Parcours dans les graphes

François Delbot

8 octobre 2019

Algorithme de Warshall

L'algorithme de Warshall calcule la fermeture transitive d'un graphe (orienté ou non).

1. Partant d'un graphe $G = (V, A)$, l'algorithme produit un second graphe $G' = (V, A')$ tel que si il existe un chemin dans G allant d'un sommet u à un sommet v alors $(u, v) \in A'$.
2. Permet de connaitre l'ensemble des relations d'accessibilité au sein du graphe G.
3. Complexité en $O(n^3)$

Algorithme de Warshall

Algorithm 4: Algorithme de Roy-Warshall

Data: $G = (V, A)$ un graphe (orienté ou non)
Result: Un graphe G' représentant la fermeture transitive de G

$A' = A$;
forall $u \in V$ **do**
  forall $v \in N^-(u)$ **do**
    forall $w \in N^+(u)$ **do**
      $A' = A' \cup (v, w)$;
    end
  end
end
return $G' = (V, A')$;

Algorithme de Warshall

Exemple de déroulement de l'algorithme.

Nous allons dérouler l'algorithme sur le graphe suivant :

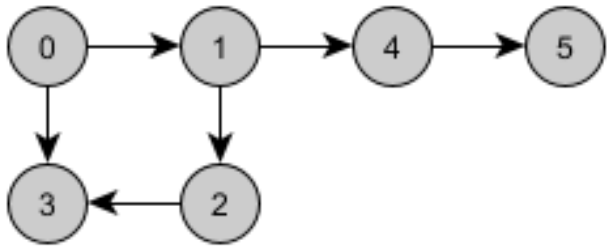


FIGURE – Graphe de départ

Pour cela, nous allons considérer les sommets dans l'ordre suivant $0, 1, 2, 3, 4, 5$.

Algorithme de Warshall

Étape 1. Considérons le sommet 0.

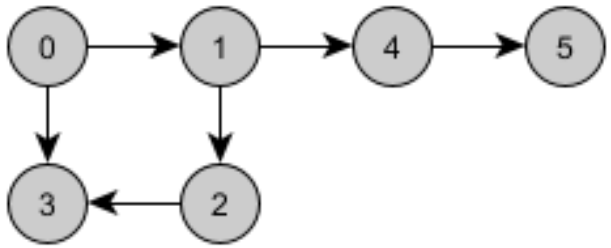


FIGURE – Graphe courant

Le voisinage entrant de 0 est vide. On passe au sommet suivant.

Algorithme de Warshall

Étape 2. Considérons le sommet 1.

$N^-(1) = 0$ et $N^+(1) = \{2, 4\}$. Ainsi, on ajoute les deux arcs $(0, 2)$ et $(0, 4)$.

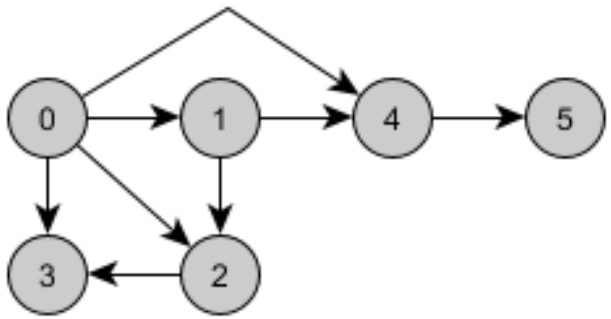


FIGURE – Graphe augmenté des arêtes $(0, 2)$ et $(0, 4)$

Algorithme de Warshall

Étape 3. Considérons le sommet 2.

$N^-(2) = \{0, 1\}$ et $N^+(2) = \{3\}$. L'arête $(0, 3)$ existe déjà. Ainsi, on ajoute uniquement l'arcs $(1, 3)$.

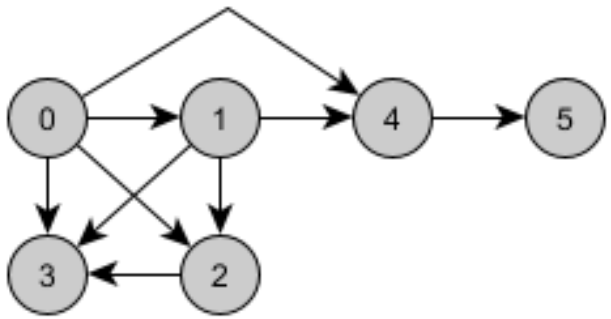


FIGURE – Graphe augmenté de l'arête $(1, 3)$

Algorithme de Warshall

Étape 4. Considérons le sommet 3.

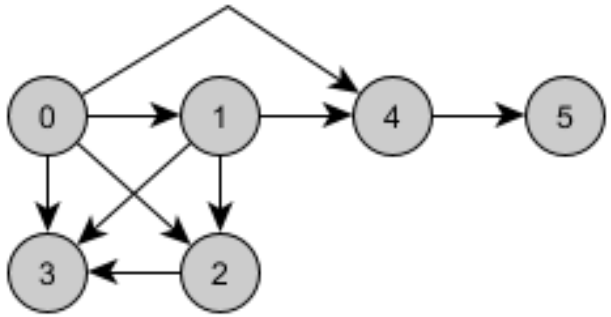


FIGURE – Graphe augmenté de l'arête $(1, 3)$

Le voisinage sortant de 3 est vide. On passe au sommet suivant.

Algorithme de Warshall

Étape 5. Considérons le sommet 4.

$N^-(4) = \{0, 1\}$ et $N^+(4) = \{5\}$. Ainsi, on ajoute les deux arcs $(0, 5)$ et $(1, 5)$.

