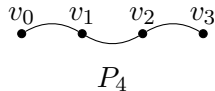


Chaînes

Définition

Une chaîne dans un graphe $G = (V, E)$ est une suite de la forme $(v_0, e_1, v_1, \dots, e_k, v_k)$, où :

- $v_i \in V$
- $e_i \in E$
- $e_{i+1} = v_i v_{i+1}$.



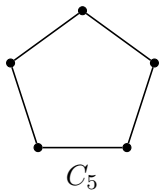
- L'entier k est la longueur de la chaîne.
- Une chaîne est élémentaire si ses sommets sont deux-à-deux distincts.
- Une chaîne élémentaire avec n sommets (de longueur $n - 1$) est notée P_n .

Chaînes (suite)

- Les sommets v_0 et v_k sont les extrémités de la chaîne.
- Lorsque le graphe est simple (sans arêtes parallèles), une chaîne peut être définie simplement par la suite (v_0, \dots, v_k) de ses sommets.
- Une sous-chaîne d'une chaîne est définie comme une sous-suite, entre deux sommets, de la suite définissant la chaîne considérée.
- Une chaîne est dite simple si ses arêtes sont deux-à-deux distinctes.
- Élémentaire entraîne simple.

Définition

Un cycle est une chaîne de longueur supérieure ou égale à 1 simple et fermée. C'est donc une chaîne de la forme $(v_0, e_1, v_1, \dots, e_k, v_0)$ où $k \geq 1$ et les e_i sont distinctes.

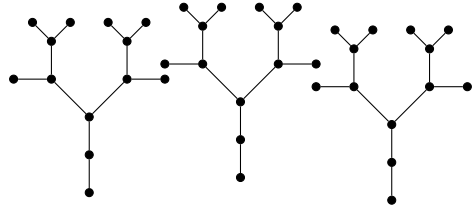
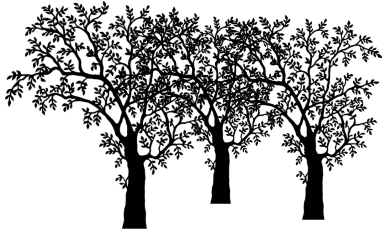


- L'entier k est la longueur du cycle ; le cycle est pair où impair suivant que sa longueur est paire ou impaire.
- Un cycle élémentaire avec n sommets (de longueur n) est noté C_n .

Forêts

Définition

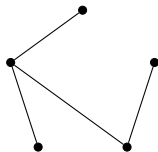
Un graphe sans cycles s'appelle un graphe acyclique, ou une forêt.



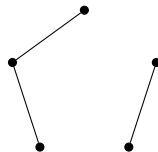
Connexité

Définition

- Un graphe G est connexe s'il existe une chaîne entre u et v , pour toute paire de sommets $u, v \in V(G)$.
- Une composante connexe d'un graphe G est un sous-graphe connexe maximal (par inclusion).



graphe connexe

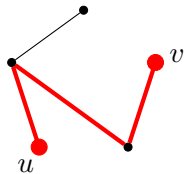


graphe non connexe

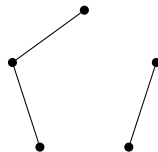
Connexité

Définition

- Un graphe G est connexe s'il existe une chaîne entre u et v , pour toute paire de sommets $u, v \in V(G)$.
- Une composante connexe d'un graphe G est un sous-graphe connexe maximal (par inclusion).



graphe connexe

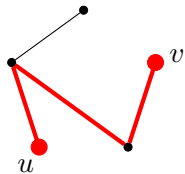


graphe non connexe

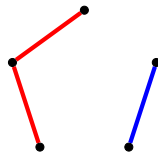
Connexité

Définition

- Un graphe G est connexe s'il existe une chaîne entre u et v , pour toute paire de sommets $u, v \in V(G)$.
- Une composante connexe d'un graphe G est un sous-graphe connexe maximal (par inclusion).



graphe connexe



graphe non connexe

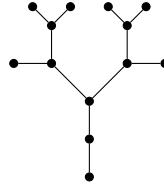
Relation d'équivalence

Remarque

- On peut aussi définir les composantes connexes en utilisant les relations d'équivalence.
- Soit $G = (V, E)$ un graphe et mettons $u \sim v$ si et seulement si il existe une chaîne entre u et v .
- On peut montrer que \sim est une relation d'équivalence.
- Les classes d'équivalence induisent les composantes connexes de G .

Définition

Un arbre est un graphe connexe et acyclique.

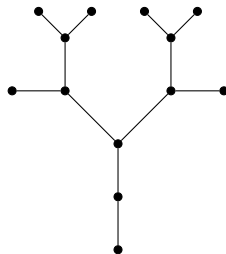


Plusieurs façons de définir un arbre...

Théorème (caractérisation des arbres)

Les énoncés suivants sont équivalents.

