

Parcours de graphes

NSI2

19 février 2023

Orientés ou non orientés ?

On peut considérer des graphes orientés ou non. Cela ne change pas grand chose au principe :

- dans le cas d'un graphe non orienté, on parcourt un graphe de voisins en voisins;
- si le graphe est orienté on le parcourt de successeurs en successeurs.

Parcours en profondeur

À partir d'un sommet de départ, on va chercher à aller :

1. le plus « loin » possible d'abord.
2. dans toutes les directions.

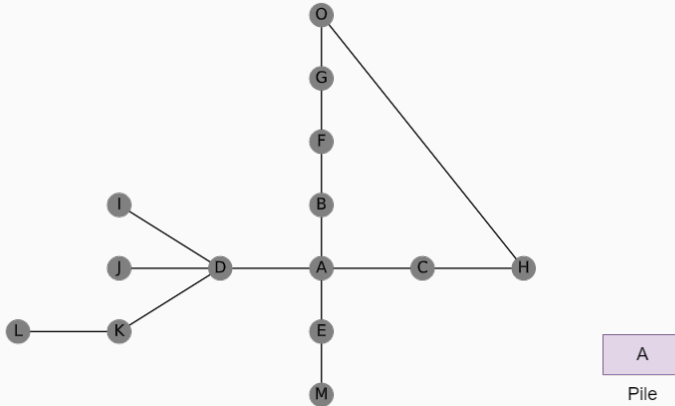
Le parcours en profondeur est appelé *Depth First Search* ou DFS en anglais.

Étant donné un graphe et un sommet de départ, il n'y a pas en général qu'un seul parcours en profondeur qui commence par ce sommet car l'ordre dans lequel on choisit chaque voisin d'un sommet est *a priori* arbitraire.

On peut implémenter un parcours DFS de manière itérative en utilisant une **pile** dans laquelle on empile les prochains sommets à traiter (voisins du sommet en cours).
Il faut cependant faire attention à ne pas empiler plusieurs fois un sommet, c'est pourquoi on crée une liste des sommets non empilés.

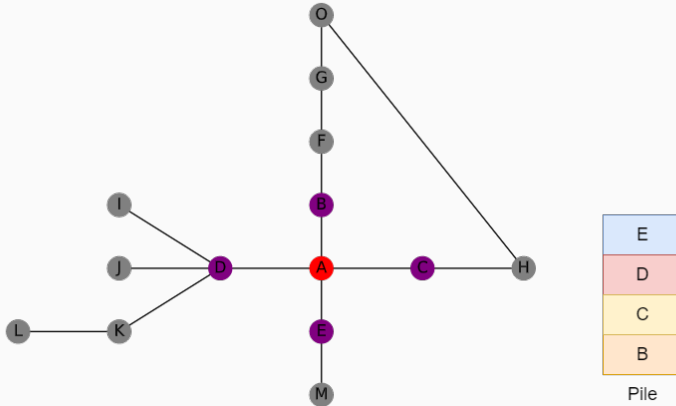
Exemple

- sommet non visité non empilé
- voisin non visité mais empilé
- sommet en cours de visite
- sommet déjà visité



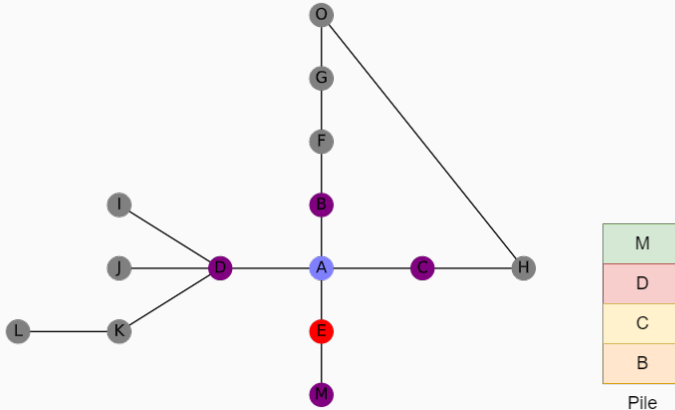
Exemple

- sommet non visité non empilé
- voisin non visité mais empilé
- sommet en cours de visite
- sommet déjà visité



