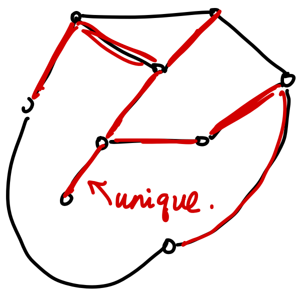
Minium Spanning Tree

Recall. Trees have only 1 path between any 2 verts.

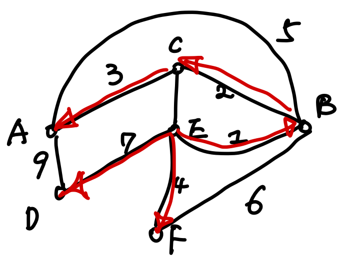
1. Spanning Tree. of a graph, often undirected. is min edges needed to connect all verts in graph.



9 verts, find 8 edges conn the verts.

— : spanning tree.

2. Minium Spanning Trees. On undir. weighted trees. find $|V|-1$ edges connect all verts, with minium of total weight.



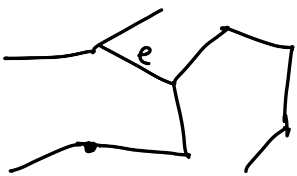
a) Prove optimal Sub-Structure for MST.

PROBLEM. $G = \{V, E\}$, ω_{ij} : weight $i \to j$, undirected.

Find min. spanning tree.

i.e. Subset of E, connect all verts, $\Sigma \omega$ is minimized.

Assume we have the optimal answer.



$E = \{e_1, e_2, \cdots, e_n\}$.

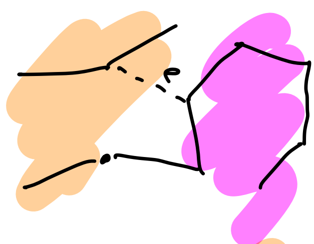
Decition point: Remove any edge from that already optimal answer

$\hookrightarrow$ a new tree spawned.

MST for some graph.

remove e

$G' = \{V', E'\}$



$G'' = \{V'', E''\}$.

find MST in , and connect two.

$V = V' + V''$.

$E = E' + E'' +$ other cross edges

Prf by contradiction, copy and paste method.

b) Greedy algo's correctness

- Pick min edge first but dont form a cycle $\boxed{\text{kruskal}}$.

We need:
- Edges in ascending order
- forming a set?
  DISJOINT SUBSETS (COLOR VERTICES).

Put every vertex in its own set.
do
    Find minium edge $\{u, v\}$
    if FINDSET(u) $\neq$ FINDSET(v)
        use that edge in MST
        UNION (SET(u), SET(v)).
    else discard that edge.
until MST has M-1 edges

Proof by contradiction: "CUT-PASTE"

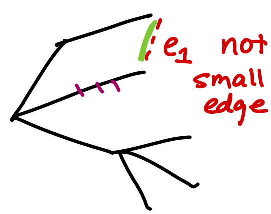
Assume have MST $G = \{V, E\}$

$$e_1 \leq e_2 \leq \cdots \leq e_{|E|-1}$$

that ones not contain the smallest edge

$+ e_1$

↪ induction on other parts



• Pick minium edge from visited vert to unvisited vert. ← no cycle. guaranteed.

$\boxed{\text{PRIM}}$.

We need fibonacci heaps for insert merged.

PROVED OPT. STRUCTURE.

Minium Spanning Trees contains Mininum Spanning Subtrees. Both ~~Prim~~ Prim and Kruscal grows by adding min-edges not forming cycles.

• DELETE MAX edges : prune

▷ STay connected. --- traversal?

no easy way to know.

Can we do divide & conquer on MST?

Exprobs.

5. Which of the following is false?

a) The spanning trees do not have any cycles

b) MST have n – 1 edges if the graph has n ~~edges~~ verts

c) Edge e belonging to a cut of the graph (partitions the vertices of a graph into two disjoint subsets), if has the weight smaller than any other edge in the same cut, then the edge e is present in all the MSTs of the graph

used in any MSTs.

d) Removing one edge from the spanning tree will not make the graph disconnected ✗

7. Kruskal's algorithm (pick minimum edge in the graph between two vertices that are not yet in the same connected component) is best suited for the dense graphs than the Prim's algorithm (pick minimum edge from visited to unvisited vertex).

a) True

b) False

skipping over things.

Fibonacci Heap

visit to unvisit

v(Prim) > v(Krusal)

12. Let e be a maximum-weight edge on some cycle of G = (V, E). Prove that there is a minimum spanning tree of G' = (V, E -{e}) that is also a minimum spanning tree of G. That is, there is a minimum spanning tree of G that does not include e.

13. In this problem, we give pseudo-code for three different algorithms. Each one takes a graph as input and returns a set of edges T. For each algorithm, you must either prove that T is a minimum spanning tree or prove that T is not a minimum spanning tree. Also, describe the most efficient implementation of each algorithm, whether or not it computes a minimum spanning tree.

```
MAYBE-MST-A(G, w)
  sort the edges into nonincreasing order of edge weights w
  T = E
  for each edge e, taken in nonincreasing order by weight
       do if T - {e} is a connected graph
             then T = T - {e}
  return T
```

✓ check will time-consuming.

not minimized

```
MAYBE-MST-B(G, w)
  T = Ø
  for each edge e, taken in arbitrary order
      do if T + {e} has no cycles
            then T = T + {e}
  return T
```

×

```
MAYBE-MST-C(G, w)
  T = Ø
  for each edge e, taken in arbitrary order
     do T = T + {e}

       if T has a cycle c
         then let e' be the maximum-weight edge on c
              T = T - {e'}
  return T
```

✓

this will work: Identifying the cycle (find max).