1. G est un arbre
2. Pour toute paire de sommets $u, v \in V(G)$, il existe une chaîne élémentaire unique entre u et v
3. G est connexe, et si l'on supprime n'importe quelle arête, le graphe devient non connexe
4. G est acyclique, et si l'on rajoute une nouvelle arête à G , le nouveau graphe contiendra un cycle
5. G est connexe et $|V(G)| = |E(G)| + 1$

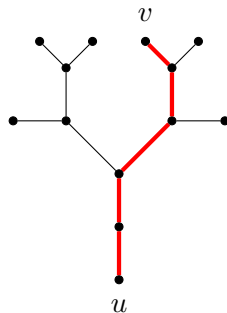


Plusieurs façons de définir un arbre...

Théorème (caractérisation des arbres)

Les énoncés suivants sont équivalents.

1. G est un arbre
2. Pour toute paire de sommets $u, v \in V(G)$, il existe une chaîne élémentaire unique entre u et v
3. G est connexe, et si l'on supprime n'importe quelle arête, le graphe devient non connexe
4. G est acyclique, et si l'on rajoute une nouvelle arête à G , le nouveau graphe contiendra un cycle
5. G est connexe et $|V(G)| = |E(G)| + 1$

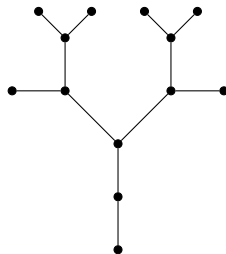


Plusieurs façons de définir un arbre...

Théorème (caractérisation des arbres)

Les énoncés suivants sont équivalents.

1. G est un arbre
2. Pour toute paire de sommets $u, v \in V(G)$, il existe une chaîne élémentaire unique entre u et v
3. G est connexe, et si l'on supprime n'importe quelle arête, le graphe devient non connexe
4. G est acyclique, et si l'on rajoute une nouvelle arête à G , le nouveau graphe contiendra un cycle
5. G est connexe et $|V(G)| = |E(G)| + 1$

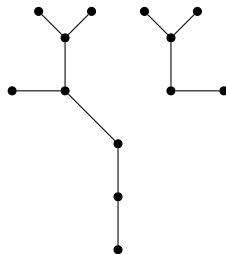


Plusieurs façons de définir un arbre...

Théorème (caractérisation des arbres)

Les énoncés suivants sont équivalents.

1. G est un arbre
2. Pour toute paire de sommets $u, v \in V(G)$, il existe une chaîne élémentaire unique entre u et v
3. G est connexe, et si l'on supprime n'importe quelle arête, le graphe devient non connexe
4. G est acyclique, et si l'on rajoute une nouvelle arête à G , le nouveau graphe contiendra un cycle
5. G est connexe et $|V(G)| = |E(G)| + 1$

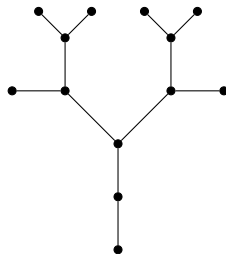


Plusieurs façons de définir un arbre...

Théorème (caractérisation des arbres)

Les énoncés suivants sont équivalents.

1. G est un arbre
2. Pour toute paire de sommets $u, v \in V(G)$, il existe une chaîne élémentaire unique entre u et v
3. G est connexe, et si l'on supprime n'importe quelle arête, le graphe devient non connexe
4. G est acyclique, et si l'on rajoute une nouvelle arête à G , le nouveau graphe contiendra un cycle
5. G est connexe et $|V(G)| = |E(G)| + 1$

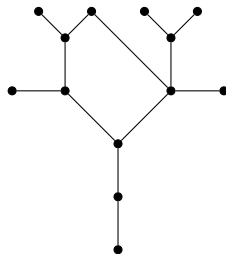


Plusieurs façons de définir un arbre...

Théorème (caractérisation des arbres)

Les énoncés suivants sont équivalents.

1. G est un arbre
2. Pour toute paire de sommets $u, v \in V(G)$, il existe une chaîne élémentaire unique entre u et v
3. G est connexe, et si l'on supprime n'importe quelle arête, le graphe devient non connexe
4. G est acyclique, et si l'on rajoute une nouvelle arête à G , le nouveau graphe contiendra un cycle
5. G est connexe et $|V(G)| = |E(G)| + 1$

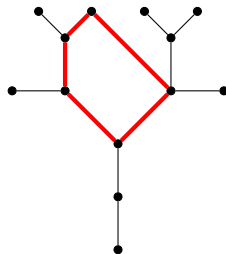


Plusieurs façons de définir un arbre...

Théorème (caractérisation des arbres)

Les énoncés suivants sont équivalents.

1. G est un arbre
2. Pour toute paire de sommets $u, v \in V(G)$, il existe une chaîne élémentaire unique entre u et v
3. G est connexe, et si l'on supprime n'importe quelle arête, le graphe devient non connexe
4. G est acyclique, et si l'on rajoute une nouvelle arête à G , le nouveau graphe contiendra un cycle
5. G est connexe et $|V(G)| = |E(G)| + 1$



Faire pousser un arbre

Lemme

Tout arbre avec au moins deux sommets contient au moins deux feuilles.

