Spanning Trees

Master MoSIG — Algorithms and Program Design

Florent Bouchez Tichadou & Nicolas Fournel

November 18, 2014

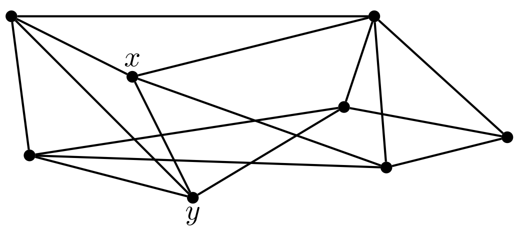
Objectives:

- Remind basic graph knowledge.
- Know examples of optimal greedy algorithms and how to prove it.

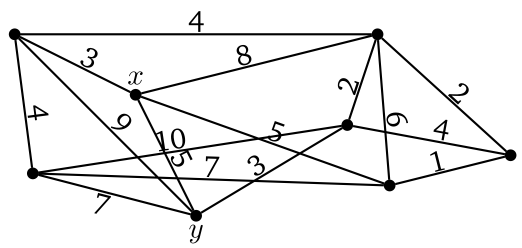
1 Graphs

Graphs are structures comprising of a set of nodes or vertices V and a set of edges connecting nodes, $E = (V \times V)$. To define a graph it is sufficient to give the pair $G = (V, E)$. Usually, $n = |V|$ is the number of nodes and $m = |E|$ is the number of edges.

Often, attributes can be added to nodes or edges, for instance numbers representing a weight, a distance, etc. To represent that, we use functions such at $w : E \to \mathbb{R}$, where for $e = (x, y) \in E$, $w(e)$ is the weight of the edge between x and y.



Regular graph



Weighted graph on edges

2 Trees

Trees are actually just particular graphs. There are many equivalent definitions of trees, for instance, a graph $G = (V, E)$, with $n = |V|$, is a tree if and only if:

1. G has no cycle and has exactly $n - 1$ edges;
2. G is connected and there is exactly $n - 1$ edges;
3. G is connected but is not if any single edge is removed;
4. any two nodes are connected by a *unique* path.

Exercise 1 (Tree definition)

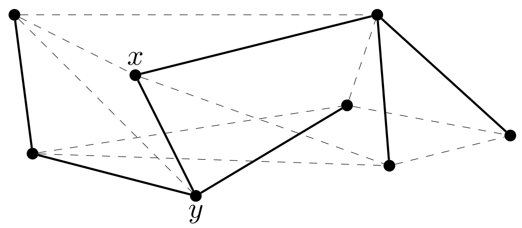
Prove the above points are equivalent.

3 Spanning tree

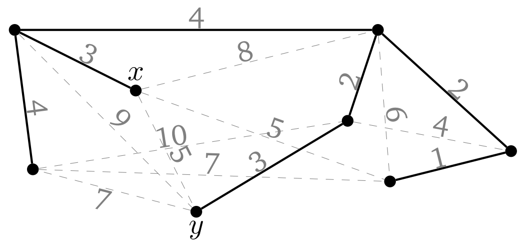
A spanning tree over a graph $G = (V, E)$ is a subset of edges $S \subset E$, that forms a tree, i.e., the graph (V, S) is a tree. If the graph is not weighted, every spanning tree is equivalent as $|S| = n - 1$.

If the graph is weighted, the weight of the spanning tree is the sum of all weights of chosen edges. Usually, we want a minimum spanning tree, i.e., to minimize:

$$\sum_{e \in S} w(e)$$



Spanning tree



Minimum spanning tree

3.1 Algorithms

Finding the minimum spanning tree of a graph is a simple problem. It is one of those few cases where a greedy algorithm is actually optimal! There are two main algorithms whose names are their authors': Prim and Kruskal.

Prim We start from a set containing only one node; at each step, we add the node that is the closest to the set (in terms of weight).

Kruskal We start from an empty set of edges; at each step, we add the edge of smallest weight that does not create a cycle.

In order to be efficient, Prim will require the use of a priority queue, where new elements can be added at a low cost; a binary heap can be used for this task. Kruskal will require checking if nodes are in the same connected component; data structures such as binomial heaps are efficient for this purpose.

Complexity In both cases, choosing the right data structure can bring the complexity down to $O((|V| + |E|) \log |V|)$.

Exercise 2 (Minimum spanning trees)

We have just given hints on the algorithms. Chose one algorithm then do the following:

Question 2.1 Write in pseudo code the main algorithm without detailing data structures.

Question 2.2 Write the pseudo code for the data structures.

Question 2.3 Prove the above complexity is correct for the algorithm you chose.

Question 2.4 Prove the optimality of the algorithm (i.e., prove it always finds an optimal solution: a spanning tree of minimum weight).