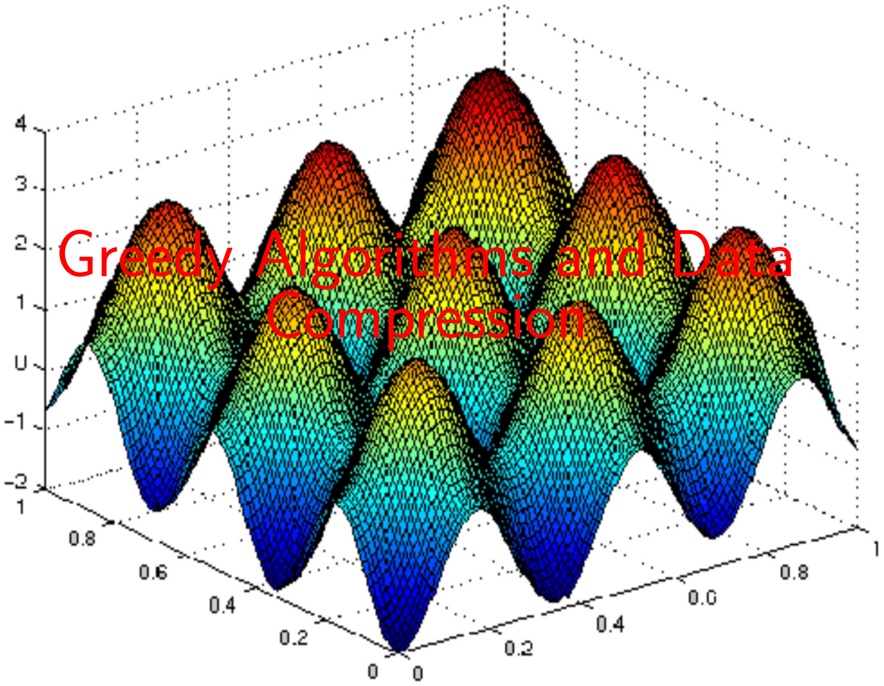
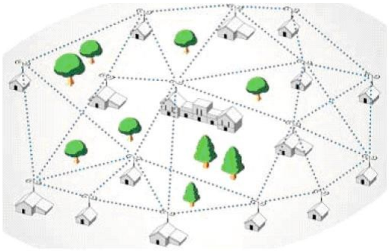
Greedy Algorithms and Data Compression



A network construction problem: Minimum Spanning Tree

CLRS 23, KT 4.5, DPV 5.1

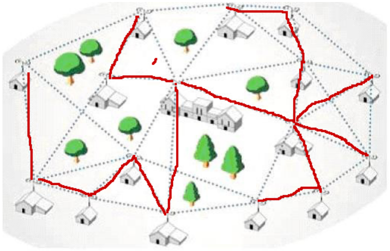
- We have a set of locations.
- For some pairs of locations it is possible to build a link connecting the two locations, but it has a cost.
- We want to build a network (if possible), connecting all the locations, with minimum cost.
- As we want to minimize the cost the resulting network must be a tree.



Network construction: Minimum Spanning Tree

- We have a set of locations.
- For some pairs of locations it is possible to build a link connecting the two locations, but it has a cost of $w(v_i, v_j)$.
- We want to build a network, connecting all the locations, with minimum cost.
- As we want to minimize the cost the resulting network must be a tree.

Construct the MST



Properties of trees

- A tree on n nodes has $n-1$ edges.
- Any connected undirected graph with n vertices and $n-1$ edges is a tree.
- An undirected graph is a tree iff there is a unique path between any pair of nodes.

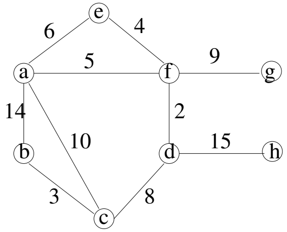
Let $G = (V, E)$ be a (undirected) graph.