Lemme

G est un arbre avec une feuille v ssi $G - v$ est un arbre.

•

Faire pousser un arbre

Lemme

Tout arbre avec au moins deux sommets contient au moins deux feuilles.

Lemme

G est un arbre avec une feuille v ssi $G - v$ est un arbre.



Faire pousser un arbre

Lemme

Tout arbre avec au moins deux sommets contient au moins deux feuilles.

Lemme

G est un arbre avec une feuille v ssi $G - v$ est un arbre.



Faire pousser un arbre

Lemme

Tout arbre avec au moins deux sommets contient au moins deux feuilles.

Lemme

G est un arbre avec une feuille v ssi $G - v$ est un arbre.



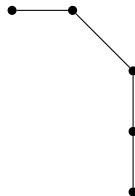
Faire pousser un arbre

Lemme

Tout arbre avec au moins deux sommets contient au moins deux feuilles.

Lemme

G est un arbre avec une feuille v ssi $G - v$ est un arbre.



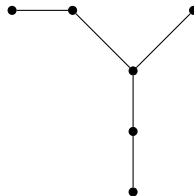
Faire pousser un arbre

Lemme

Tout arbre avec au moins deux sommets contient au moins deux feuilles.

Lemme

G est un arbre avec une feuille v ssi $G - v$ est un arbre.



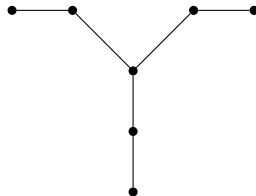
Faire pousser un arbre

Lemme

Tout arbre avec au moins deux sommets contient au moins deux feuilles.

Lemme

G est un arbre avec une feuille v ssi $G - v$ est un arbre.



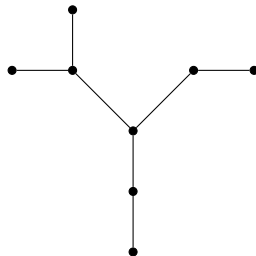
Faire pousser un arbre

Lemme

Tout arbre avec au moins deux sommets contient au moins deux feuilles.

Lemme

G est un arbre avec une feuille v ssi $G - v$ est un arbre.



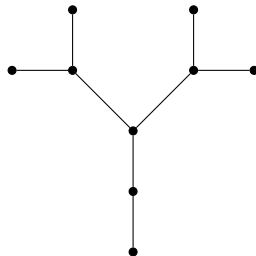
Faire pousser un arbre

Lemme

Tout arbre avec au moins deux sommets contient au moins deux feuilles.

Lemme

G est un arbre avec une feuille v ssi $G - v$ est un arbre.



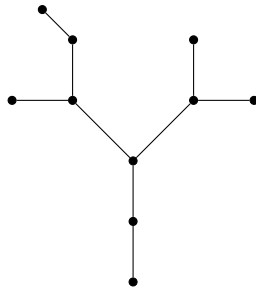
Faire pousser un arbre

Lemme

Tout arbre avec au moins deux sommets contient au moins deux feuilles.

Lemme

G est un arbre avec une feuille v ssi $G - v$ est un arbre.



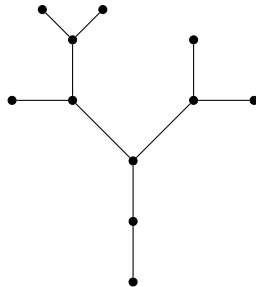
Faire pousser un arbre

Lemme

Tout arbre avec au moins deux sommets contient au moins deux feuilles.

Lemme

G est un arbre avec une feuille v ssi $G - v$ est un arbre.



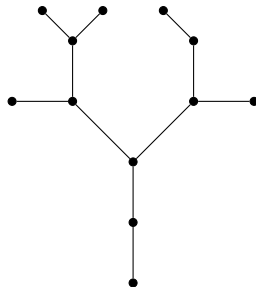
Faire pousser un arbre

Lemme

Tout arbre avec au moins deux sommets contient au moins deux feuilles.

Lemme

G est un arbre avec une feuille v ssi $G - v$ est un arbre.



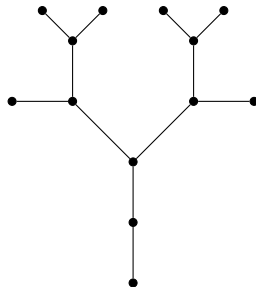
Faire pousser un arbre

Lemme

Tout arbre avec au moins deux sommets
contient au moins deux feuilles.

Lemme

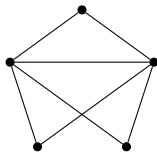
G est un arbre avec une feuille v ssi $G - v$ est un arbre.



Arbre couvrant

Définition

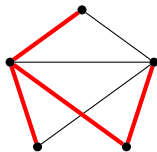
Soit G un graphe. Un arbre couvrant de G est un sous-graphe couvrant de G qui est un arbre.



