Foundations of Data Mining

Homework Assignment 1C

Frequent Subsequence and Subgraph Mining

Homework group 4:

Arngrímur Einarsson
Stefán G. Jónsson
Guðmundur Orri Pálsson
Nick Geerjens
Troy Maasland

Frequent Subsequence Mining

We impose a minimal support constraint of

$s_{min} = 0.75$

Homework 1C

Data:

Sequence ID	Sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(ab)cbc>

minimal support: $s_{min} = 0.75$

A. Initial candidates: <a>, <b>, <c>, <d>, <e>, <f>, <g>

Candidate	Support
<a>	1
<b>	1
<c>	1
<d>	0.75
<e>	0.75
<f>	0.5
<g>	0.25

-> frequent length-1 candidate sequences

GSP

-> Scan dataset once find all frequent items

Repeat until done:

-> feed seed set of sequential patterns to join operator, which determines new set of candidates

-> Scan dataset once to determine support of new candidate set prune those that do not satisfy minimal support constraint.

Resulting set: seed for next iteration

B. All combinations of the frequent length-1 candidate subsequences are length-2 candidate subsequences

Also all combinations of the frequent length-1 candidate subsequences in a single event are length-2 candidate subsequences (for which the order does not matter)

If N is the amount of frequent length-1 candidate subsequences, then this leaves us with

N^2 combinations

$N(N-1)/2$ events

$\rightarrow N^2 + N(N-1)/2$ length-2 candidate subsequences

length-2

Candidate subsequences:

	⟨a⟩	⟨b⟩	⟨c⟩	⟨d⟩	⟨e⟩
⟨a⟩	⟨aa⟩	⟨ab⟩	⟨ac⟩	⟨ad⟩	⟨ae⟩
⟨b⟩	⟨ba⟩	⟨bb⟩	⟨bc⟩	⟨bd⟩	⟨be⟩
⟨c⟩	⟨ca⟩	⟨cb⟩	⟨cc⟩	⟨cd⟩	⟨ce⟩
⟨d⟩	⟨da⟩	⟨db⟩	⟨dc⟩	⟨dd⟩	⟨de⟩
⟨e⟩	⟨ea⟩	⟨eb⟩	⟨ec⟩	⟨ed⟩	⟨ee⟩

	⟨a⟩	⟨b⟩	⟨c⟩	⟨d⟩	⟨e⟩
⟨a⟩		⟨(ab)⟩	⟨(ac)⟩	⟨(ad)⟩	⟨(ae)⟩
⟨b⟩			⟨(bc)⟩	⟨(bd)⟩	⟨(be)⟩
⟨c⟩				⟨(cd)⟩	⟨(ce)⟩
⟨d⟩					⟨(de)⟩
⟨e⟩					

Now for all candidates we have to check the support. Here, only the candidates with high enough support (0.75) are written down: (so that the following are the frequent).

⟨ab⟩: s = 0.75 ⟨ac⟩: s = 1 ⟨cc⟩: s = 0.75

⟨cb⟩: s = 0.75 ⟨bc⟩: s = 0.75 ⟨dc⟩: s = 0.75

⟨(ab)⟩: s = 0.75

4. $N_f = 5$, $N_{2C} = N(N-1)/2 + N^2$

$= (5 \cdot 4)/2 + 5^2$

$= 35$ length-2 candidate subsequences

We would have generated $(7 \cdot 6)/2 + 7^2 = 70$ length-2 candidate subsequences if we did not prune away infrequent length-1 candidate subsequences

5. $\langle ab \rangle, \langle cb \rangle, \langle ac \rangle, \langle bc \rangle, \langle cc \rangle, \langle dc \rangle, \langle (ab) \rangle$

Join operation

Given a seed set of sequential patterns the join operation

	$\langle ab \rangle$	$\langle cb \rangle$	$\langle ac \rangle$	$\langle bc \rangle$	$\langle cc \rangle$	$\langle dc \rangle$	$\langle (ab) \rangle$
$\langle ab \rangle$	X	X	X	$\langle abc \rangle$	X	X	X
$\langle cb \rangle$	X	X	X	$\langle cbc \rangle$	X	X	X
$\langle ac \rangle$	X	$\langle acb \rangle$	X	X	$\langle acc \rangle$	X	X
$\langle bc \rangle$	X	$\langle bcb \rangle$	X	X	$\langle bcc \rangle$	X	X
$\langle cc \rangle$	X	$\langle ccb \rangle$	X	X	$\langle ccc \rangle$	X	X
$\langle dc \rangle$	X	$\langle dcb \rangle$	X	X	$\langle dcc \rangle$	X	X
$\langle (ab) \rangle$	X	X	X	$\langle (ab)c \rangle$	X	X	X