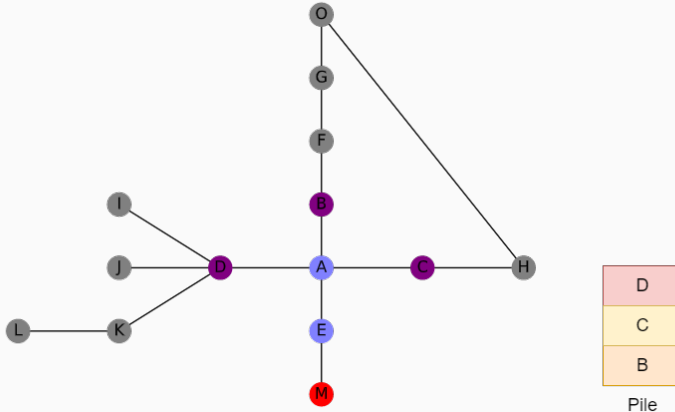
Exemple

- sommet non visité non empilé
- voisin non visité mais empilé
- sommet en cours de visite
- sommet déjà visité



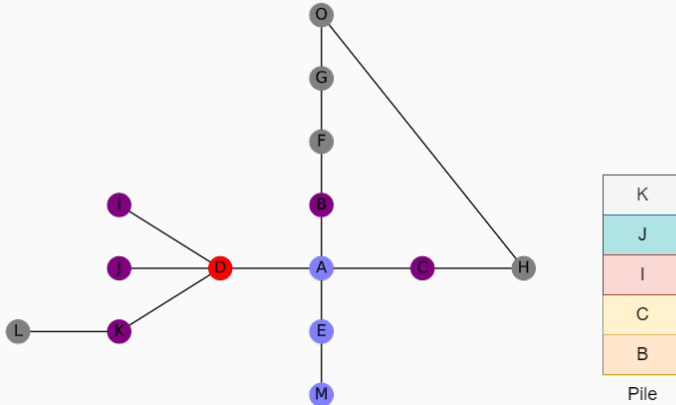
Exemple

- sommet non visité non empilé
- voisin non visité mais empilé
- sommet en cours de visite
- sommet déjà visité



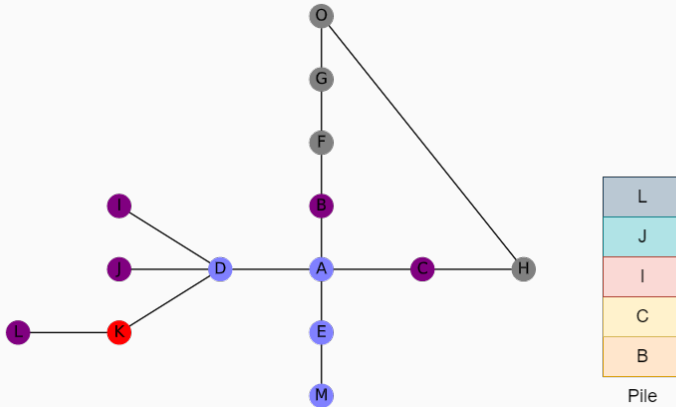
Exemple

- sommet non visité non empilé
- voisin non visité mais empilé
- sommet en cours de visite
- sommet déjà visité



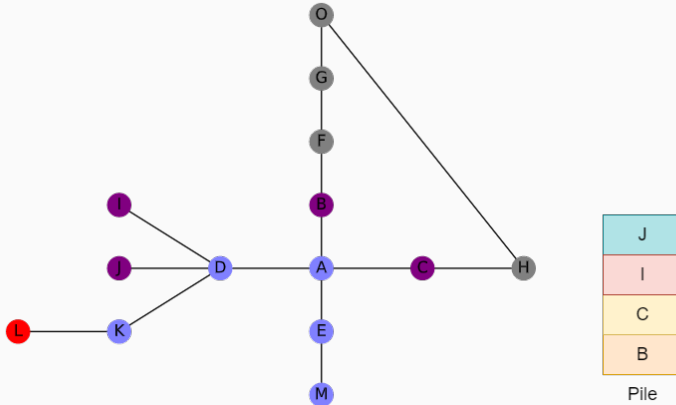
Exemple

- sommet non visité non empilé
- voisin non visité mais empilé
- sommet en cours de visite
- sommet déjà visité



