Summary on Lecture 23, November 22, 2017

Optimal spanning trees

2. Optimal spanning trees. Let $G = (V(G), E(G))$ be a finite graph. As in the case of directed graphs, we say that G is a *weighted graph* if we are given a *weight function* $\mathsf{wt} : E(G) \to [0, \infty)$. The if $H \subset G$ is a subgraph of G, then a *weight* $W(H)$ is the sum of the weights of edges in H.

Optimal spanning tree problem: For a given finite connected graph $G = (V(G), E(G))$, find a spanning tree $T \subset G$ of minimal weight. Such a spanning tree is called *optimal* (or *minimal* in some other sources).

Our next algorithm builds an optimal spanning tree for a weighted graph $G = (V(G), E(G))$, $|E(G)| = m$, whose edges $e_1, \dots, e_m$ have been initially sorted so that

$$\mathsf{wt}(e_1) \le \mathsf{wt}(e_2) \le \cdots \le \mathsf{wt}(e_m).$$

The algorithm proceeds one by one through the list of edges of G, beginning with the smallest weights, choosing edges that do not introduce cycles. When the algorithm stops, the set E is supposed to be the set of edges in a minimum spanning tree for G. The notation $E \cup \{e_j\}$ in the statement of the algorithm stands for the subgraph whose edge set is $E \cup \{e_j\}$ and whose vertex set is $V(G)$.

Kruskal's Algorithm $(G = (V(G), E(G)),\ \mathsf{wt} : E(G) \to (0, \infty))$
`Input:` A finite weighted connected graph (G, wt) with edges listed in order of increasing weight
`Output:` A set E of edges of an optimal spanning tree for G)
`Set` $E = \emptyset$`, for` $j = 1$ `to` $|E(G)|$ `do`
  `if` $E \cup \{e_j\}$ `is acyclic then`
  `Put` e_j `in` E.
`return` E

Exercise. Use the **Kruskal's Algorithm** algorithm for the following graph:

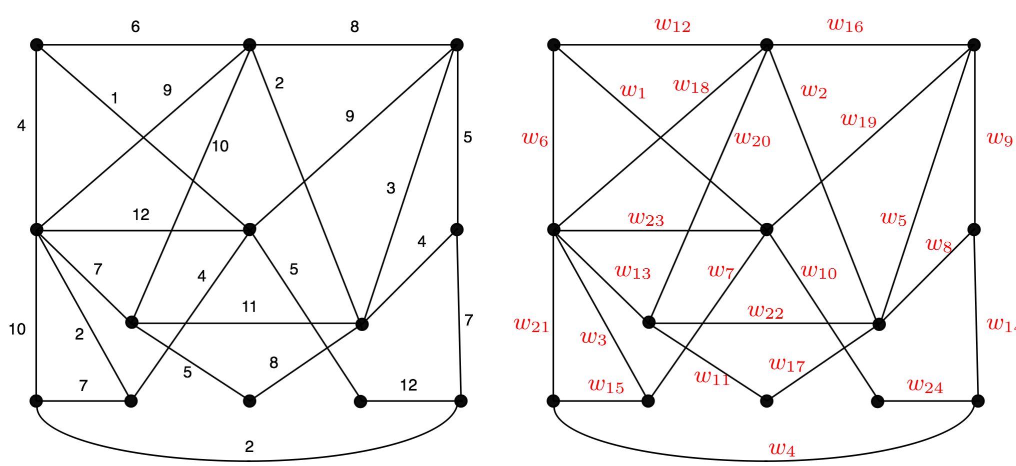

Fig. 3. Here the weights $w_i = \mathsf{wt}(e_i)$ of the edges are already ordered.

Prim's Algorithm$(G = (V(G), E(G)),\ \mathtt{wt} : E(G) \to (0, \infty))$

```
Input: A finite weighted connected graph (G, wt) with edges listed in any order
Output: A set E of edges of an optimal spanning tree for G)
Set E = ∅.  Choose w in V(G) and set V := {w}.
while |V| < |V(G)| do
   Choose an edge {u, v} in E(G) of smallest possible weight
      with u ∈ V and v ∈ V(G) \ V.
   Put {u, v} in E and put v in V.
return E
```

Exercise. Use the **Prim's Algorithm** algorithm for the graph given at Fig. 3.