Lecture #18

10/12/2016

- Last time we started a new topic "Mathematics of Networks".

Reminder:

- * network is a connected graph
- * weighted network is a network, where each edge has a prescribed weight (cost)
- * degree of separation (of two vertices) is the length of the shortest path between these 2 vertices
- * tree – a network without circuits
- * Key properties of trees:
  - (1) There is only one path b/w any 2 vertices
  - (2) All edges are bridges!
    ! Discuss this in class!
  - (3) In a tree with N vertices, there are exactly N-1 edges.

Def: In a network with N vertices and M edges, the redundancy R is defined by $R := M - (N-1)$.

Rmks: (1) $R \geq 0$

(2) $R = 0$ if and only if, the network is a tree

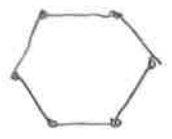
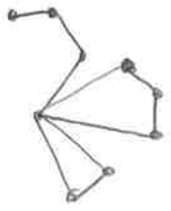
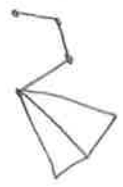
(3) The general rule is that the bigger the redundancy the bigger number of circuits in the network.

- * subtree (of a network) – subgraph of our ambient graph which in turn is a tree.
- * spanning tree – a subtree which includes all the nodes (= vertices) of the network.

Lecture #18 10/12/2016

- Last time we concluded the lecture by counting spanning trees.

Example 1: Compute the number of spanning trees in the following networks:

(a)  (b)  (c) 

▸ See page 5 of notes to Lecture #17.

! Note that you know the total number of edges we need to discard in each of these cases: it is the redundancy of the network ▪

Q: Why do we care about spanning trees?

This was already discussed last time, but let me remind again. In the real life, we often need to find a "subnetwork" of a network, which includes all vertices and has the minimal (or maximal) total costs. See Example 7.9 in the Textbook.

If we assume that all the weights are positive (or all negative), then it is clear that such an "optimal" subnetwork must be a spanning tree!

Def: In a weighted network, a minimum spanning tree (MST) is a spanning tree with least total weight/cost.

While in most applications, the weights represent costs and therefore we want to minimize to total weight, sometimes those weights represent profits, meaning we would like to maximize the total weight.

Def: In a weighted network, a maximum spanning tree (MaxST) is a spanning tree with highest total weight/cost.

Rmk: In general, there can be several MSTs and MaxSTs

Lecture #18

10/12/2016

- Finally, we will address the key question:

Q: How to find MST?

One of the efficient algorithms is called the Kruskal's Algorithm, named after American mathematician Joseph Kruskal.

~~Idea~~: Apply the same reasoning as in the cheapest-link algorithm.

Kruskal's Algorithm

Step 1: Pick the cheapest edge available, Mark it (e.g. red color) (if there are several cheapest - choose any)

Step 2: Pick the next cheapest edge available and mark it.

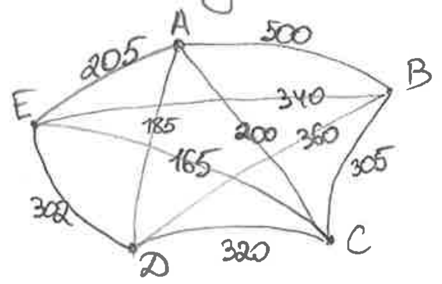
Step 3, .., N-1: At each step, pick the cheapest unmarked edge available that does not create a circuit.

! After these N-1 steps, you are done!

Let us illustrate how it works on the following example.

Example 2 [Exercise 7.3.33]

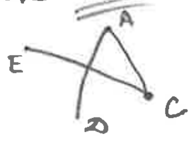
Find the MST of the network below using Kruskal's algorithm. Find its weight



- Step 1: Pick EC

Step 2: Pick AD

Step 3: Pick AC



Step 4: The next cheapest is EA, but it makes a circuit, skip it. Next is ED, but it also makes a circuit, skip! Next is BC. Done! Draw picture!

Total weight: 165 + 185 + 200 + 305 = 855 ■

Lecture #18 10/12/2016

Let us point out that any algorithm finding MST can be also applied to find MaxST. For this, change the signs in all weights and apply your algorithm. Since switching the signs makes MST into MaxST and other way around, we are all set.

Upshot: Given a weighted network, reverse signs of all the weights, apply Kruskal's algorithm.

Example 3: Find MaxST in Example 2.

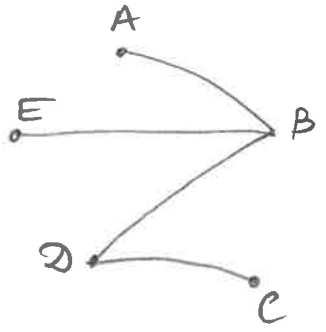
Step 1: Pick AB

Step 2: ~~P~~ Pick BD

Step 3: Pick BE

Step 4: Next is CD → pick it.

We are done. We chose:



Total weight:
$500 + 340 + 360 + 320 = 1520$

- This completes our discussion of Section 7.

Ask if there are any q-s.

We will skip Section 8 and start Section 9.