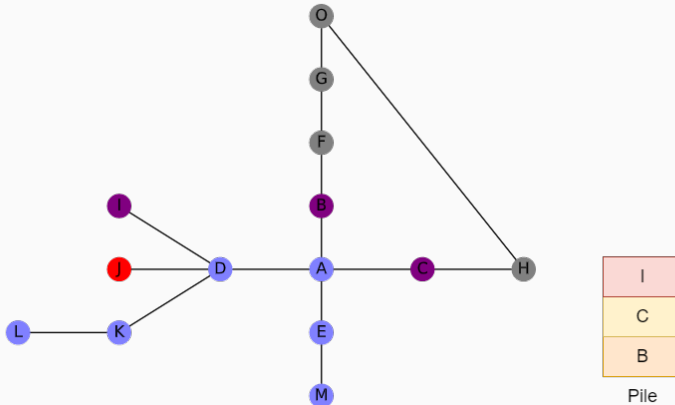
Example

- sommet non visité non empilé
- voisin non visité mais empilé
- sommet en cours de visite
- sommet déjà visité



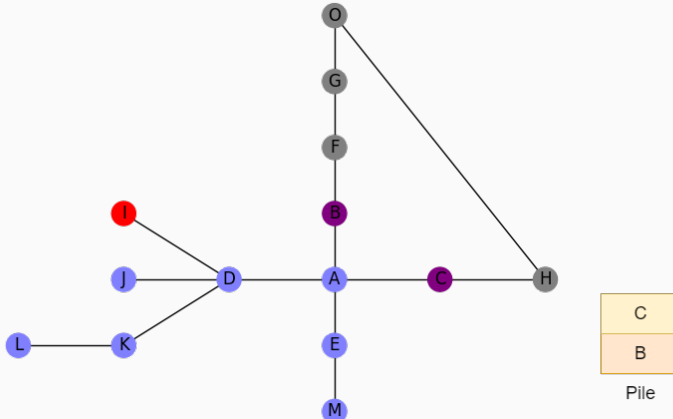
Exemple

- sommet non visité non empilé
- voisin non visité mais empilé
- sommet en cours de visite
- sommet déjà visité



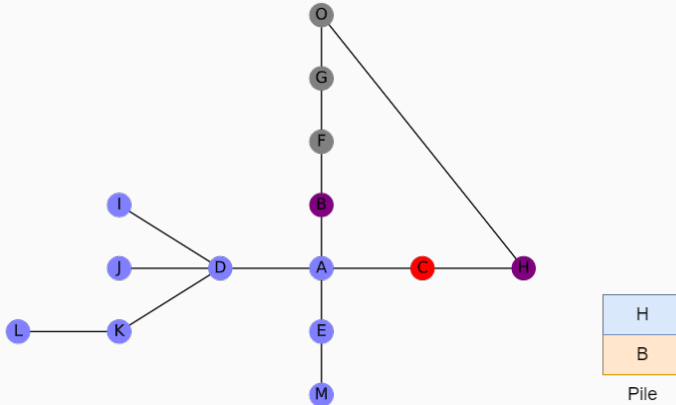
Exemple

- sommet non visité non empilé
- voisin non visité mais empilé
- sommet en cours de visite
- sommet déjà visité



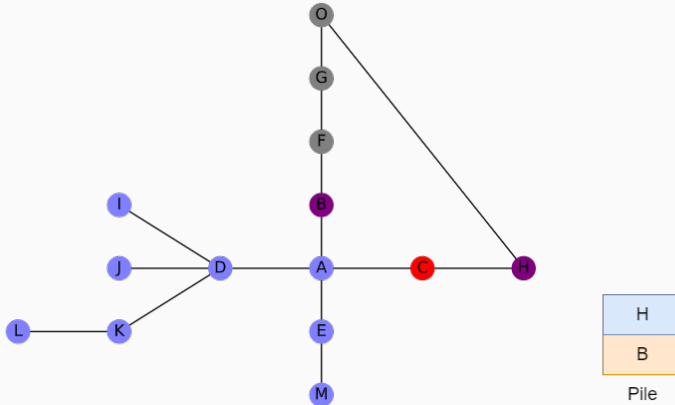
Exemple

- sommet non visité non empilé
- voisin non visité mais empilé
- sommet en cours de visite
- sommet déjà visité