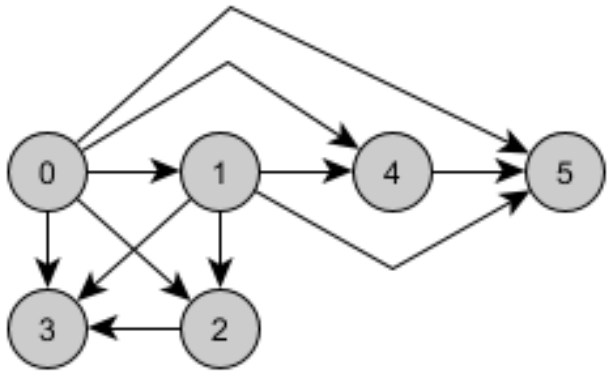


FIGURE – Graphe augmenté des arêtes $(0, 5)$ et $(1, 5)$

Algorithme de Warshall

Étape 6. Considérons le sommet 5.

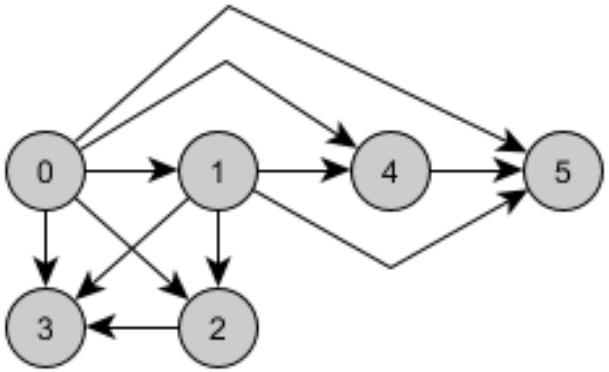
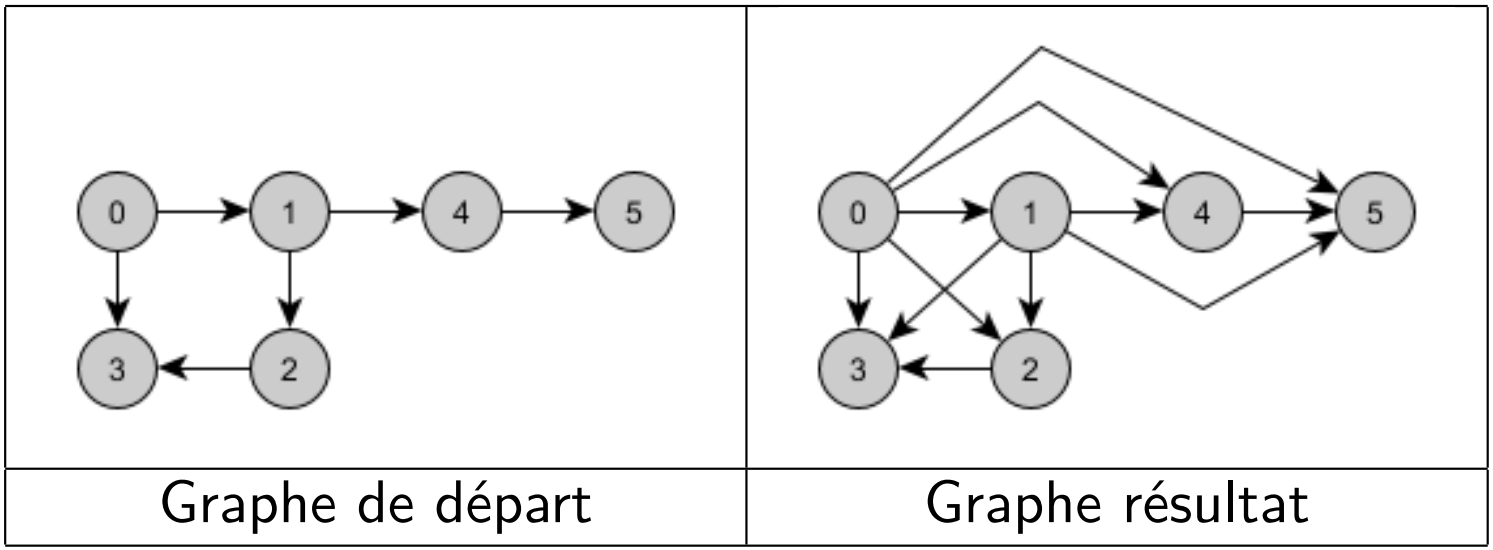


FIGURE – Graphe courant

Le voisinage sortant de 5 est vide. L'algorithme se termine.

Algorithme de Warshall

Résultat et utilisation



Graphe de départ	Graphe résultat

1. Est-ce que le sommet 5 est atteignable depuis le sommet 0 ?
   L'arc $(0, 5)$ existe dans le graphe résultat, donc oui.
2. Est-ce que le sommet 5 est atteignable depuis le sommet 3 ?
   L'arc $(3, 5)$ n'existe pas dans le graphe résultat, donc non.

Parcours en Largeur

Cet algorithme permet de parcourir un graphe depuis un sommet d'origine.

1. Parcourt les sommets du graphe, de manière concentrique. Les sommets situés à une distance 1 de l'origine, puis les sommets à distance 2 etc.
2. Produit un arbre des plus courts chemins (si le graphe est non pondéré).
3. Connaissance de la connexité du graphe.
4. Complexité en $O(n^2)$.