So, the length-3 candidate sequences are

$\langle acb \rangle, \langle bcb \rangle, \langle ccb \rangle, \langle dcb \rangle, \langle abc \rangle, \langle cbc \rangle, \langle (ab)c \rangle, \langle acc \rangle, \langle bcc \rangle, \langle ccc \rangle, \langle dcc \rangle$

6.

c.

A sequential pattern s is closed if no strict superpattern s' ⊃ s exists such that s' and s have the same support

<acb>: support: 0.75

Superpatterns: <acba>, <acbb>, <acbc>, <acbd>, <acbe>, <acbf>, <acbg>

f, g support < 0.75, so those are left out

<acba>: support 0.25

<acbb>: support 0

<acbc>: support 0.25

<acbd>: support 0

<acbe>: support 0.25

So, yes, <acb> is closed within Ω

<adc>: support = 0.5

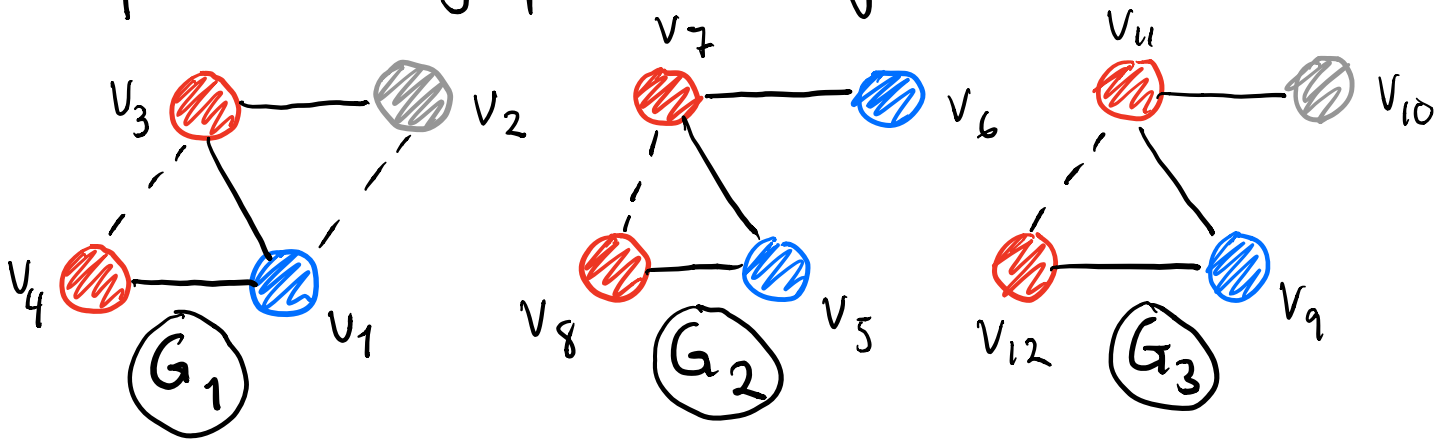
<adca>: s = 0

<adcb>: s = 0.25

The rest does not have to be checked. Only two sequences contain <adc>. The first has only an f after the c, within the same event (which does not count)

→ support is at maximum 0.25

Frequence Subgraph Mining



5. Characterize each of the graphs G_i by writing down their vertex set V_i, their edge set E_i, their label set L_i, and their label function $l_i : V_i \cup E_i \to L_i$. For their edge sets, you may choose yourself if you want to write them down by the adjency matrix or the adjency list

Solution: We get

$G_1 = (V(G_1), E(G_1))$

$V(G_1) = \{v_1, v_2, v_3, v_4\}$

$E(G_1) = \{E_1 = (v_1, v_2),\ E_2 = (v_1, v_3),\ E_3 = (v_2, v_3),\ E_4 = (v_1, v_4),\ E_5 = \{v_3, v_4\}\}$