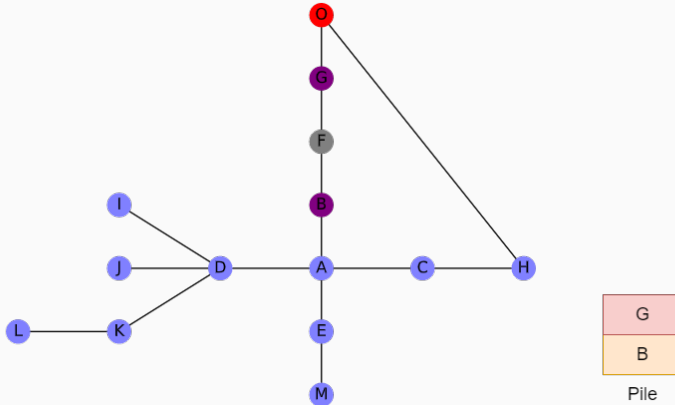
Exemple

- sommet non visité non empilé
- voisin non visité mais empilé
- sommet en cours de visite
- sommet déjà visité



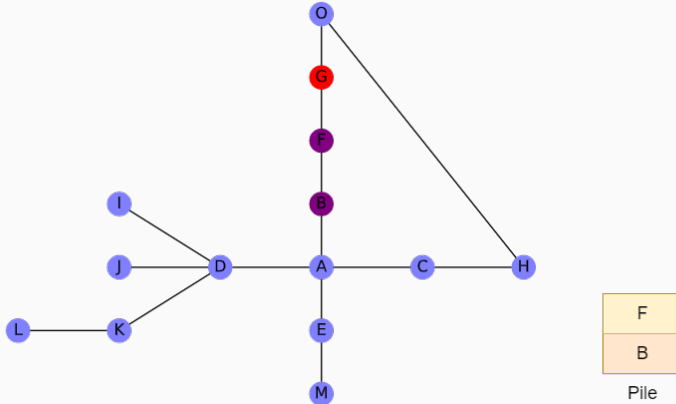
Exemple

- sommet non visité non empilé
- voisin non visité mais empilé
- sommet en cours de visite
- sommet déjà visité



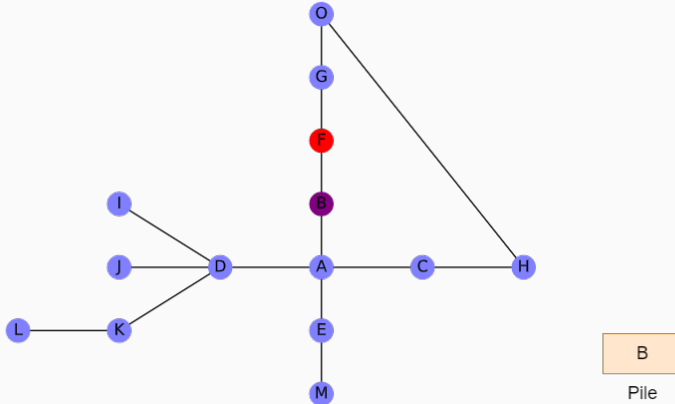
Exemple

- sommet non visité non empilé
- voisin non visité mais empilé
- sommet en cours de visite
- sommet déjà visité



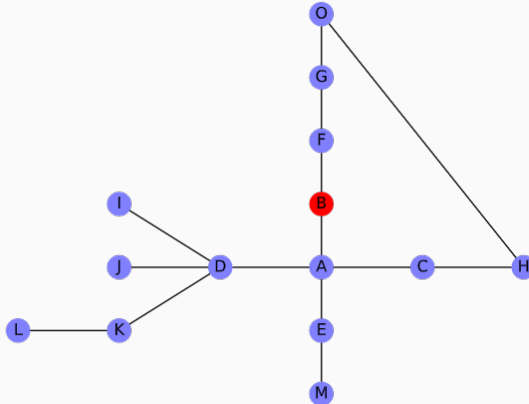
Exemple

- sommet non visité non empilé
- voisin non visité mais empilé
- sommet en cours de visite
- sommet déjà visité



Exemple

- sommet non visité non empilé
- voisin non visité mais empilé
- **sommet en cours de visite**
- sommet déjà visité



Pile

Implémentation itérative

```
fonction dfs( G : graphe, d : départ ) -> liste
variables
  p : pile
  sommets_restants, parcours : liste
début
  sommets_restants = sommets_de_G
  empiler d sur p
  retirer d de sommets_restants
  tant que p n'est pas vide :
    s = dépiler de p
    ajouter s à parcours
    pour chaque voisin v de s :
      si v est dans sommets_restants :
        empiler v sur p
        retirer v de sommets_restants
  renvoyer parcours
fin
```

On peut également implémenter le parcours DFS de manière récursive.