Algorithme du parcours en largeur

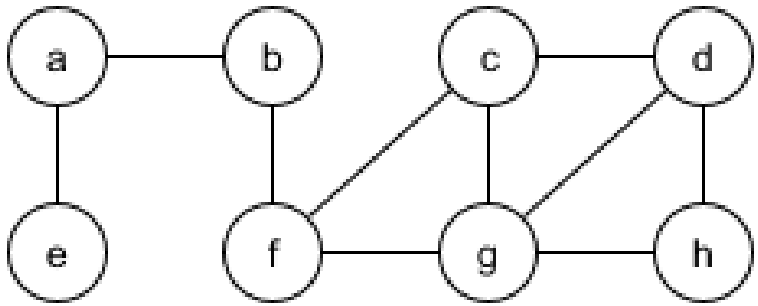
Algorithm 5: Parcours_en_Largeur(G,s)

Data: $G = (V, E)$ un graphe et $s \in V$ un sommet du graphe
Result: Un parcours en largeur issu du sommet s
forall $u \in V - \{s\}$ **do**
    couleur[u]=Blanc;
    distance[u]=∞;
    pere[u]=NIL;
end
couleur[s]=Gris;
distance[s]=0;
pere[s]=NIL;
F=$\{s\}$;
while $F \neq \emptyset$ **do**
    u=Defiler[F];
    for $v \in N(u)$ **do**
        if *couleur[v]=Blanc* **then**
            couleur[v]=Gris;
            distance[v]=distance[u]+1;
            pere[v]=u;
            Enfiler(F,v);
        end
    end
    couleur[u]=Noir;
end

Algorithme du parcours en largeur

Graphe initial

Nous allons appliquer l'algorithme sur le graphe suivant, en partant du sommet b.



Il s'agit d'un exemple classique provenant du livre "Introduction à l'algorithmique" (Cormen, Leiserson, Rivest, Stein) dont je recommande la lecture !

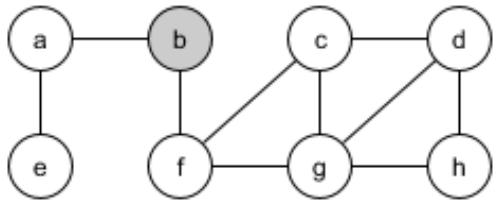
Algorithme du parcours en largeur

Phase d'initialisation

On place la couleur du sommet b à Gris, et celle de tous les autres sommets à Blanc. Le sommet b est à une distance de 0 de lui-même, et tous les autres sommets sont, pour le moment, à une distance infinie. Le sommet b est ajouté à la file F.

Sommet	a	b	c	d	e	f	g	h
Couleur	B	G	B	B	B	B	B	B
Distance	∞	0	∞	∞	∞	∞	∞	∞
Père	-	-	-	-	-	-	-	-

$F = [b]$



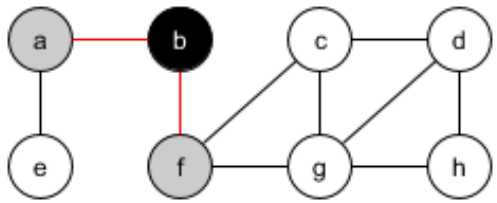
Algorithme du parcours en largeur

Étape 1.

On défile le premier élément de la file F, soit le sommet b. On parcours ses voisins de couleur blanche, soient les sommets f et a (peu importe l'ordre de ces sommets). Leur couleur devient grise, leur distance passe à $0 + 1 = 1$ et ils sont atteignables par le sommet b, qui devient donc leur père. Le sommet b devient noir.

Sommet	a	b	c	d	e	f	g	h
Couleur	G	N	B	B	B	G	B	B
Distance	1	0	∞	∞	∞	1	∞	∞
Père	b	-	-	-	-	b	-	-

$F = [f, a]$



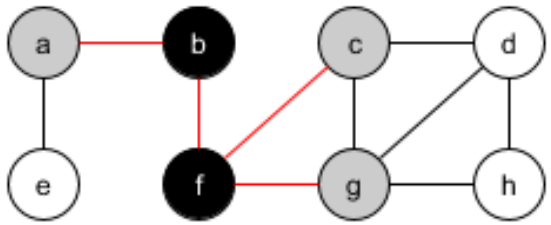
Algorithme du parcours en largeur

Étape 2.

On défile le premier élément de la file F, soit le sommet f. On parcours ses voisins de couleur blanche, soient les sommets c et g (peu importe l'ordre de ces sommets). Leur couleur devient grise, leur distance passe à $1 + 1 = 2$ et ils sont atteignables par le sommet f, qui devient donc leur père. Le sommet f devient noir.

Sommet	a	b	c	d	e	f	g	h
Couleur	G	N	G	B	B	N	G	B
Distance	1	0	2	∞	∞	1	2	∞
Père	b	-	f	-	-	b	f	-

$F = [a, c, g]$



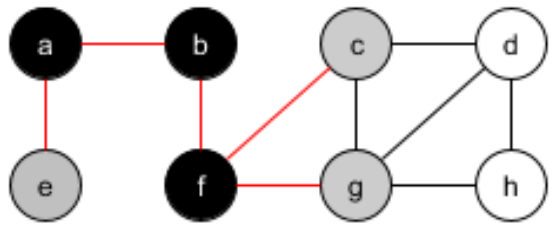
Algorithme du parcours en largeur

Étape 3.

On défile le premier élément de la file F, soit le sommet a. On parcours ses voisins de couleur blanche (il n'y en a qu'un), soit le sommet e. Sa couleur devient grise, sa distance passe à $1 + 1 = 2$ et il est atteignable par le sommet a, qui devient donc son père. Le sommet a devient noir.

Sommet	a	b	c	d	e	f	g	h
Couleur	N	N	G	B	G	N	G	B
Distance	1	0	2	∞	2	1	2	∞
Père	b	-	f	-	a	b	f	-

$F = [c, g, e]$



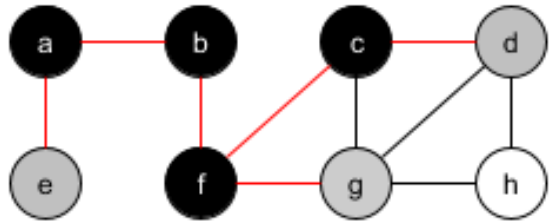
Algorithme du parcours en largeur

Étape 4.