$\ell(v_1) =$ BLUE, $\ell(v_2) =$ GRAY, $\ell(v_3) =$ RED, $\ell(v_4) =$ RED

$\ell(E_1) =$ dotted, $\ell(E_2) =$ straight, $\ell(E_3) =$ straight, $\ell(E_4) =$ straight, $\ell(E_5) =$ dotted

$\mathcal{L}_1 = \{$BLUE, RED, GRAY, dotted, straight$\}$

$G_2 = (V(G_2), E(G_2))$

$V(G_2) = \{v_5, v_6, v_7, v_8\}$

$E(G_2) = \{E_1 = (v_5, v_8),\ E_2 = (v_5, v_7),\ E_3 = (v_6, v_7),\ E_4 = (v_7, v_8)\}$

$\ell(v_5) =$ BLUE, $\ell(v_6) =$ BLUE, $\ell(v_7) =$ RED, $\ell(v_8) =$ RED

$\ell(E_1) =$ straight, $\ell(E_2) =$ straight, $\ell(E_3) =$ straight, $\ell(E_4) =$ dotted

$\mathcal{L}_2 = \{$BLUE, RED, dotted, straight$\}$

$G_3 = (V(G_3), E(G_3))$

$V(G_3) = \{v_9, v_{10}, v_{11}, v_{12}\}$

$E(G) = \{E_1 = (v_9, v_{12}),\ E_2 = (v_9, v_{11}),\ E_3 = (v_{10}, v_{11}),\ E_4 = (v_{11}, v_{12})\}$

$\ell(v_9) =$ BLUE, $\ell(v_{10}) =$ GRAY, $\ell(v_{11}) =$ RED, $\ell(v_{12}) =$ RED

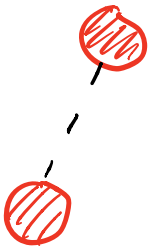
$\ell(E_1) =$ straight, $\ell(E_2) =$ straight, $\ell(E_3) =$ straight, $\ell(E_4) =$ dotted