Parcours en largeur

À partir d'un sommet de départ, on va chercher à visiter

1. tous les voisins directs d'abord, dans toutes les « directions » possibles.
2. puis les voisins des voisins et ainsi de suite le plus loin possible.

Le parcours en largeur est appelé *Breadth First Search* ou BFS en anglais.

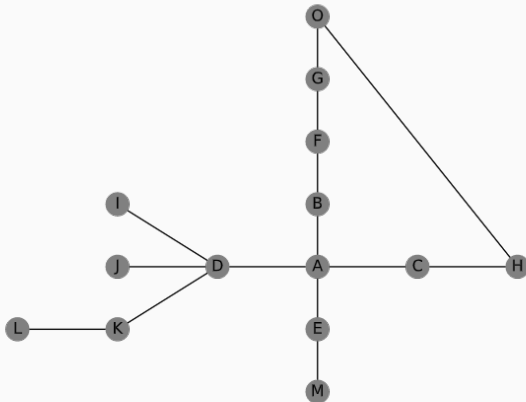
Tout comme DFS, BFS dépend de l'ordre dans lequel on va parcourir les sommets voisins, il n'y a donc pas (en général) unicité du parcours en largeur partant d'un sommet donné.

On peut implémenter un parcours BFS de manière identique à DFS, à ceci près qu'on utilise une **file** à la place de la pile.

Implémentation itérative

```
fonction bfs( G : graphe, d : départ ) -> liste
variables
  f : file
  sommets_restants, parcours : liste
début
  sommets_restants = sommets_de_G
  enfiler d dans f
  retirer d de sommets_restants
  tant que f n'est pas vide :
    s = défiler de f
    ajouter s à parcours
    pour chaque voisin v de s :
      si v est dans sommets_restants :
        enfiler v dans f
        retirer v de sommets_restants
  renvoyer parcours
fin
```