On défile le premier élément de la file F, soit le sommet c. On parcours ses voisins de couleur blanche (il n'y en a qu'un), soit le sommet d. Sa couleur devient grise, sa distance passe à $2 + 1 = 3$ et il est atteignable par le sommet c, qui devient donc son père. Le sommet c devient noir.

Sommet	a	b	c	d	e	f	g	h
Couleur	N	N	N	G	G	N	G	B
Distance	1	0	2	3	2	1	2	∞
Père	b	-	f	c	a	b	f	-

$F = [g, e, d]$



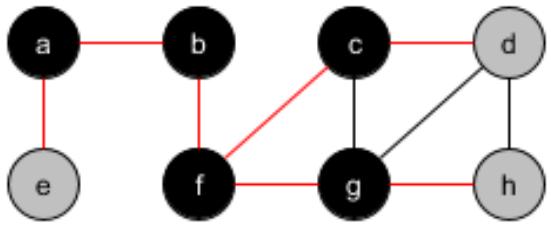
Algorithme du parcours en largeur

Étape 5.

On défile le premier élément de la file F, soit le sommet g. On parcours ses voisins de couleur blanche (il n'y en a qu'un), soit le sommet h. Sa couleur devient grise, sa distance passe à $2 + 1 = 3$ et il est atteignable par le sommet g, qui devient donc son père. Le sommet g devient noir.

Sommet	a	b	c	d	e	f	g	h
Couleur	N	N	N	G	G	N	N	G
Distance	1	0	2	3	2	1	2	3
Père	b	-	f	c	a	b	f	g

$F = [e, d, h]$



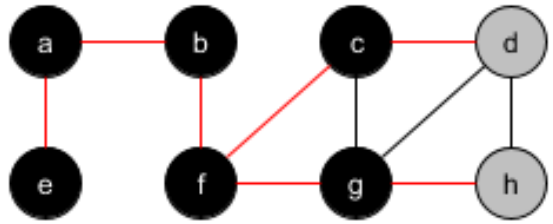
Algorithme du parcours en largeur

Étape 6.

On défile le premier élément de la file F, soit le sommet e. On parcours ses voisins de couleur blanche (il n'y en a aucun). Le sommet e devient noir.

Sommet	a	b	c	d	e	f	g	h
Couleur	N	N	N	G	N	N	N	G
Distance	1	0	2	3	2	1	2	3
Père	b	-	f	c	a	b	f	g

$F = [d, h]$



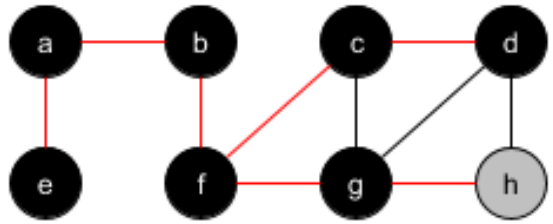
Algorithme du parcours en largeur

Étape 7.

On défile le premier élément de la file F, soit le sommet d. On parcours ses voisins de couleur blanche (il n'y en a aucun). Le sommet d devient noir.

Sommet	a	b	c	d	e	f	g	h
Couleur	N	N	N	N	N	N	N	G
Distance	1	0	2	3	2	1	2	3
Père	b	-	f	c	a	b	f	g

$F = [h]$



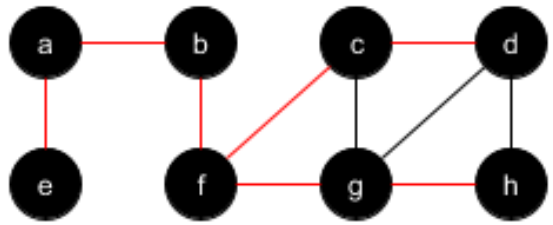
Algorithme du parcours en largeur

Étape 8.

On défile le premier élément de la file F, soit le sommet h. On parcours ses voisins de couleur blanche (il n'y en a aucun). Le sommet h devient noir. La file F est vide, l'algorithme se termine.

Sommet	a	b	c	d	e	f	g	h
Couleur	N	N	N	N	N	N	N	N
Distance	1	0	2	3	2	1	2	3
Père	b	-	f	c	a	b	f	g

$F = []$



Algorithme du parcours en largeur

Résultat

1. Si l'un des sommets se trouve à une distance infinie, cela signifie que le graphe n'est pas connexe.
2. A partir d'un sommet, et en remontant père après père, on obtient un plus court chemin vers l'origine.
3. Par exemple, le père du sommet h est le sommet g. Le père du sommet g est le sommet f. Le père du sommet f est le sommet b. Nous obtenons ainsi une chaine reliant le sommet h au sommet b : h - g - f - b.

Parcours en profondeur

Cet algorithme permet de parcourir un graphe depuis un sommet d'origine.

1. Parcourt les sommets du graphe.
2. Permet la détection de cycles.
3. Connaissance de la connexité du graphe.
4. Complexité en $O(n^2)$.

