Minimum Spanning Trees

Thursday, April 16, 2020 3:25 PM

Sum of all weights

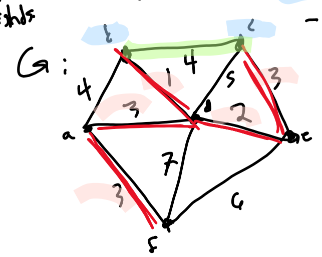
has all the nodes

undirected graphs
- no loops/cycles.
- at most one incoming
- Connected.

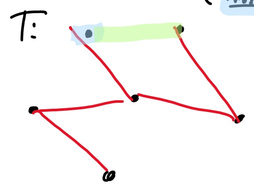
at least one path for u → v

at most one path

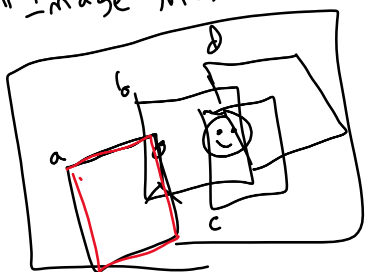
G:

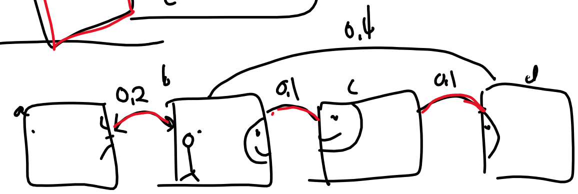


T:



App. Image Mosaics.





Algorithms for MST:

3 algs, all variations of same idea

Start: T as an empty graph (no edges) E: edges in G

Loop: While E nonempty:

pop some edge e from E

if condition(e):

add e to T

return T.

- Kruskal's
  - process edges e ∈ E in order by cost.
  - Add if new edge does not create a cycle in T
  - Start with T as all vertices, no edges (disjoint forest)

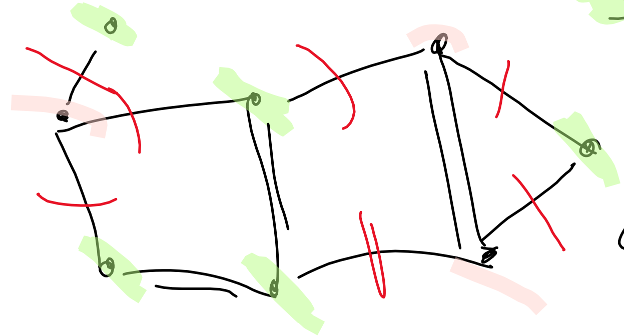
- Prim's
  - Add the smallest edge adjacent to some node in T.
  - T starts as empty graph.

- Borůvka's
  - For each component S in T
    add the smallest edge adjacent to some node in S

— T starts as disjoint forest.

Cuts: a partition in two of V, ← partition 1

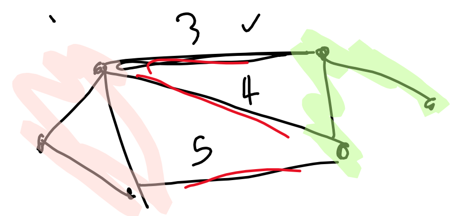
$S \subsetneq V$, and $V \setminus S$ ← everything else



Cut edges: (u, v)

Safe edges:

e is safe iff it's the ~~unique~~ smallest edge of some cut



Cut property: If e is safe then it is in every MST for G.

Proof: Assume e is safe.

Suppose T is an MST.

and for contradiction, $e \notin T$.

e safe means. $\exists$ some S, V/S cut, s.t. e is min cut edge

T is a tree, so ⟹ unique path from v to w in T.

Let $e' = (v', w')$ be the first cut edge
on this path $v \rightsquigarrow w$.
e safe, $e < e'$. Look $T' = T \cup \{e\} \setminus \{e'\}$
— T' has smaller total weight.
— T' is still a spanning tree.
For any $u, x \in G$,
path $u \rightsquigarrow x \in T$,
path $u \rightsquigarrow v \xrightarrow{e} w \rightsquigarrow x$ so T' is still a spanning tree.

Reducing non-unique to unique.

Some edges could neither safe nor unsafe

e safe $\Longrightarrow$ every MST has e
e unsafe $\Longrightarrow$ no MST has e

Claim: if edge weights are unique, every edge is either safe or unsafe
Corollary: unique edge weights $\Longrightarrow$ exactly one MST.

ex. 1 2 3 3 4

1.001 2.002 3.003 3.004 4.005 ...

Running time:
Boruvka's alg:
in each iteration:
merge each CC with one other.
$\Longrightarrow$ # of CC's cut in half each time.

$\log_2 n$ iters, each iteration

- find the CCs $O(m+n)$
- iterate edge $O(m)$

$O((m+n)\log n)$

Prim's: just like Dijkstra's

fib heap $O(m + n\log n)$

Kruskal's: union-find $O((m+n)\log m)$

		$m=O(n)$	$m=O(n^2)$
Prim	$n\log n + m$	$n\log n$	n^2
Boruvka	$m\log n$	$n\log n$	$n^2\log n$
Kruskal	$(m+n)\log n$	$n\log n$	$n^2\log n$