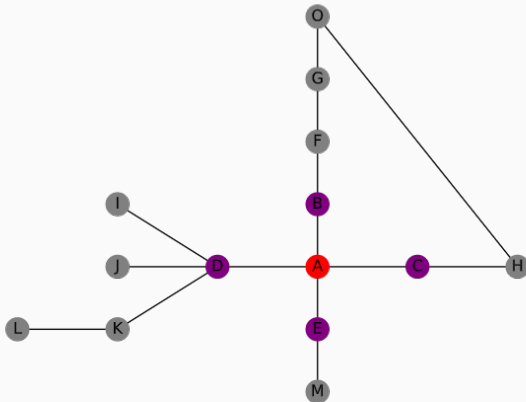
Exemple



> File >

A

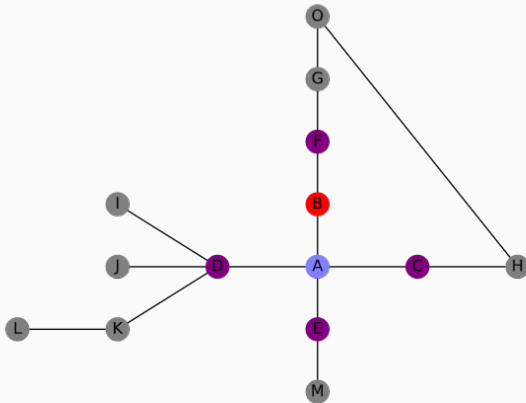
Exemple



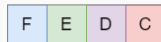
> File >



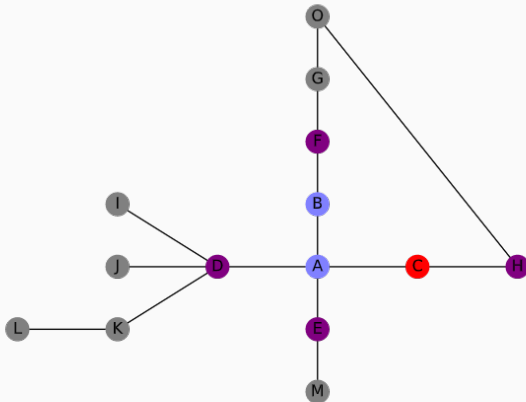
Exemple



> File >



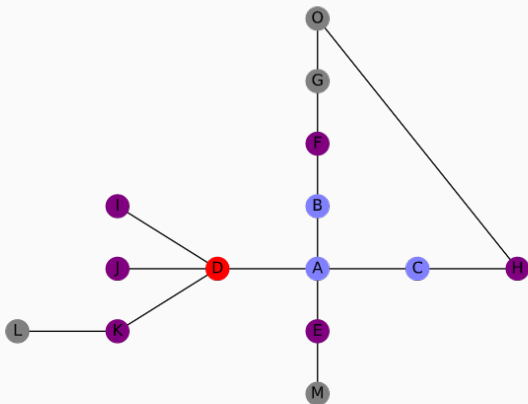
Exemple



> File >



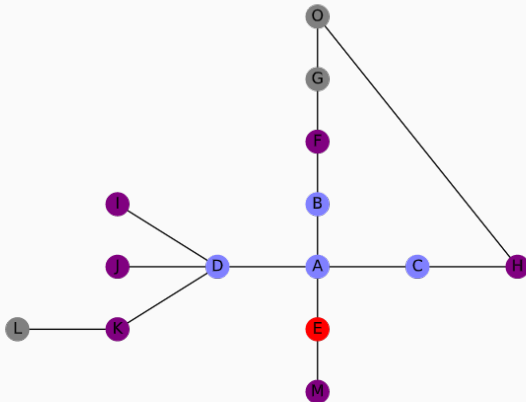
Exemple



> File >



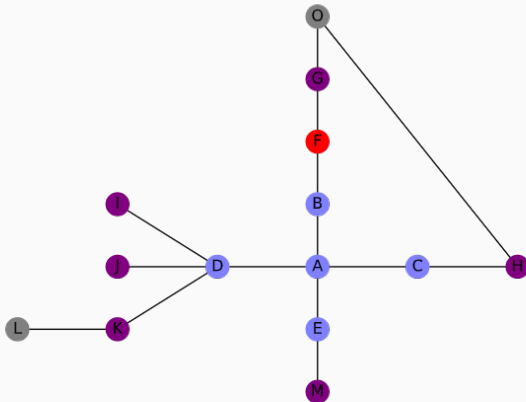
Exemple



> File >



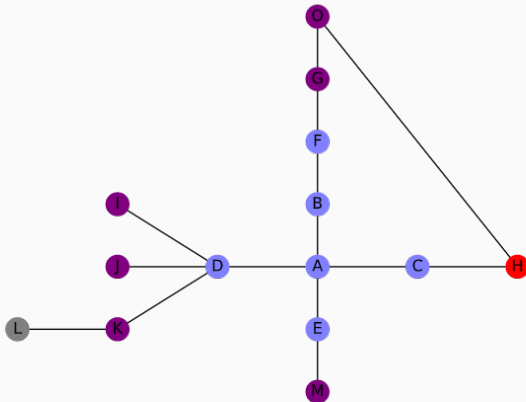
Exemple



> File >