Arbre couvrant

Définition

Soit G un graphe. Un arbre couvrant de G est un sous-graphe couvrant de G qui est un arbre.



Démonstration

Proposition

Un graphe est connexe \iff il contient un arbre couvrant.

Démonstration

- Soit G un graphe connexe.
- Retirons de G , tant qu'il est possible, une arête qui ne coupe pas le graphe (le graphe reste connexe).
- On obtient un sous-graphe partiel T qui est connexe par la condition sur les arêtes, et il n'a pas de cycles puisque s'il y aurait un cycle, on pourrait enlever une arête du cycle sans couper le graphe.
- T est donc un arbre.
- Le converse est évident.

Comment décider si un graphe est connexe ?

Une question fondamentale que l'on puisse se poser à propos d'un graphe :

Question

Le graphe G est-il connexe ?

Sinon, quelles sont les composantes connexes de G ?

- Pour des petits graphes, on peut le faire par inspection.
- Pour des grands graphes, il faut un algorithme !

Algorithme de marquage

Entrées : graphe $G = (V, E)$ et sommet $s \in V$

début

marquer(s) ;

tant que $\exists uv \in E$ où u est marqué et v non marqué **faire**

└ marquer(v)

Comment améliorer cet algorithme ?

- Si, à un moment donné, tous les voisins d'un sommet marqué sont marqués, nous n'avons plus besoin de considérer ce sommet (puisque seul un sommet non marqué peut devenir marqué, et non l'inverse).
- Il suffit de ne considérer que les sommets marqués “sur le bord”.
- En spécialisant l'ordre dans lequel nous recherchons les arêtes à partir des sommets marqués, nous obtenons des parcours spéciaux, adaptés à différents degrés pour de différents types de problèmes.
- Le parcours en largeur prend systématiquement les arêtes les plus proches à la racine.

Parcours en largeur (BFS¹)

Entrées : graphe $G = (V, E)$ et sommet $s \in V$

début

créer file(Q) ;

marquer(s) ;

enfiler(Q, s) ;

tant que $Q \neq \emptyset$ **faire**

$u \leftarrow$ défiler(Q) ;

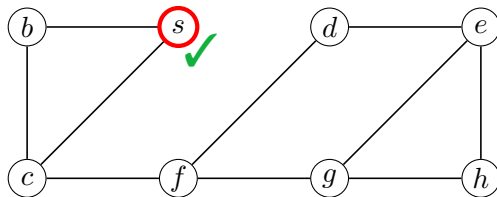
pour tous les $uv \in E$ **faire**

si v non marqué **alors**

 marquer(v) ;

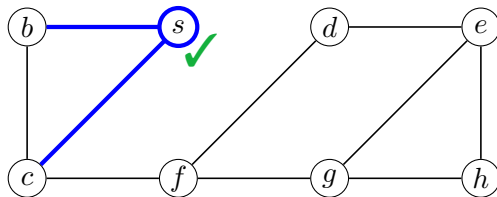
 enfiler(Q, v)