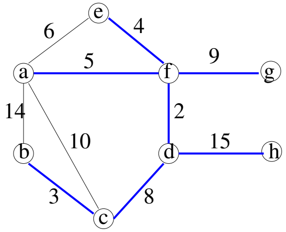
- $G' = (V', E')$ is a subgraph of G if $V' \subseteq V$ and $E' \subseteq E$.
- A subgraph $G' = (V', E')$ of G is spanning if $V' = V$.
- A spanning tree of G is a spanning subgraph that is a tree.

Any connected graph has a spanning tree

Minimum Spanning Tree problem (MST)

Given as input an edge weighted graph $G = (V, E, w)$, where $w : E \to \mathbb{R}$. Find a tree $T = (V, E')$ with $E' \subseteq E$, such that it minimizes $w(T) = \sum_{e \in E(T)} w(e)$.





Some definitions

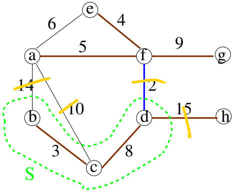
For a graph $G = (V, E)$:

A path is a sequence of consecutive edges.

A cyle is a path with no repeated vertices other that the one that it starts and ends.

A cut is a partition of V into two sets S and $V - S$.

The cut-set of a cut is the set of edges with one end in S and the other in $V - S$. $cut(S, V - S) = \{e = (u, v) \in E \mid u \in S \; v \notin S\}$



MST: Properties

Given a weighted graph $G = (V, E, w)$, assume that all edge weights are different.
A MST T in G has the following properties:

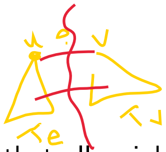
- Cut property
  $e \in T \Leftrightarrow e$ is the lightest edge across some cut in G.
- Cycle property
  $e \notin T \Leftrightarrow e$ is the heaviest edge on some cycle in G.

The MST algorithms are methods for ruling edges in or out of T.

The $\Leftarrow$ implication of the cut property yields the blue (include) rule, which allow us to include safely in T a min weight edge from some identified cut.

The $\Rightarrow$ implication of the cycle property will yield the red (exclude) rule which allow us to exclude from T a max weight edge from some identified cycles.

The cut property



Let $G = (V, E, w)$, $w : E \to \mathbb{R}^+$, such that all weights are different. Let T be a MST of G.

Removing an edge $e = (u, v)$ from T yields two disjoint trees T_u and T_v, so that $V(T_u) = V - V(T_v)$, $u \in T_u$ and $v \in T_v$. Let us call $S_u = V(T_u)$ and $S_v = V(T_v)$.

Claim

$e \in E(T)$ is the min-weight edge among those in cut(S_u, S_v).

Proof.

Otherwise we can replace e by an edge with smaller weight and create a new spanning tree with smaller total weight.

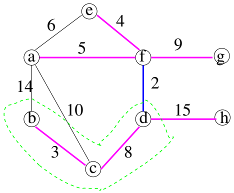
The cut property

Claim

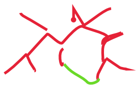
For $S \subseteq V$, let $e = (u, v)$ be the min-weight edge in $cut(S, V - S)$, then $e \in T$.

Proof.

Assume $e \notin T$ and that $u \in S$ and $v \notin S$. As T is a spanning tree there must be a path from u to v in T. As $u \in S$ and $v \notin S$, there is an edge $e' \in cut(S, V - S)$ in this path. Replacing e' with e produces another spanning tree. But then, as $w(e) > w(e')$, T was not optimal. $\square$



The cycle property



For an edge $e \notin T$, adding it to T creates a graph $T + e$ having a unique cycle involving e. Lets call this cycle C_e.

Otherwise we can replace e by an edge with smaller weight and create a new spanning tree with smaller total weight.

Claim

For $e \notin E(T)$, e is the max-weight edge in C_e.

Proof. Otherwise, removing any edge different from e in $T + e$ produces a spanning tree with smaller total weight.

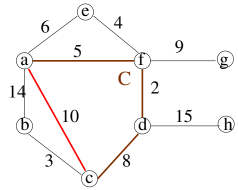
The cycle property

Claim (The cycle rule)

For a cycle C in G, the edge $e \in C$ with max-weight can not be part of T.

