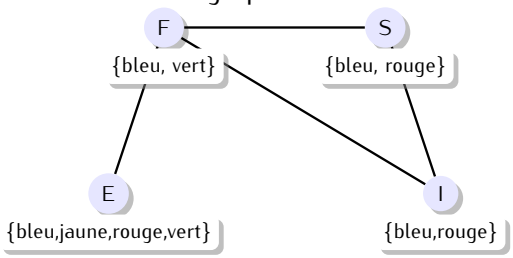
Raisonnement par Contraintes

On considère le problème de coloration de graphe suivant :



1. Utiliser l'algorithme `Backtrack` ci-dessous pour résoudre le problème. On examinera les variables dans l'ordre F, E, S, I ; et on examinera les variables dans l'ordre lexicographique.

```
1  BackTrack (CSP net, Affectation a)
2    si l'affectation a est complète alors retourne a
3    var ← variable suivante non affectée
4    Pour toutes valeurs var ∈ D(var)
5      si {var=val} ne viole aucune contrainte
6          a = a ∪ {var=val}
7          result ← Backtrack(net, a)
8          si result ≠ échec
9              retourne result
10         sinon retourne échec
```

2. Une fois qu'on a mis à jour l'affectation, on va maintenant utiliser la propagation de contraintes pour limiter le nombre de valeurs possibles.

```
1  ForwardChecking (CSP net, Affectation a)
2    si l'affectation a est complète alors retourne a
3    var ← variable suivante non affectée
4    Pour toutes valeurs var ∈ D(var)
5      si {var=val} ne viole aucune contrainte
6          a = a ∪ {var=val}
7          pour toutes variables v connectée à var
8              verifier arc-cohérence de var et v
9          result ← Backtrack(net, a)
10         si result ≠ échec
11             retourne result
12         sinon retourne échec
```

   Utiliser `ForwardChecking` pour trouver une solution en conservant les mêmes ordres que dans la question 1 pour examiner les variables et les valeurs.

3. utiliser des heuristiques pour le choix des valeurs et des variables
   — choisir la variable qui a le moins de valeurs disponibles
   — choisir la variable qui a le plus de contraintes avec des variables non affectées.
   — choisir la valeur qui contraint le moins les voisins