Illustration du parcours en largeur



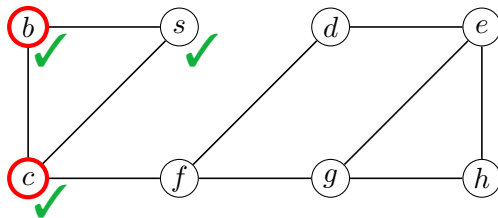
$$Q = [s]$$

Illustration du parcours en largeur



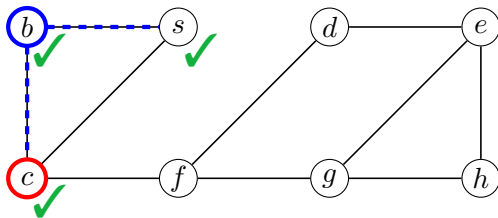
$$Q = \square$$
$$u = s$$

Illustration du parcours en largeur



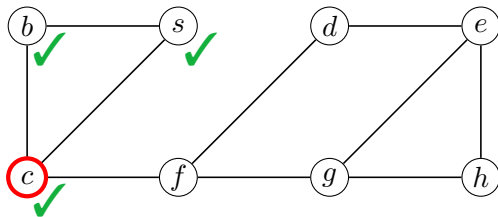
$$Q = [b, c]$$

Illustration du parcours en largeur



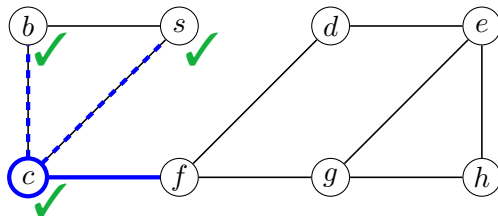
$$Q = [c]$$
$$u = b$$

Illustration du parcours en largeur



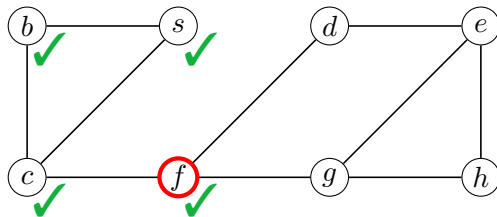
$$Q = [c]$$

Illustration du parcours en largeur



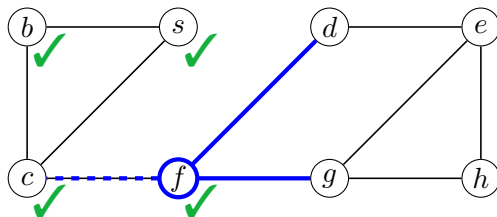
$$Q = \square$$
$$u = c$$

Illustration du parcours en largeur



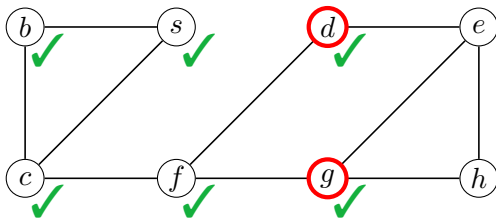
$$Q = [f]$$

Illustration du parcours en largeur



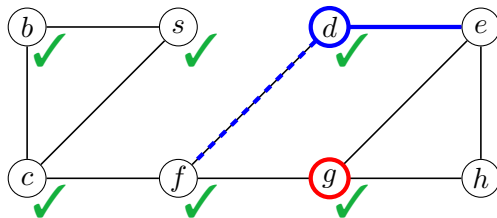
$$Q = \square$$
$$u = f$$

Illustration du parcours en largeur



$$Q = [d, g]$$

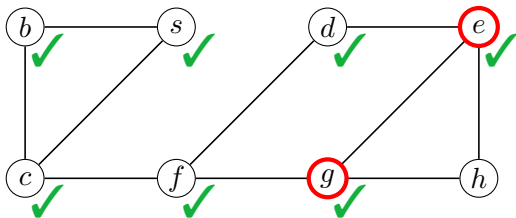
Illustration du parcours en largeur



$$Q = [g]$$

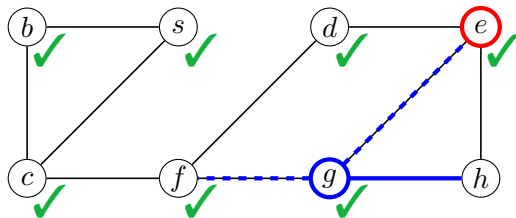
$$u = d$$

Illustration du parcours en largeur



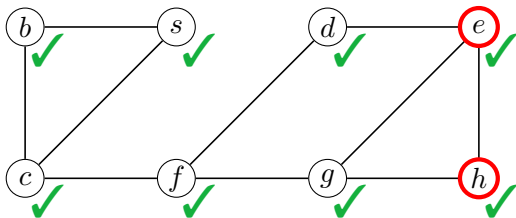
$$Q = [g, e]$$

Illustration du parcours en largeur



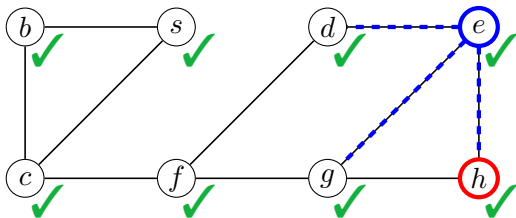
$$Q = [e]$$
$$u = g$$

Illustration du parcours en largeur



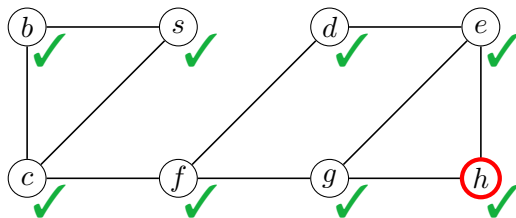
$$Q = [e, h]$$

Illustration du parcours en largeur



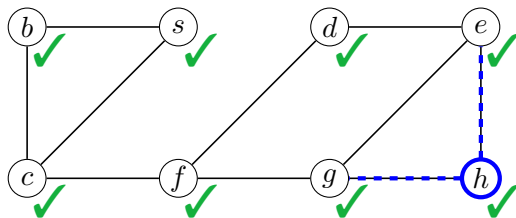
$$Q = [h]$$
$$u = e$$

Illustration du parcours en largeur



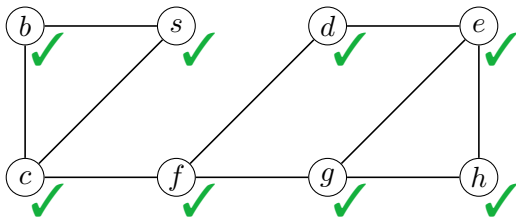
$$Q = [h]$$

Illustration du parcours en largeur



$$Q = \square$$
$$u = h$$

Illustration du parcours en largeur



$$Q = \square$$

Parcours en largeur (version avec distances)

Entrées : graphe $G = (V, E)$ et sommet $s \in V$

début

pour tous les $u \in V \setminus \{s\}$ **faire**

$d(u) \leftarrow \infty$

$\text{dist}(s) \leftarrow 0$;

