OBS Notes!

Defns:

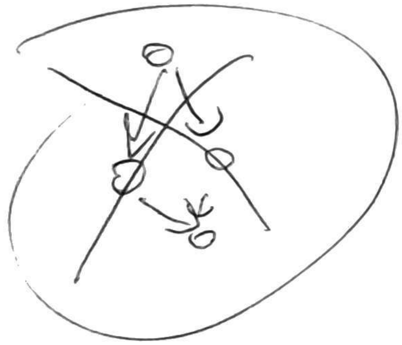
- Branching: tl;dr → just simply a forest where every tree has at most one entering edge per node (i.e. a collection of arborescences):

e.g.: arborescences!

Yes Branching

Yes branching

Yes branching

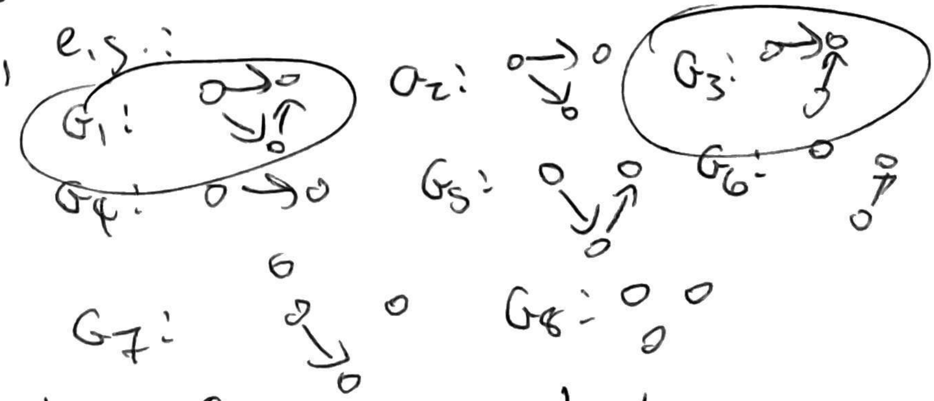


Not a branching

Question: What is a branching in G?

Imagine taking the $2^{|E|}$ set of subgraphs in G, where we either include the edge e or not for all edges e, e.g.:

G:

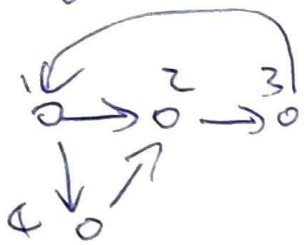


Then, G_i is a branching in G iff G_i is a branching.

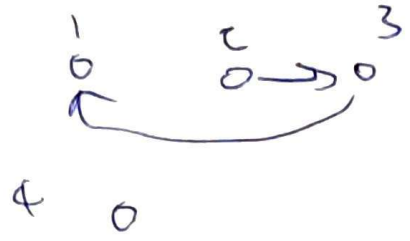
In this case, only G_1 and G_3 are not branchings.

What does specifying $r(i)$-branchings mean?

IMO it means marking certain nodes as not having any edges directed towards it. e.g.:



specifying root nodes 4 and 2 gives:



since 2 and 4 cannot have incoming edges as roots

Question: Are $r(i)$'s the only allowed roots?

(ie does $r \in \{2,4\} \Rightarrow 1 \neq r(i)$).

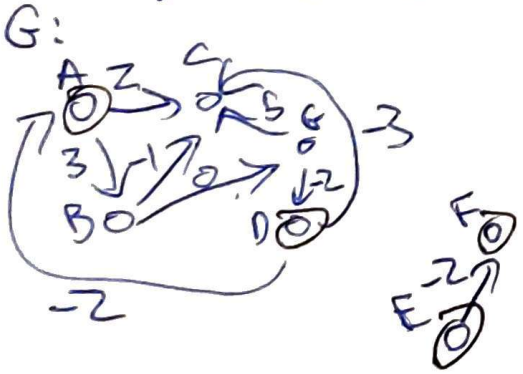
So, paragraph 1 reduces to:

DSS is a question of whether we can use a polynomial time algo. to solve:

Given all subgraphs of a digraph G, who form a branching, and these branchings have root nodes R, what is the least cost collection of these trees, or this branching? e.g.: $V(G) = \{A, \dots, F\}$.