$\mathcal{L}_2 = \{$BLUE, RED, GRAY, dotted, straight$\}$

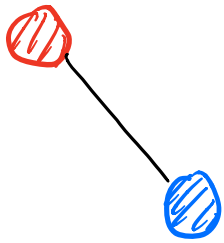
---

9. Give all frequent 1-subgraphs

Solution: We have $s_{min} = 2/3$

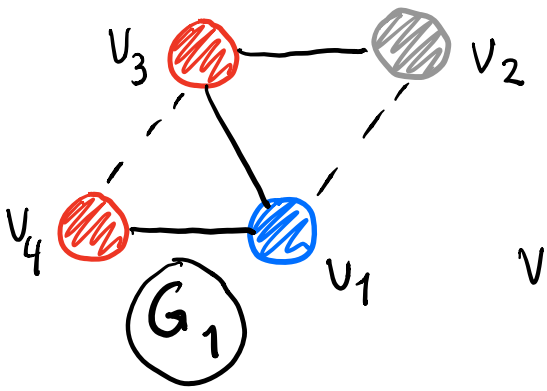


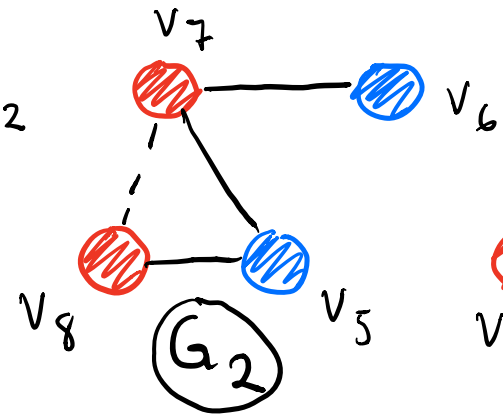
$s = \frac{3}{3}$

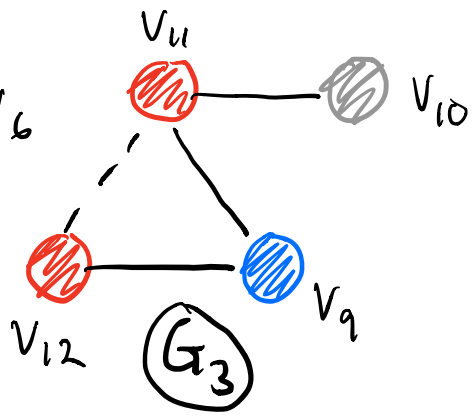


$s = \frac{4}{3}$

$s = \frac{2}{3}$

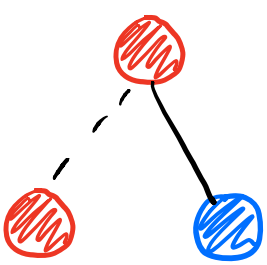




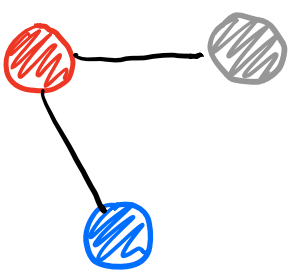


0. Give all frequent 2-subgraphs