$Q \leftarrow [s]$;

tant que $Q \neq \emptyset$ **faire**

$u \leftarrow \text{défiler}(Q)$;

pour tous les $uv \in E$ **faire**

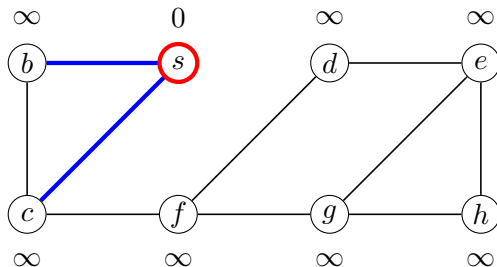
si $d(v) = \infty$ **alors**

$d(v) = d(u) + 1$;

$\text{enfiler}(Q, v)$;

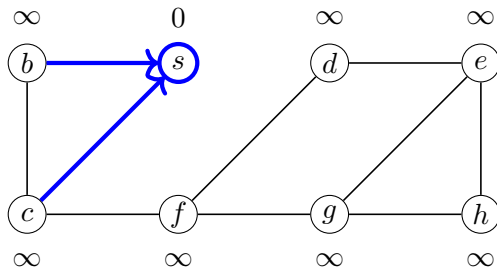
$\text{parent}(v) \leftarrow u$

Illustration du parcours en largeur (version avec distances)



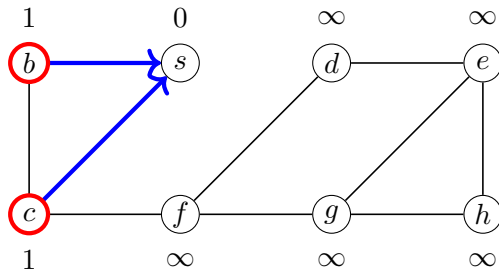
$$Q = [s]$$

Illustration du parcours en largeur (version avec distances)



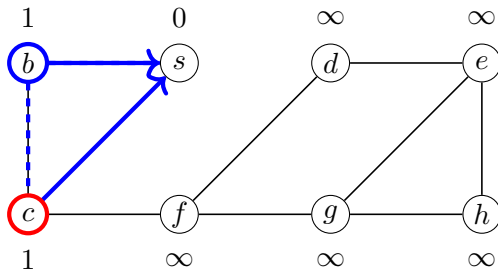
$$Q = []$$
$$u = s$$

Illustration du parcours en largeur (version avec distances)



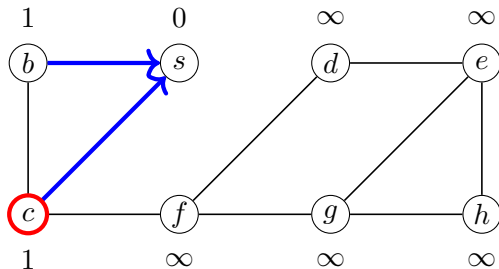
$$Q = [b, c]$$

Illustration du parcours en largeur (version avec distances)



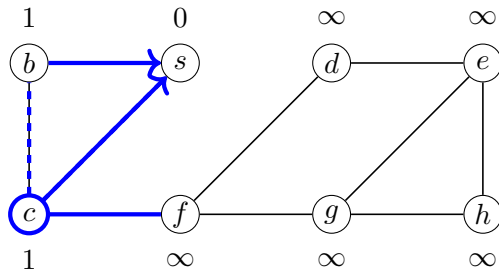
$$Q = [c]$$
$$u = b$$

Illustration du parcours en largeur (version avec distances)



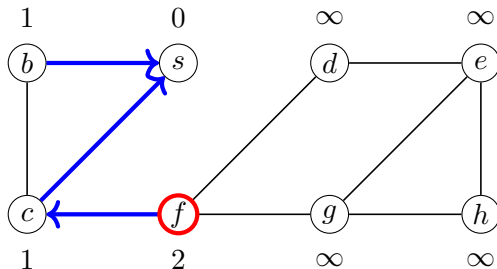
$$Q = [c]$$

Illustration du parcours en largeur (version avec distances)



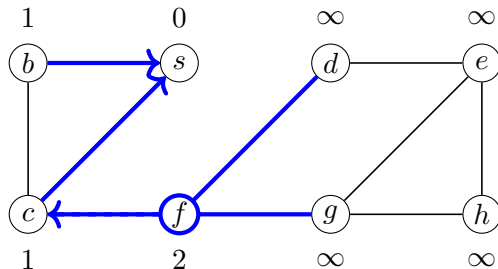
$$Q = \square$$
$$u = c$$

Illustration du parcours en largeur (version avec distances)