Algorithme du parcours en profondeur

Algorithm 6: DFS(G)

Data: $G = (V, E)$ un graphe
Result: Une forêt de parcours en profondeur du graphe G.
forall $u \in V$ **do**
  couleur[u]=Blanc;
  pere[u]=NIL;
end
temps = 0;
forall $u \in V$ **do**
  if *couleur[v]=Blanc* **then**
    DFS_Visit(G,u);
  end
end

Algorithm 7: DFS_Visit(G,u)

Data: $G = (V, E)$ un graphe
Result: Une forêt de parcours en profondeur du graphe G.
couleur[u]=Gris;
debut[u]=temps;
temps=temps+1;
forall $v \in N(u)$ **do**
  if *couleur[v]=Blanc* **then**
    pere[v]=u;
    DFS_Visit(G,v);
  end
end
couleur[u]=Noir;
fin[u]=temps;
temps=temps+1;

Algorithme du parcours en profondeur

Évaluation du temps

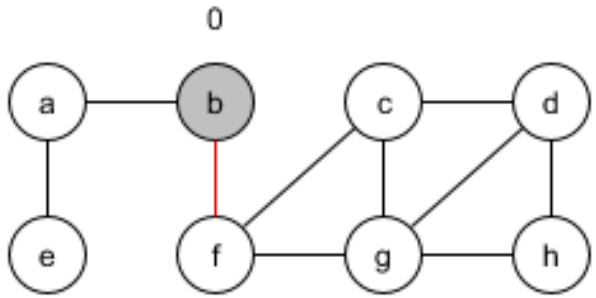
- Temps (discret) de la coloration d'un sommet en gris (moment de la première visite de ce sommet).
- Temps (discret) de la coloration d'un sommet en noir d'un sommet (fin de traitement).
- Une variable nommée *temps*, initialisée à 0 qui va augmenter de 1 lors de chaque étape.

Algorithme du parcours en profondeur

Étape 1.

Le sommet b devient gris. Le premier passage (debut) se fait lorsque temps vaut 0. S'agissant du premier sommet, il n'a pas de père. On augmente le temps de 1. On considère ensuite les voisins de b dans l'ordre suivant : $\{f, a\}$, par exemple. La couleur du sommet f est blanche, donc le père de f devient b et on réalise un appel récursif sur le sommet f.

Sommet	a	b	c	d	e	f	g	h
Couleur	B	G	B	B	B	B	B	B
Père	-	-	-	-	-	b	-	-
Début		0						
Fin								

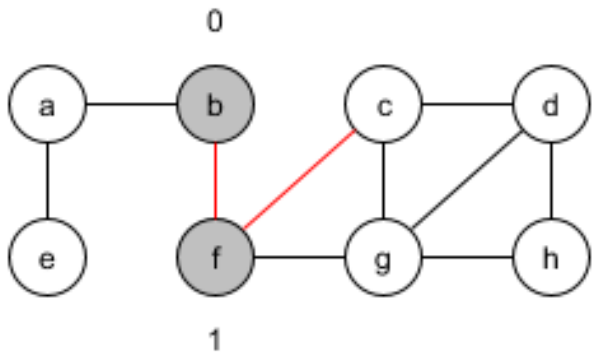


Algorithme du parcours en profondeur

Étape 2.

Le sommet f devient gris. Le premier passage (debut) se fait lorsque temps vaut 1. On augmente le temps de 1. On considère les voisins de f dans l'ordre suivant : $\{b, c, g\}$, par exemple. La couleur du sommet b n'est pas blanche, donc on ne réalise pas d'appel récursif sur ce sommet. On considère ensuite le sommet c. Sa couleur est blanche donc le père de c devient f et on réalise un appel récursif sur c.

Sommet	a	b	c	d	e	f	g	h
Couleur	B	G	B	B	B	G	B	B
Père	-	-	f	-	-	b	-	-
Debut		0				1		
Fin								

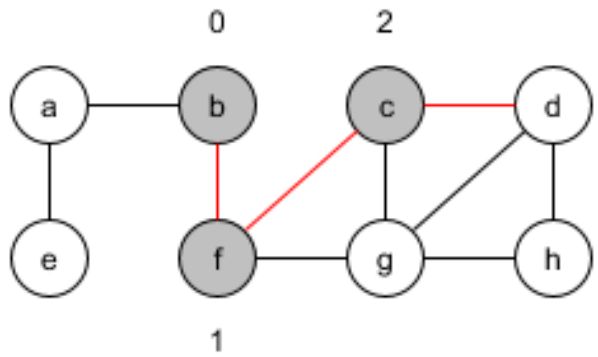


Algorithme du parcours en profondeur

Étape 3.

Le sommet c devient gris. Le premier passage (debut) se fait lorsque temps vaut 2. On augmente le temps de 1. On considère les voisins de c dans l'ordre suivant : $\{d, f, g\}$, par exemple. On considère le sommet d. Sa couleur est blanche donc le père de d devient c et on réalise un appel récursif sur d.

Sommet	a	b	c	d	e	f	g	h
Couleur	B	G	G	B	B	G	B	B
Père	-	-	f	c	-	b	-	-
Debut		0	2			1		
Fin								

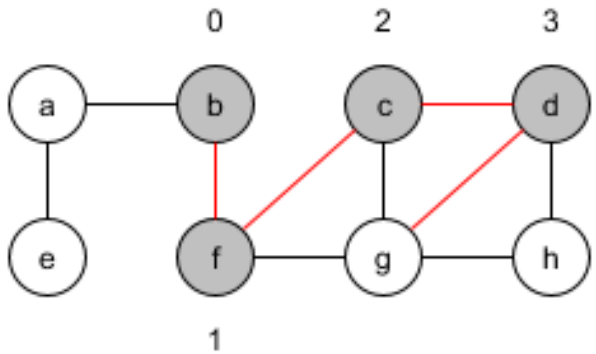


Algorithme du parcours en profondeur

Étape 4.

Le sommet d devient gris. Le premier passage (debut) se fait lorsque temps vaut 3. On augmente le temps de 1. On considère les voisins de d dans l'ordre suivant : $\{c, g, h\}$, par exemple. On considère le sommet c. Sa couleur n'est pas blanche donc on ne réalise pas d'appel récursif sur ce sommet. On considère ensuite le sommet g. Sa couleur est blanche donc le père de g devient d et on réalise un appel récursif sur g.