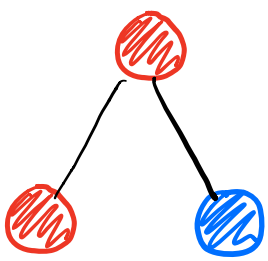
Solution: We get



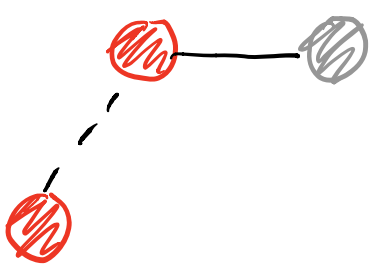
$s = \frac{4}{3}$



$s = \frac{2}{3}$



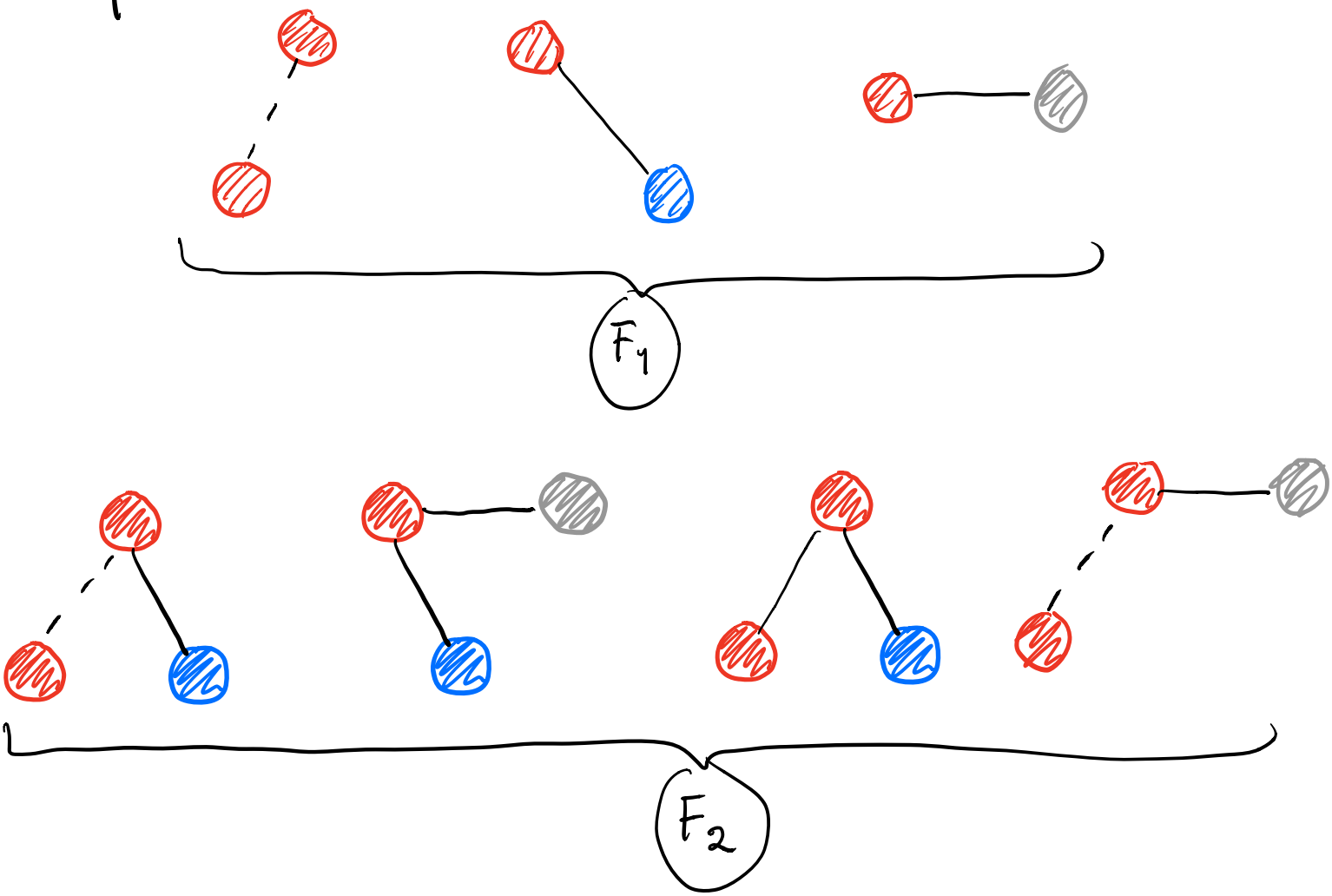
$s = \frac{3}{3}$



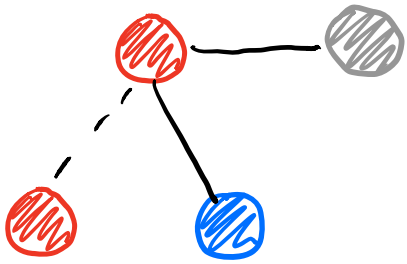
$s = \frac{2}{3}$

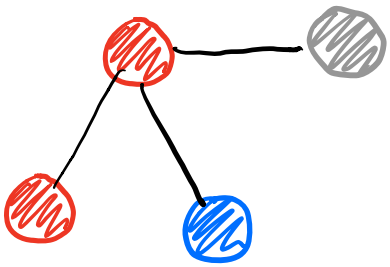
4. Use the FSG algorithm, without any optimizations, to find the set of all frequent subgraphs.

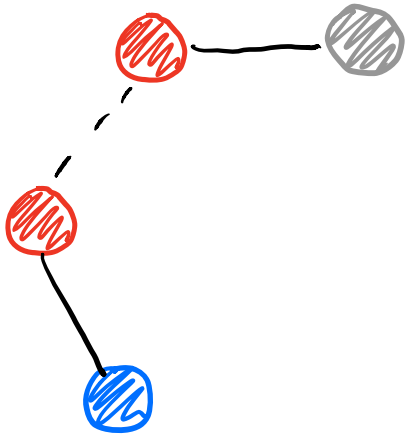
Solution. We have F_1 and F_2 from previous steps