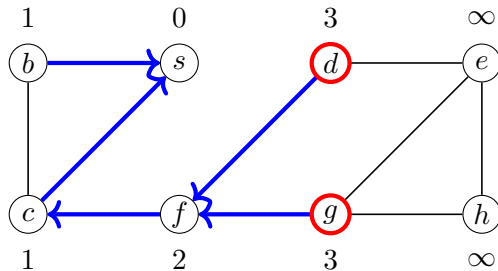
$$Q = [f]$$

Illustration du parcours en largeur (version avec distances)



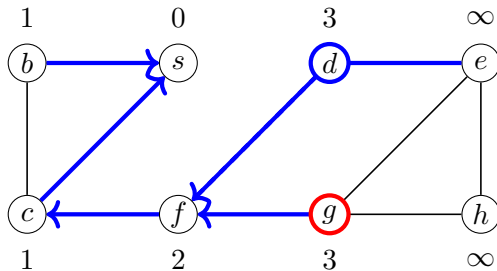
$$Q = \square$$
$$u = f$$

Illustration du parcours en largeur (version avec distances)



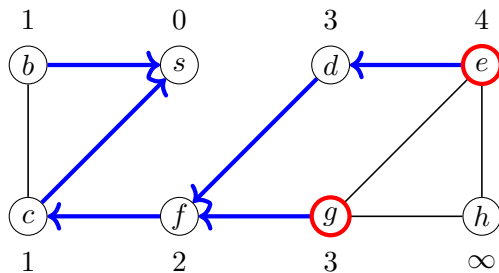
$$Q = [d, g]$$

Illustration du parcours en largeur (version avec distances)



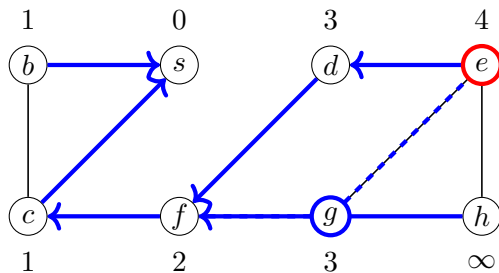
$$Q = [g]$$
$$u = d$$

Illustration du parcours en largeur (version avec distances)



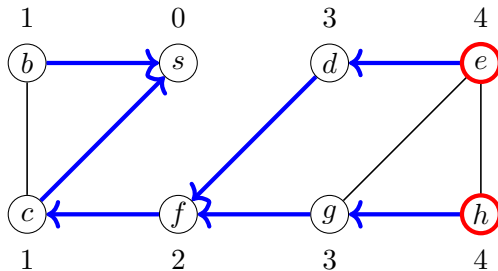
$$Q = [g, e]$$

Illustration du parcours en largeur (version avec distances)



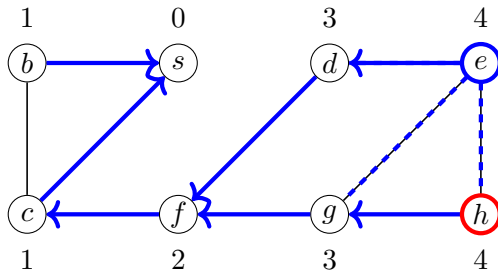
$$Q = [e]$$
$$u = g$$

Illustration du parcours en largeur (version avec distances)



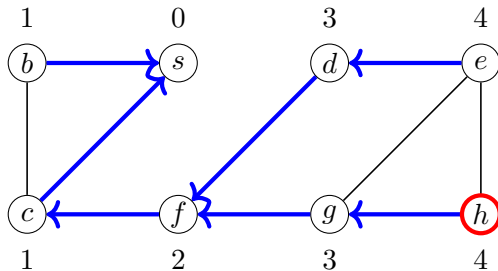
$$Q = [e, h]$$

Illustration du parcours en largeur (version avec distances)



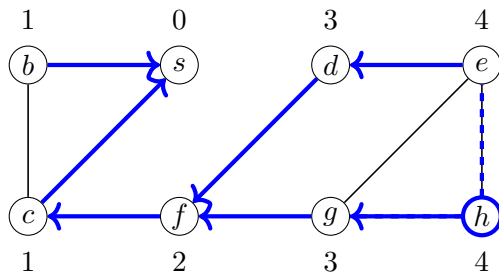
$$Q = [h]$$
$$u = e$$

Illustration du parcours en largeur (version avec distances)



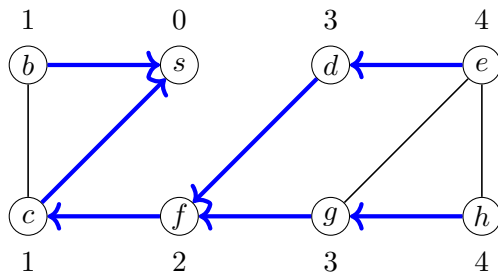
$$Q = [h]$$

Illustration du parcours en largeur (version avec distances)



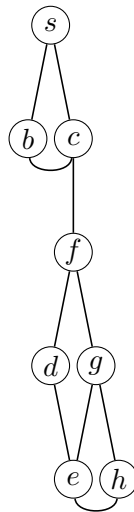
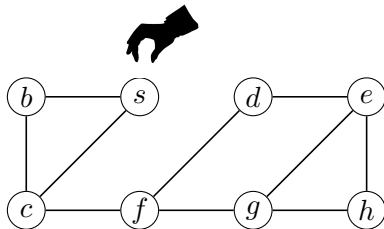
$$Q = \square$$
$$u = h$$