Sommet	a	b	c	d	e	f	g	h
Couleur	B	G	G	G	B	G	B	B
Père	-	-	f	c	-	b	d	-
Debut		0	2	3		1		
Fin								

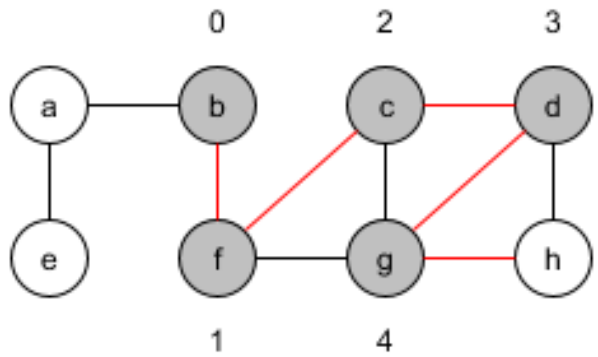


Algorithme du parcours en profondeur

Étape 5.

Le sommet g devient gris. Le premier passage (debut) se fait lorsque temps vaut 4. On augmente le temps de 1. On considère les voisins de g dans l'ordre suivant : $\{c, d, f, h\}$, par exemple. On considère le sommet c,d et f. leur couleur n'est pas blanche donc on ne réalise pas d'appel récursif sur ces sommets.

Sommet	a	b	c	d	e	f	g	h
Couleur	B	G	G	G	B	G	G	B
Père	-	-	f	c	-	b	d	g
Debut		0	2	3		1	4	
Fin								

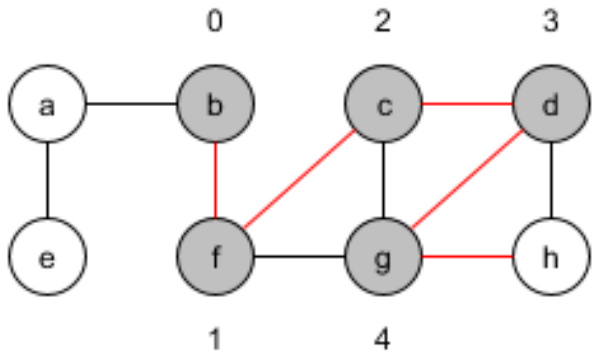


Algorithme du parcours en profondeur

Étape 5.

On considère ensuite le sommet h. Sa couleur est blanche donc le père de h devient g et on réalise un appel récursif sur h.

Sommet	a	b	c	d	e	f	g	h
Couleur	B	G	G	G	B	G	G	B
Père	-	-	f	c	-	b	d	g
Debut		0	2	3		1	4	
Fin								

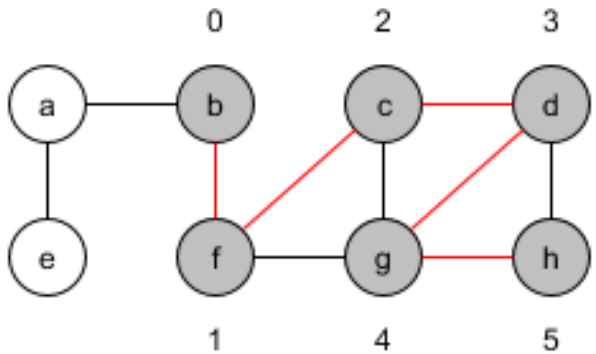


Algorithme du parcours en profondeur

Étape 6.

Le sommet h devient gris. Le premier passage (debut) se fait lorsque temps vaut 5. On augmente le temps de 1. On considère les voisins de h dans l'ordre suivant : $\{d, g\}$. La couleur des sommets d et g n'est pas blanche donc on ne réalise pas d'appel récursif sur ces sommets. La phase d'appels récursifs sur le voisinage du sommet h est terminée.

Sommet	a	b	c	d	e	f	g	h
Couleur	B	G	G	G	B	G	G	G
Père	-	-	f	c	-	b	d	g
Debut		0	2	3		1	4	5
Fin								

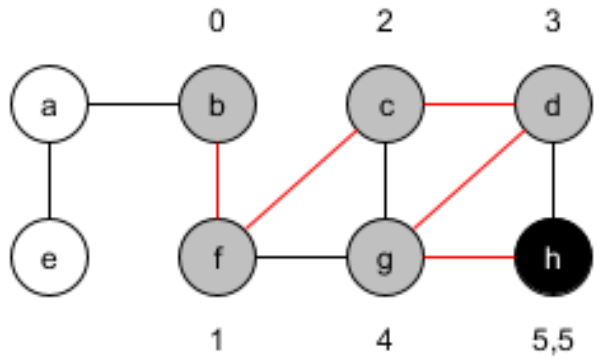


Algorithme du parcours en profondeur

Étape 7.

La phase d'appels récursifs sur le voisinage du sommet h est terminée. Le sommet h devient noir. Le moment de fin de traitement vaut 5. On augmente le temps de 1. L'appel récursif se termine. On retourne sur le traitement des voisins du sommet g.

Sommet	a	b	c	d	e	f	g	h
Couleur	B	G	G	G	B	G	G	N
Père	-	-	f	c	-	b	d	g
Debut		0	2	3		1	4	5
Fin								5

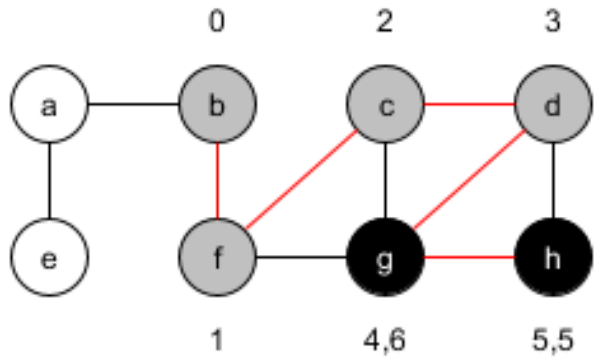


Algorithme du parcours en profondeur

Étape 8.