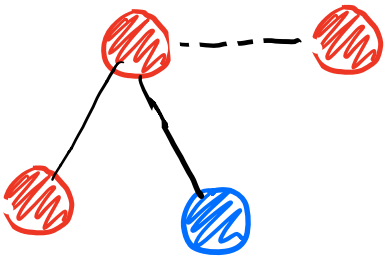


We generate candidates

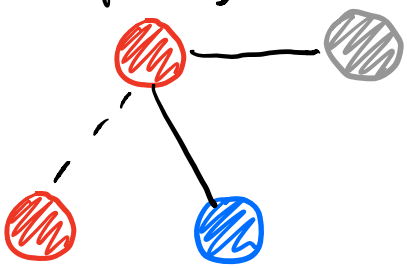


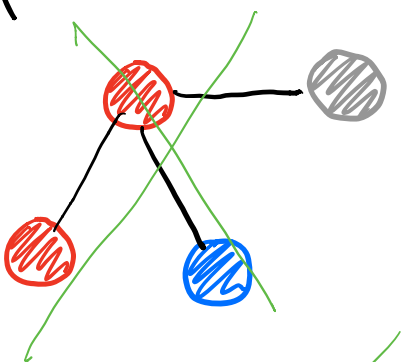


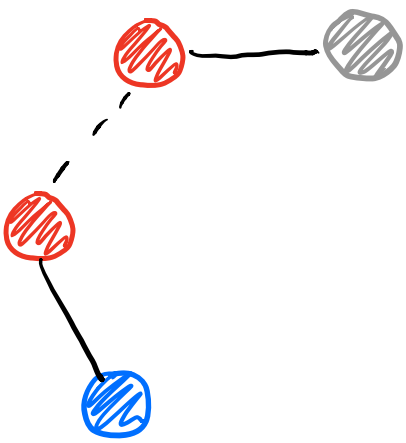


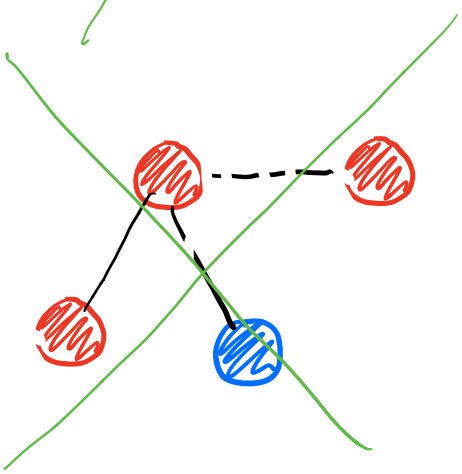


Candidates that do not satisfy antimonotonocity property are pruned

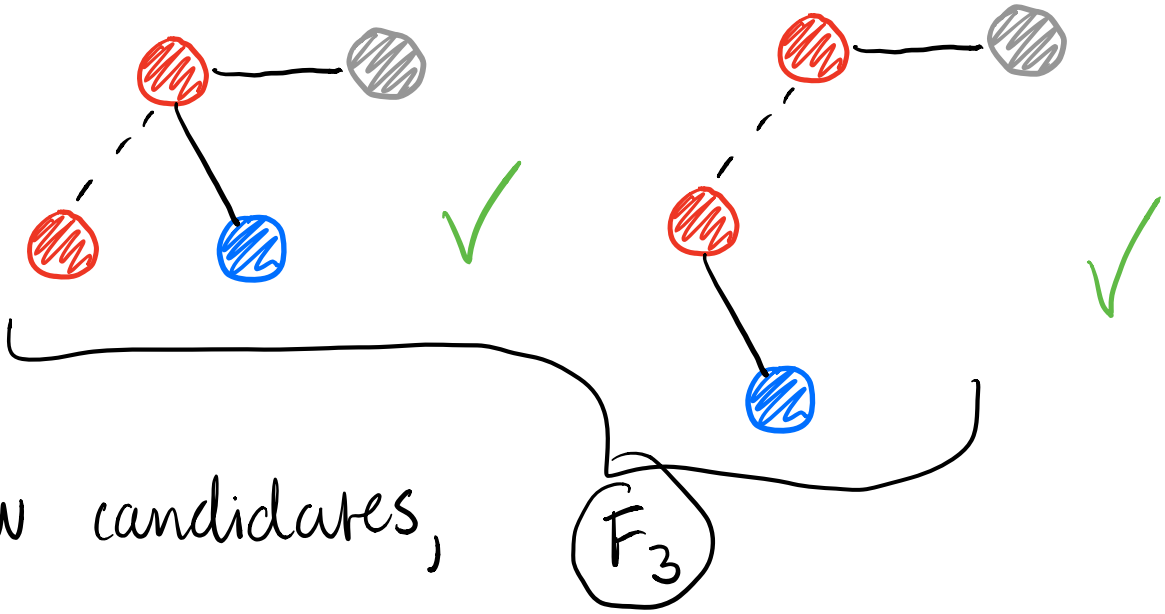