Illustration du parcours en largeur (version avec distances)



$$Q = \square$$

Modèle physique



Complexité du BFS

- Supposons que $G = (V, E)$ est représenté par une liste d'adjacence.
- BFS considère chaque arête deux fois.
- Chaque sommet est enfilé et défilé une fois
- Donc la complexité de BFS est de $O(n + m)$, où $n = |V|$ et $m = |E|$.
- La complexité est donc linéaire par rapport à la taille de la représentation.
- Quelle est la complexité du BFS si G est représenté par une matrice d'adjacence ?

Correction du BFS

Lemme 1

Après terminaison du BFS, $d(v) \geq \text{dist}(s, v)$ pour tout sommet $v \in V$.

Lemme 2

Si au cours du BFS $Q = [v_1, v_2, \dots, v_r]$, alors $d(v_r) \leq d(v_1) + 1$ et $d(v_i) \leq d(v_{i+1})$ pour $i = 1, 2, \dots, r - 1$.

Corollaire 1

Si le sommet v_i est enfilé dans Q avant v_j , alors $d(v_i) \leq d(v_j)$.

Démonstration du Lemme 2

Démonstration (1/2)

- On montre que la propriété est invariante par les opérations sur la file.
- Initialement, $Q = [s]$: la propriété est vraie.

Cas 1 : le sommet v_1 est défilé de Q .

- Si Q devient vide, alors la propriété devient trivialement vraie.
- Sinon, par l'hypothèse de récurrence, $d(v_1) \leq d(v_2)$ et $d(v_r) \leq d(v_1) + 1$, et donc $d(v_r) \leq d(v_2) + 1$.
- Les inégalités entre les autres éléments de Q restent inchangées.

Démonstration du Lemme 2

Démonstration (1/2)

Cas 2 : un nouveau sommet v_{r+1} est enfilé dans Q .

- Soit u le dernier sommet défilé de Q , dont les voisins (parmi lesquels se trouve v_{r+1}) sont en train d'être examinés.
- Cas trivial : le défilage de u a vidé Q . Dans ce cas, tous les sommets de Q , y compris v_{r+1} , ont la même valeur de d (égale à $d(u) + 1$).
- Sinon, par l'hypothèse de récurrence, $d(u) \leq d(v_1)$. Donc,
 $d(v_{r+1}) = d(u) + 1 \leq d(v_1) + 1$.
- On a aussi $d(v_r) \leq d(u) + 1 = d(v_{r+1})$ par l'hypothèse de récurrence.

Correction du BFS

Théorème

BFS découvre tous les sommets accessibles à partir de s , et après terminaison, $d(v) = \text{dist}(s, v)$ pour tout $v \in V$.

Démonstration

- Supposons qu'il existe $v \in V$ tel que $d(v) \neq \text{dist}(s, v)$.
- Soit v un tel sommet qui minimise $\text{dist}(s, v)$.
- Grâce au Lemme 1, on a $d(v) > \text{dist}(s, v)$.
- Considérons une chaîne de longueur minimale de s à v , et soit u le prédécesseur immédiat de v sur cette chaîne : $\text{dist}(s, v) = \text{dist}(s, u) + 1$.
- Par la minimalité de v , on a $d(v) > \text{dist}(s, v) = \text{dist}(s, u) + 1 = d(u) + 1$.

Correction du BFS (2/2)

Démonstration (2/2)

- Considérons le moment où u est défilé de Q .

Cas 1 v n'est pas marqué : $d(v) = d(u) + 1$

Cas 2 v a été traité : $d(v) \leq d(u)$

Cas 3 v est dans la file : il est inséré au moment du traitement d'un sommet w , extrait avant u , pour lequel $d(v) = d(w) + 1$. Par le Corollaire 1, on a $d(w) \leq d(u)$, et donc $d(v) \leq d(u) + 1$.

Résumé du BFS

- Algorithme efficace pour explorer un graphe : de complexité $O(n + m)$.
- Parcourt les sommets à partir d'un sommet "source" par des couches :
 - s d'abord
 - ensuite les voisins de s ,
 - ensuite les voisins des voisins de s ,
 - etc. . .
- Bien adapté aux problèmes où l'on cherche un plus court chemin : par exemple, si l'on veut ranger le cube de Rubik en un nombre minimum de coups.

Bonus : implémentation Python qui n'utilise pas les files

```
def bfs (s,Adj):  
    level = {s: 0}  
    parent = {s: None}  
    i = 1  
    frontier = [s]  
    while frontier:  
        next = []  
        for u in frontier:  
            for v in Adj[u]:  
                if v not in level:  
                    level[v] = i  
                    parent[v] = u  
                    next.append(v)  
        frontier = next  
        i += 1
```