La phase d'appels récursifs sur le voisinage du sommet g est terminée. Le sommet g devient noir. Le moment de fin de traitement vaut 6. On augmente le temps de 1. L'appel récursif se termine. On retourne sur le traitement des voisins du sommet d.

Sommet	a	b	c	d	e	f	g	h
Couleur	B	G	G	G	B	G	N	N
Père	-	-	f	c	-	b	d	g
Debut		0	2	3		1	4	5
Fin							6	5

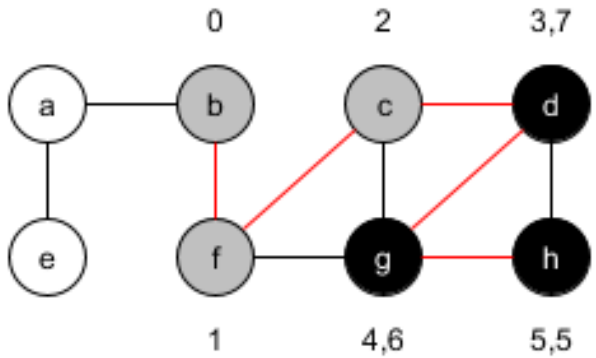


Algorithme du parcours en profondeur

Étape 9.

Le voisinage du sommet d contient les sommets $\{c, g, h\}$. Il reste à traiter le cas du sommet h. La couleur du sommet h n'est pas blanche, donc on ne réalise pas d'appel récursif. La phase d'appels récursifs sur le voisinage du sommet d est terminée. Le moment de fin de traitement vaut 7. On augmente le temps de 1. L'appel récursif se termine. On retourne sur le traitement des voisins du sommet c.

Sommet	a	b	c	d	e	f	g	h
Couleur	B	G	G	G	B	G	N	N
Père	-	-	f	c	-	b	d	g
Debut		0	2	3		1	4	5
Fin				7			6	5

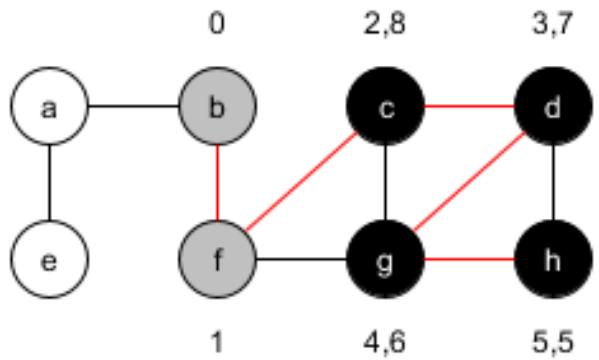


Algorithme du parcours en profondeur

Étape 10.

Le voisinage du sommet c contient les sommets $\{d, f, g\}$. Il reste à traiter le cas des sommets f et g. La couleur de ces deux sommets n'est pas blanche, donc on ne réalise pas d'appel récursif. La phase d'appels récursifs sur le voisinage du sommet c est terminée. Le sommet c devient noir. Le temps passe à 9. L'appel récursif se termine. On retourne sur le traitement des voisins du sommet f.

Sommet	a	b	c	d	e	f	g	h
Couleur	B	G	N	G	B	G	N	N
Père	-	-	f	c	-	b	d	g
Debut		0	2	3		1	4	5
Fin			8	7			6	5

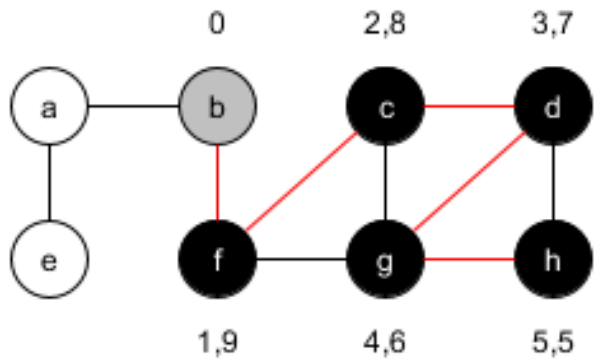


Algorithme du parcours en profondeur

Étape 11.

Le voisinage du sommet f contient les sommets $\{b, c, g\}$. Il reste à traiter le cas du sommet g. La couleur de ce sommes n'est pas blanche, donc on ne réalise pas d'appel récursif. La phase d'appels récursifs sur le voisinage du sommet f est terminée. Le sommet f devient noir. Le temps passe à 10. L'appel récursif se termine. On retourne sur le traitement des voisins du sommet b.

Sommet	a	b	c	d	e	f	g	h
Couleur	B	G	N	G	B	N	N	N
Père	-	-	f	c	-	b	d	g
Debut		0	2	3		1	4	5
Fin			8	7		9	6	5

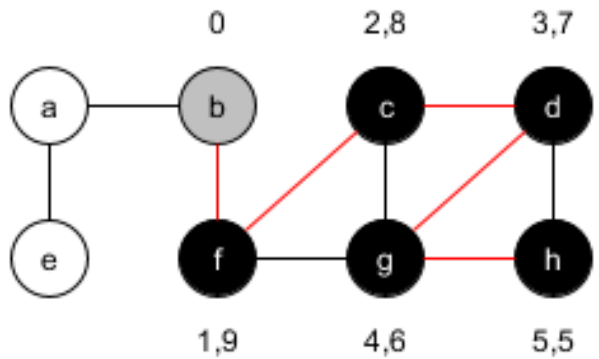


Algorithme du parcours en profondeur

Étape 12.

