Graph Algorithms

Minimum Spanning Trees & Shortest Paths

Francesco Andreussi

Bauhaus-Universität Weimar

24 May 2019

Bauhaus-Universität Weimar
Fakultät Medien

Minimum Spanning Tree

What is it?

Minimum Spanning Tree

What is it?
It is the graph considering **only** the set of edges of **minimal weight** that connects together **all** the nodes of a graph. As a consequence, the graph is acyclic, and it can be called a tree.

Kruskal Algorithm

Pseudocode

```
MST-KRUSKAL(G):
  A = emptySet
  foreach v in G.V:
    MAKE-SET(v)
  SORT-WEIGHTS(G.E)
  foreach (u,v) in G.E:
    if SET(u) != SET(v):
      A.add({(u,v)})
      UNION(u,v)
  return A
```

G is a weighted graph.

sort the edges $G.E$ in a NON-decrescent way w.r.t. their weights

if u and v are in different sets, put the edge in A and unite their sets

$A =$ set of edges defining the MST

Kruskal Algorithm

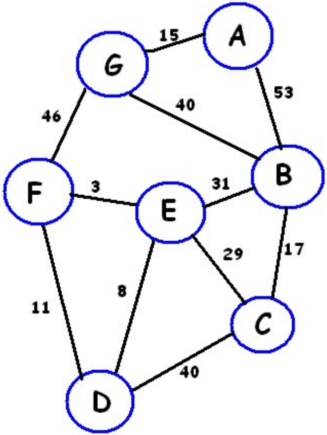

Fig. 1: Find by hand the MST using Kruskal

Prim Algorithm

Pseudocode

```
MST-PRIM(G,r):
  foreach u in G.V:
    u.key = INFINITY
    u.parent = NIL
  r.key = 0
  Q = G.V
  while Q != emprySet:
    u = EXTRACT-MIN(Q)
    foreach v in u.Neighbours:
      if v in Q
            && w(u,v)<v.key:
        v.parent = u
        v.key = w(u,v)
```

G is a weighted graph and r is the first node

Q is a min-heap

if v is not fully analysed and a better way of reaching it, update its values

Prim Algorithm

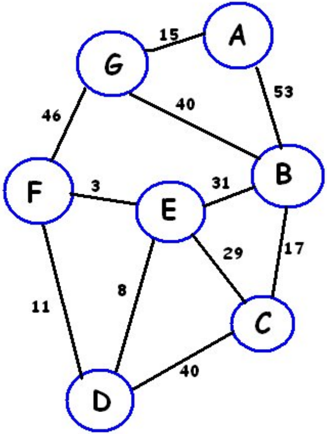

Fig. 2: Find by hand the MST using Prim

Single-Root Shortest Paths

What is it?

Single-Root Shortest Paths

What is it?
The problem of finding the Single-Root (or Single-Source) Shortest Paths has, as its goal, the computation of the shortest (lighter) path from a given node v in a graph $G(V,E)$ to all the others.

Single-Root Shortest Paths

What is it?
The problem of finding the Single-Root (or Single-Source) Shortest Paths has, as its goal, the computation of the shortest (lighter) path from a given node v in a graph $G(V,E)$ to all the others.
An algorithm that answers to this question can also solve the following problems: **Single-Destination** Shortest Paths, Shortest Path **Between Two Nodes** and Shortest Paths **Every Pair of Nodes** and it has no theoretical limitation w.r.t. the type of graph (actually, there can be algorithm-specific restrictions).

Relaxing an Edge

An **edge** is **relaxed** when it is possible to reach an **already discovered** node from the same source following a lighter path.
In that case the distance from the source is updated with the new value, as well as the "reference to the parent/preceder" of the considered node.

```
RELAX(u,v,w):
  if v.dist > w(u,v) + u.dist:
    v.parent = u
    v.dist = w(u,v) + u.dist
```

Bellman-Ford Algorithm

Pseudocode

This algorithm is designed to deal with **directed weighted** graphs; it fails when a **negative** weighted **cycle** is detected.

```
BELLMAN-FORD(G,s):
  foreach u in G.V:
    u.dist = INFINITY
    u.parent = NIL
  s.dist = 0
  for i=1 to |G.V|-1:
    foreach (u,v) in G.E:
      RELAX(u,v,w(u,v))
  foreach (v,v) in G.E:
    if v.dist > w(u,v)+u.dist:
        return FALSE
  return TRUE
```

G is a weighted graph and s is the chosen source

check |$G.V$|−1 times if every edge can be *relaxed*

after this cycle either an edge can be relaxed (there is a negative cycle), or the nodes have the correct path information for solving the problem

Bellman-Ford Algorithm

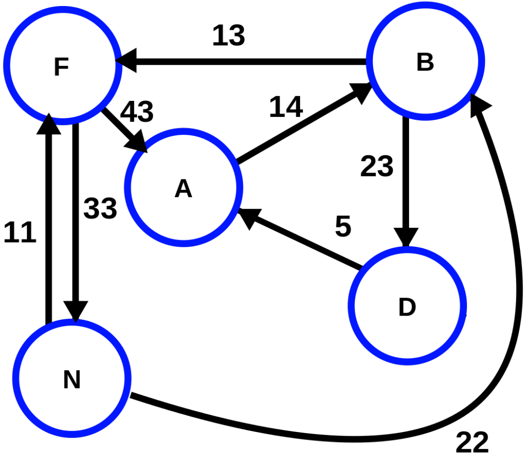

Fig. 3: Find by hand the Shortest-Paths from A using Bellman-Ford

Bellman-Ford Algorithm

Solution

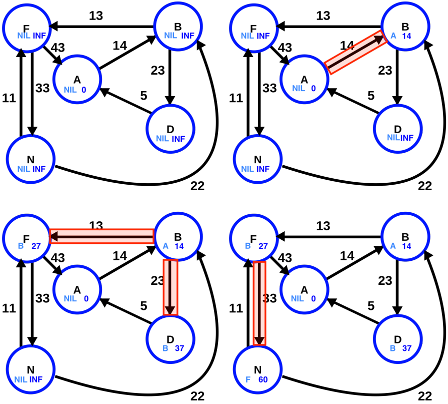

Fig. 4: *dist* and *parent* values for each of the $|G.V|-1$ execution of main cycle, in red the **relaxed** edges for each iteration

Thanks for the Attention!