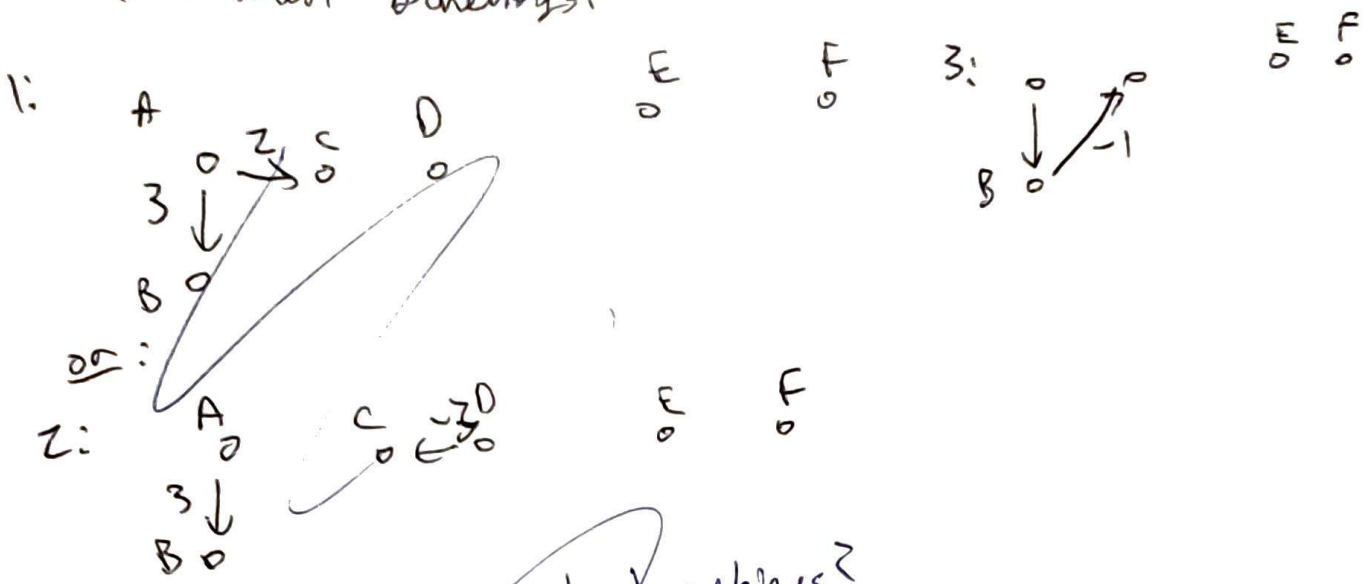
$E(G) = \{(A,C,2), (A,B,3), (B,C,-1), (B,G,0), (G,C,5), (G,D,-2), (D,C,-3), (D,A,2), (E,F,-2)\}$.

G:



$R = \{E, F, A, D\}$

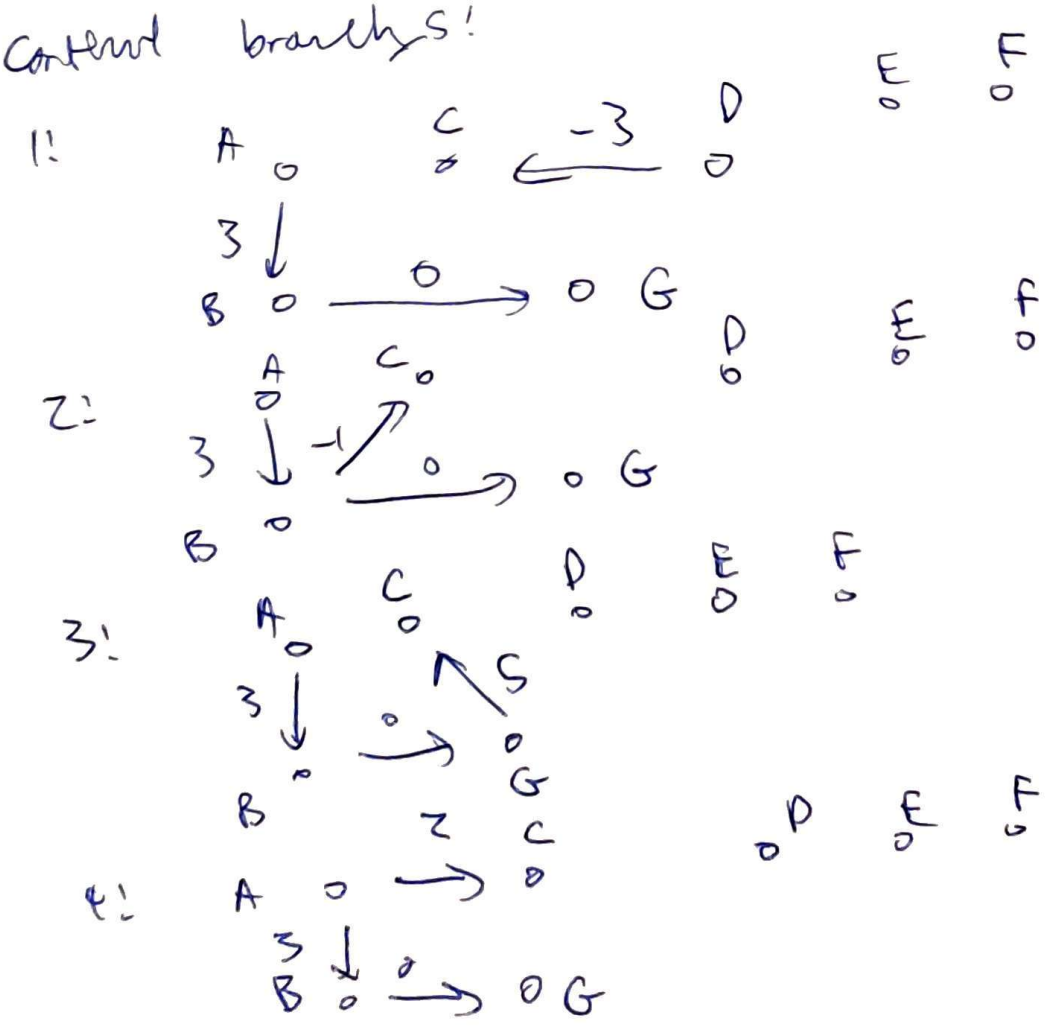
Then, contrived branchings:



Why are these our only branchings?

1. A, D, E, F must be roots!
2. A-C or A-B

Contrived branchings!



So, why are these our only branchings?

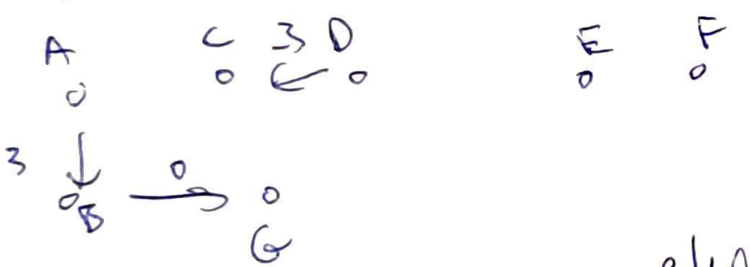
1. Well, A, D, E, F are only roots, so B – G must always exist (only entry to G), similarly, A – B must exist.

Then, C has 4 choices, which we enumerate out. Since A, D are roots, we can't have D – A or G – D. Similarly, E – F is good.

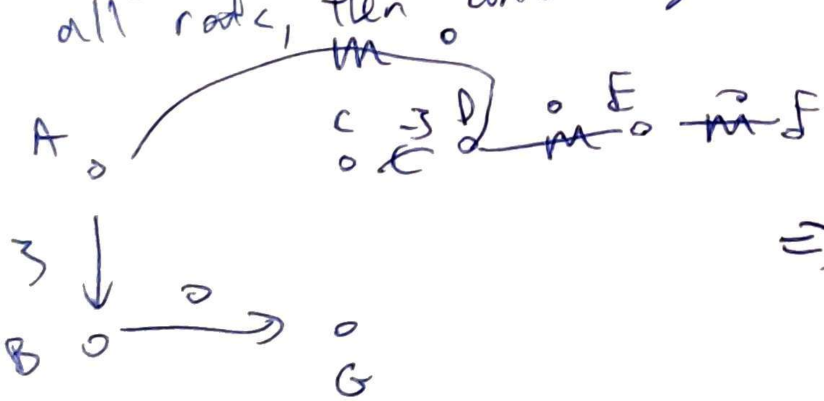
<u>Paragraph 2:</u>

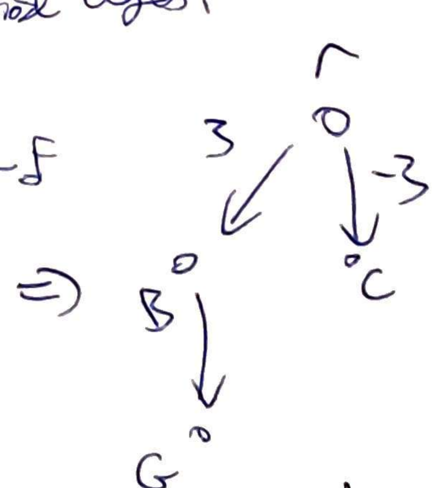
what does reducing all root nodes r(i) to same mean? well, we can reformulate the problem s.t. we get a single root node so that it's easier to think about.

Take: #1 from before!



Imagine building a new edge of weight 0 between all roots, then contracting those edges!





we're left with an arborescence!

So, what is k? k describes the # of connections you make!
Or rather, # of arborescences you combine into 1.

So, for the previous example, we had $k=4$.

In general, $k = 1 \dots |V(G)|$ because we have at most k spanning trees to make the graph.

Why does this matter?

1. Based on criterion, any brute-force algorithm would take exponential time.
2. Matroid Intersection nicely solves this issue.
3. Unlike optimal branchings, we define a set of roots, or a potential # of roots.