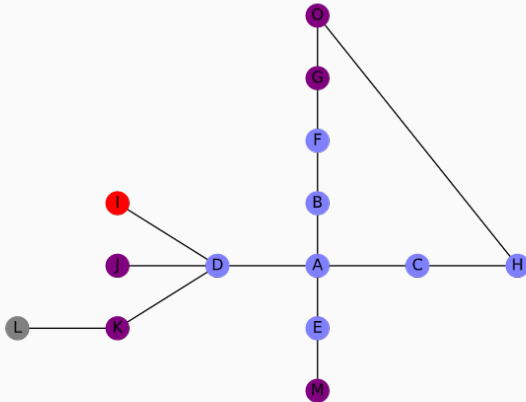
Exemple



> File >



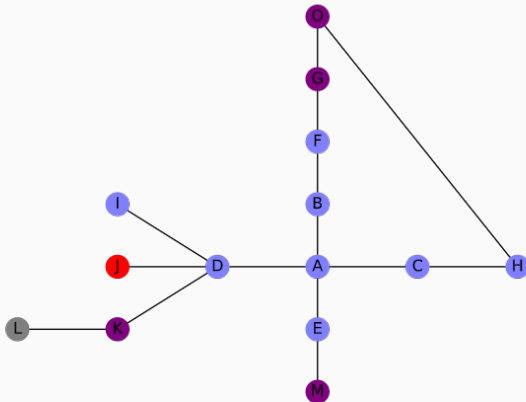
Exemple



> File >



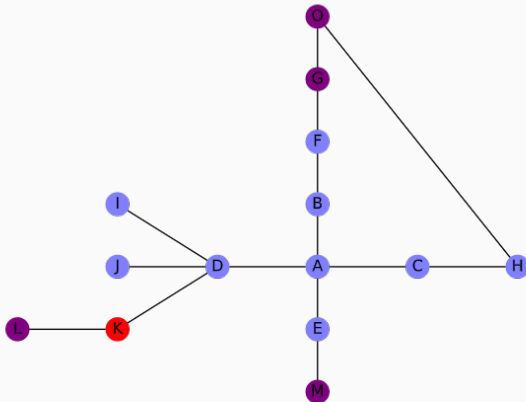
Exemple



> File >



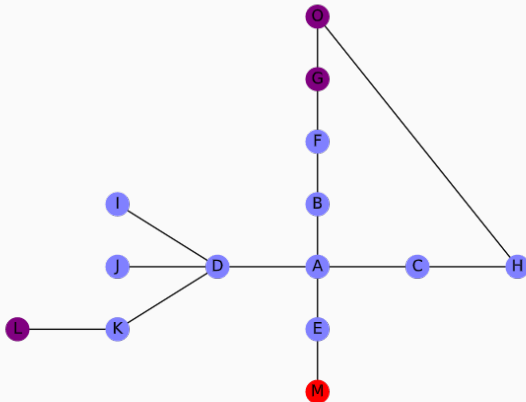
Exemple



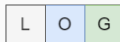
> File >



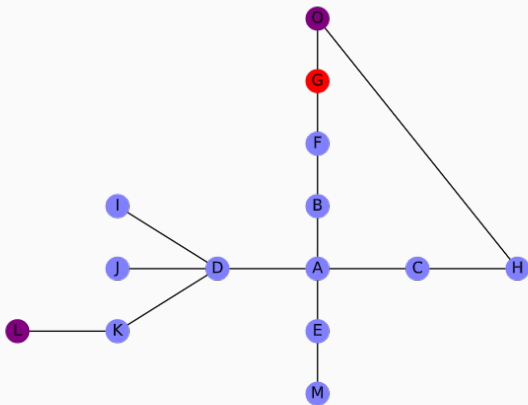
Exemple



> File >



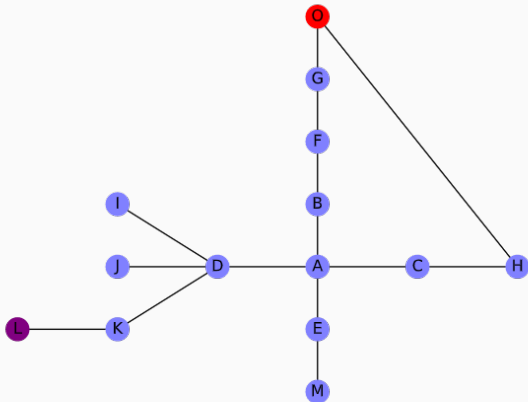
Exemple



> File >



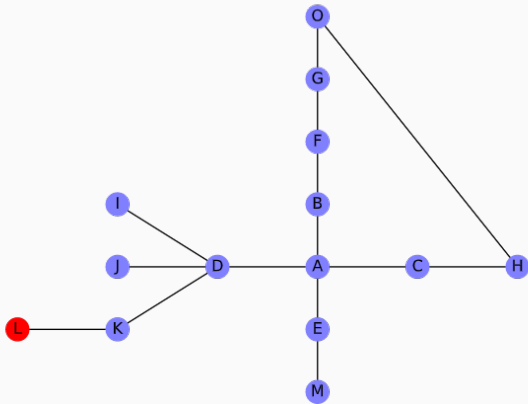
Exemple



> File >



Exemple



> File >