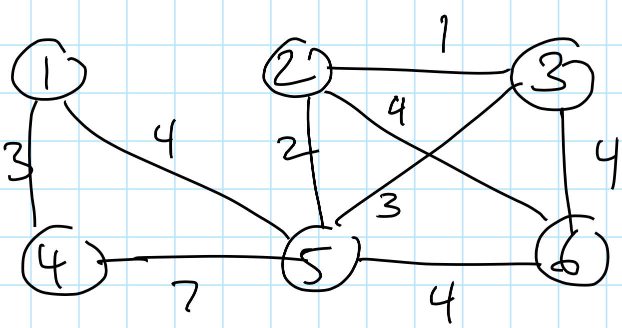
but first: Vojtech Jarník 1930

<u>idea</u>
- start at any vertex
- in each stage, add one edge that connects <u>one</u> new vertex

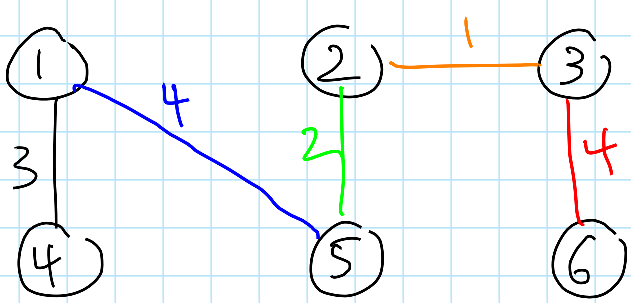
  choose this edge greedily (i.e., with lowest weight) from all edges incident to one of the previously connected vertices

Example 48 ctd.

G:



start at vertex ① (for example)



greedily choose

(i) add (1, 4)
(ii) add (1, 5)
(iii) add (5, 2)
(iv) add (2, 3)
(v) add (3, 6)

OR: add (5,6)

or add (2,6)

implementation: exactly like Dijkstra!

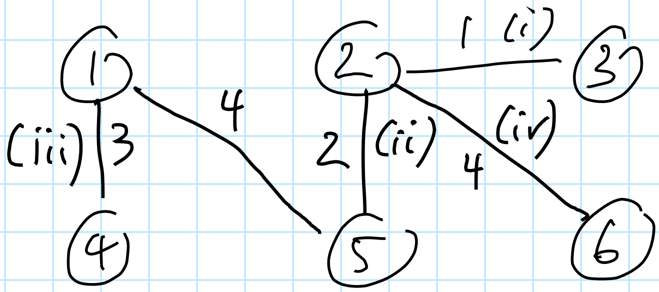
$O(|V|^2)$

with same improvements possible for sparse graphs.

4.5.2 Kruskal's Algorithm

Joseph Kruskal, 1956

idea • keep greedily adding edges of lowest weight, if they don't create a cycle



(i) add (2,3)
(ii) add (2,5)
(iii) add (1,4)
(iv) add (2,6)
(v) add (1,5)

at intermediate steps, the constructed structure is a *forest*.

implementation: union-find structure (disjoint set class)

see Chapter 8 in textbook

$O(|E| \log |E|)$ running time for Kruskal