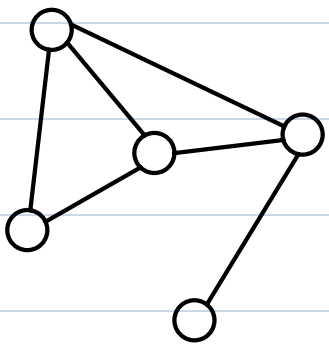
Friday, Feb. 3, 2023
Lecture # 8
MSSC 6000

Announcements

* HW 1 due next Mon, 11:59pm → turn in on D2L

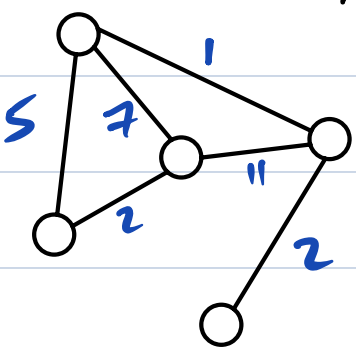
Problem #2 Minimum Spanning Tree problem

Def: A graph is a set of vertices or nodes, connected in pairs by edges.



← a graph with 5 vertices and 6 edges

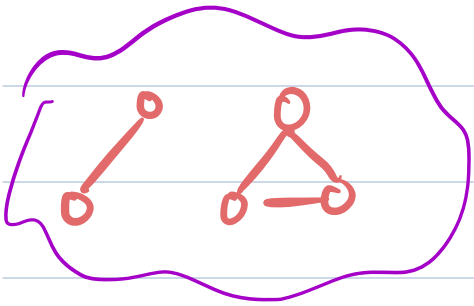
A weighted graph is a graph whose edges have real #s as "weights"

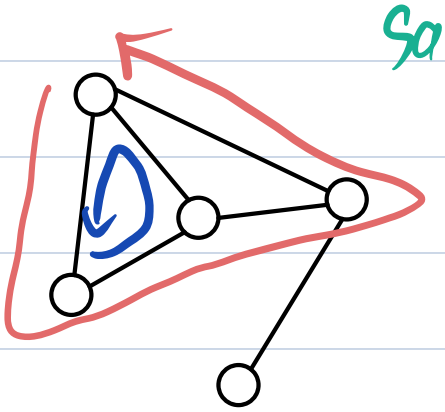


← weighted graph

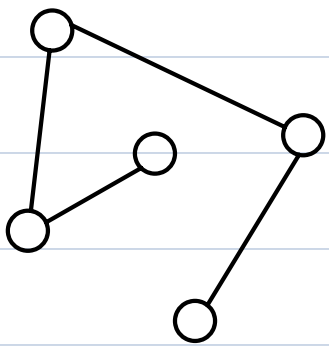
A tree is a graph that is
* connected → can reach every vertex from every other vertex eventually
* has no cycles → a cycle is a sequence of distinct edges that starts and ends at the same vertex

Not connected:





Ex of a tree:

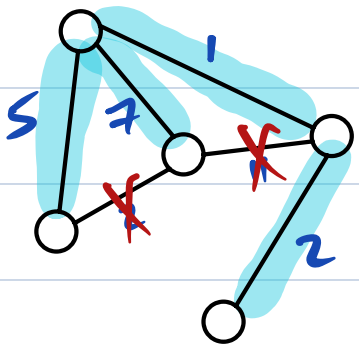


connected ✓
no cycles ✓

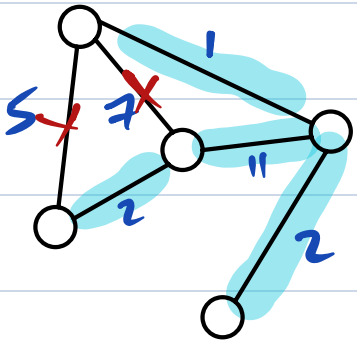
Minimum Spanning Tree Problem:

Given a weighted graph G, find the subset of edges that forms a tree with minimum weight.

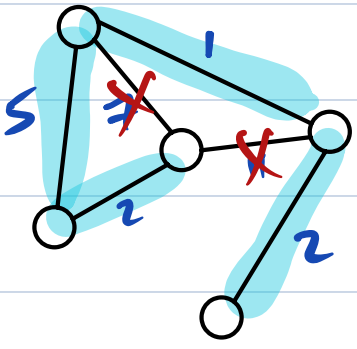
Goal: Delete edges until what remains is a tree. There are many ways to do this — which way leaves behind the smallest total weight of the edges.