Proof.

Observe that, as G is connected, $G' = (V, E - \{e\})$ is connected. Furthermore, a MST for G' is a MST for G. $\square$



C=cycle spanning {a,c,d,f}

Generic greedy for MST: Apply blue and/or red rules

The Min. Spanning Tree problem has the optimal substructure and we can solve it with a greedy algorithm.
Robert Tarjan: Data Structures and Network Algorithms, SIAM , 1984

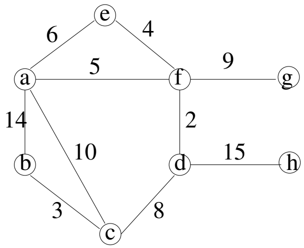
The algorithms color edges of G with blue (in the MST) or red (not in MST) using the rules:

Blue rule: Given a cut-set between S and $V - S$ with no blue edges, select from the cut-set a non-colored edge with min weight and paint it blue

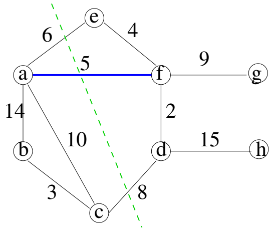
Red rule: Given a cycle C with no red edges, selected an non-colored edge in C with max weight and paint it red.

Greedy scheme:
Given G, $|V(G)| = n$, apply the red and blue rules until having $n - 1$ blue edges, those form the MST.

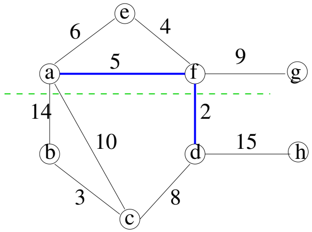
Application of red/blue rules



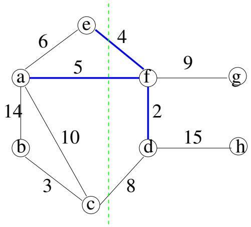
Application of red/blue rules



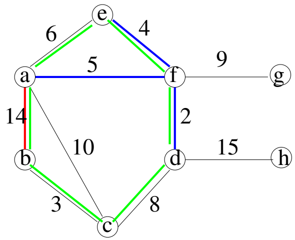
Application of red/blue rules



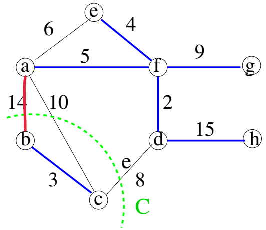
Application of red/blue rules



Application of red/blue rules



Application of red/blue rules



Greedy for MST : Correctness

Theorem

The algorithm finishes and at the end of the execution the blue edges form a MST

Sketch of proof As in each iteration an uncolored edge is colored, the algorithm finishes after at most m applications of the rules.

For the secod part the proof is an induction on number of iterations, for blue and red rules.
The base case (no edges colored) is trivial.
The induction step is the same the one in the proofs of the cut and cycle rules. $\square$

We need implementations for the algorithm!

A short history of MST implementation

There has been extensive work to obtain the most efficient algorithm to find a MST in a given graph:

- O. Borůvka gave the first greedy algorithm for the MST in 1926. V. Jarnik gave a different greedy for MST in 1930, which was re-discovered by R. Prim in 1957. In 1956 J. Kruskal gave a different greedy algorithms for the MST. All those algorithms run in $O(m \lg n)$.
- Fredman and Tarjan (1984) gave a $O(m \log^* n)$ algorithm, introducing a new data structure for priority queues, the Fibbonacci heap. Recall $\log^* n$ is the number of times we have to apply iteratively the log operator to n to get a value ≤ 1, for ex. $\log^* 1000 = 2$.
- Gabow, Galil, Spencer and Tarjan (1986) improved Fredman-Tarjan to $O(m \log(\log^* n))$.
- Karger, Klein and Tarjan (1995) $O(m)$ randomized algorithm.
- In 1997 B. Chazelle gave an $O(m\alpha(n))$ algorithm, where $\alpha(n)$ is a very slowly growing function, the inverse of the Ackermann function.