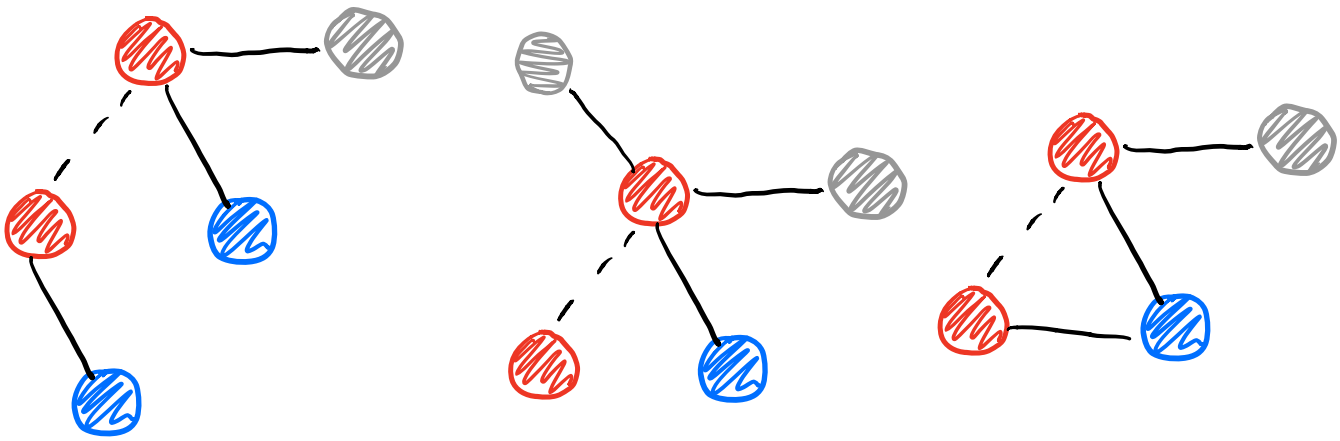




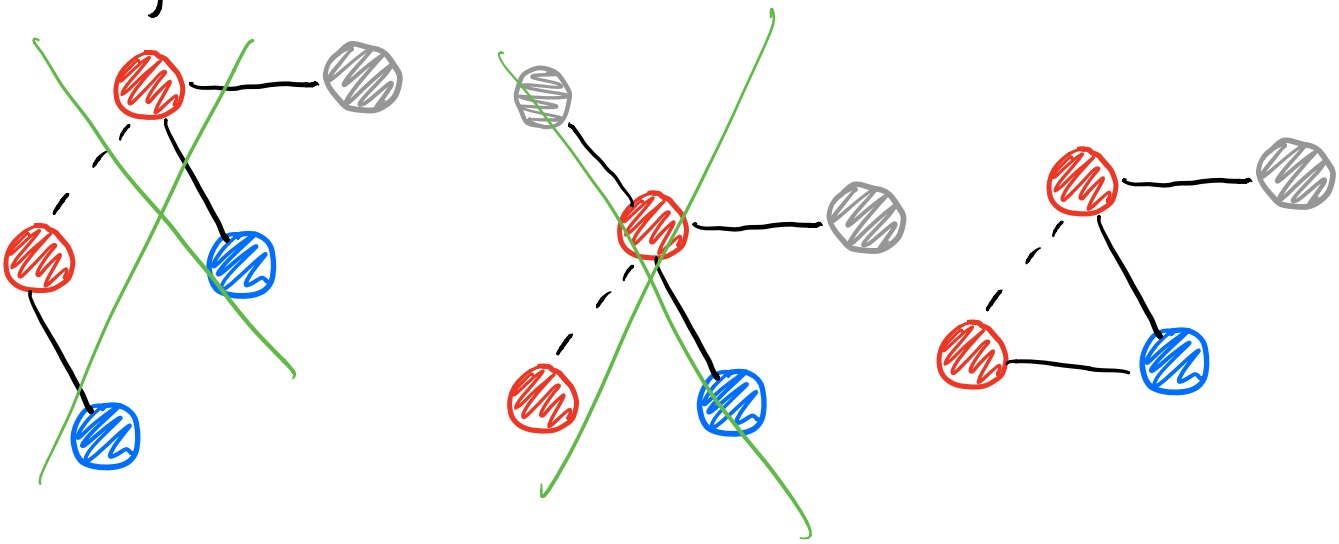
Check if candidates satisfy minimal support constraint, $s_{min} = 2/3$



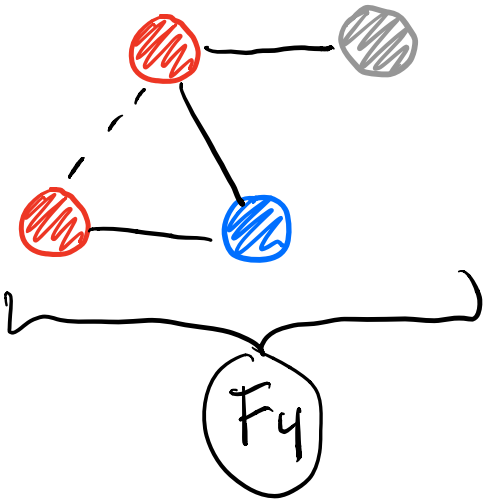
New candidates, F_3

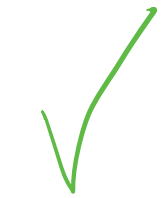


Prune,



Check minimum support:





We return $F_1 \cup F_2 \cup F_3 \cup F_4$