Weight = 5+7+1+2
= 15



2+1+1+2
= 16



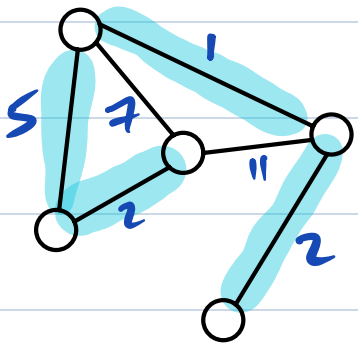
5+2+1+2 = 10

minimal spanning tree

Ex: Suppose you need to connect a bunch of buildings with fiber optic cable to make a network.
Between any two buildings, there is some cost to connect those two.
Make a graph: vertices = buildings

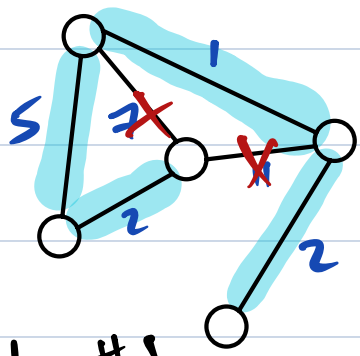
edges connect building that could be connected

Weights of edges = cost to connect them

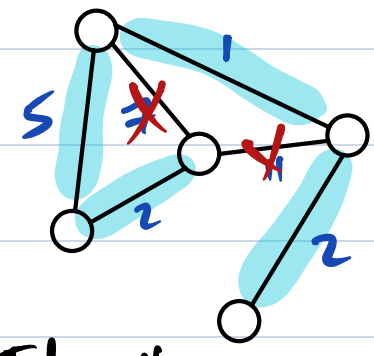


Possible greedy algorithms:

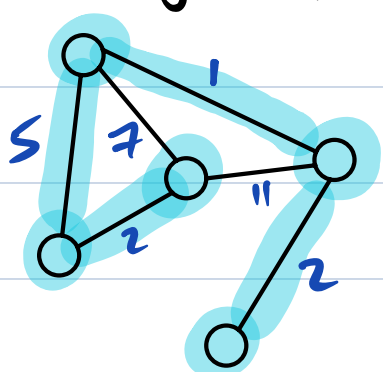
* start with no edges and then always the cheapest edge that doesn't make a cycle
* start with all edges, and always delete the most expensive one as long as it doesn't disconnect the graph
* pick a particular vertex as the "start" and repeatedly add the cheapest edge that touches a vertex you've reached so far (no cycles)



Idea #1



Idea #2



Idea #3

These three algos don't always give the same solution.

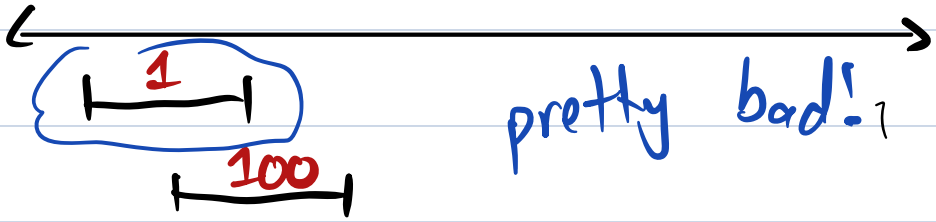
Are any of these guaranteed to give optimal solutions?

Theorem: They all do! (We won't prove this.)

Problem #3: Weighted Interval Scheduling

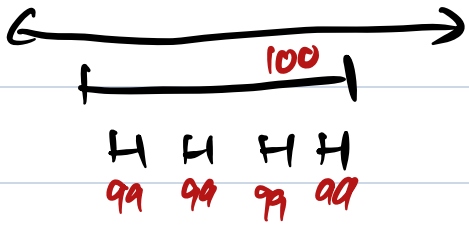
This is like regular interval scheduling, except each request r_i comes with a value v_i and your goal is to maximize the total value of requests satisfied.

How does our previous greedy algo do?



Possible Greedy Algos:

* best = highest value



* best = shortest meeting

* best = highest value density
  ↳ $\frac{\text{value}}{\text{duration}}$