Le voisinage du sommet b contient les sommets $\{a, f\}$. Il reste à traiter le cas du sommet a. La couleur de ce sommet est blanche, donc on réalise un appel récursif sur le sommet a.

Sommet	a	b	c	d	e	f	g	h
Couleur	B	G	N	G	B	N	N	N
Père	b	-	f	c	-	b	d	g
Debut		0	2	3		1	4	5
Fin			8	7		9	6	5

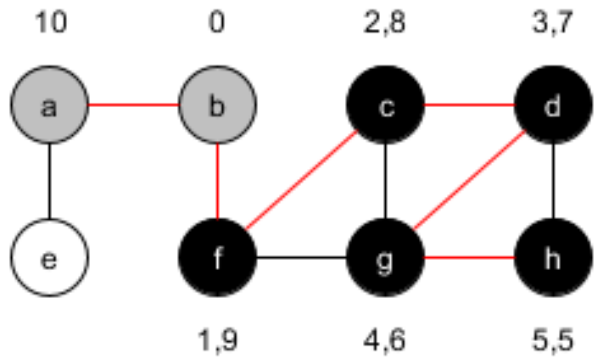


Algorithme du parcours en profondeur

Étape 13.

Le voisinage du sommet a contient les sommets $\{b, e\}$. La couleur du sommet b n'est pas blanche, donc on ne réalise pas d'appel récursif. La couleur du sommet e est blanche, donc le père de e devient a et on réalise un appel récursif sur le sommet e.

Sommet	a	b	c	d	e	f	g	h
Couleur	G	G	N	G	B	N	N	N
Père	b	-	f	c	a	b	d	g
Debut	10	0	2	3		1	4	5
Fin			8	7		9	6	5

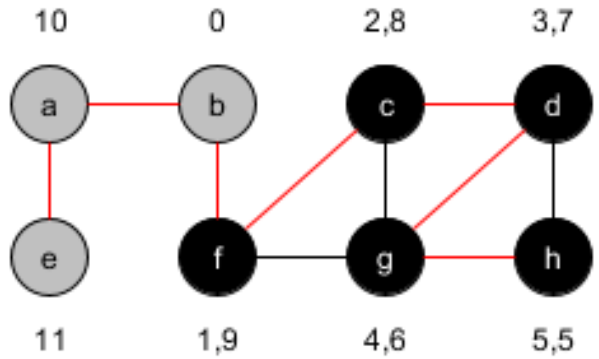


Algorithme du parcours en profondeur

Étape 14.

Le sommet e devient gris. Le premier passage (debut) se fait lorsque temps vaut 11. On augmente le temps de 1. On considère le voisin de e : $\{a\}$. La couleur du sommet a n'est pas blanche donc on ne réalise pas d'appel récursif sur ces sommets. La phase d'appels récursifs sur le voisinage du sommet e est terminée.

Sommet	a	b	c	d	e	f	g	h
Couleur	G	G	N	G	B	N	N	N
Père	b	-	f	c	-	b	d	g
Debut	10	0	2	3	11	1	4	5
Fin			8	7		9	6	5

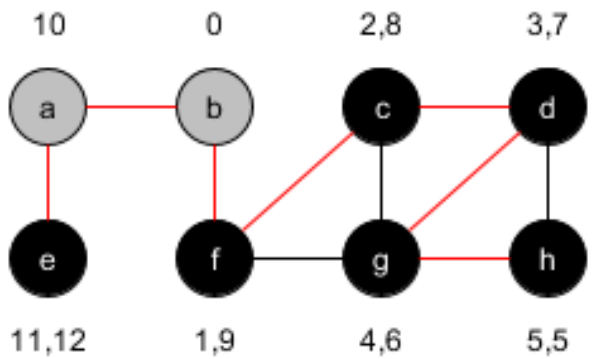


Algorithme du parcours en profondeur

Étape 15.

La phase d'appels récursifs sur le voisinage du sommet *e* est terminée. Le sommet *e* devient noir. Le temps passe à 13. L'appel récursif se termine. On retourne sur le traitement des voisins du sommet *a*.

Sommet	a	b	c	d	e	f	g	h
Couleur	G	G	N	N	B	N	N	N
Père	b	-	f	c	-	b	d	g
Debut	10	0	2	3	11	1	4	5
Fin			8	7	12	9	6	5

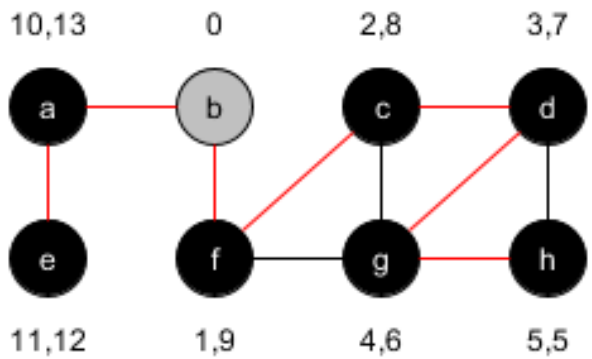


Algorithme du parcours en profondeur

Étape 16.

La phase d'appels récursifs sur le voisinage du sommet *a* est terminée. Le sommet *a* devient noir. Le temps passe à 14. L'appel récursif se termine. On retourne sur le traitement des voisins du sommet *b*.

Sommet	a	b	c	d	e	f	g	h
Couleur	N	G	N	N	B	N	N	N
Père	b	-	f	c	-	b	d	g
Debut	10	0	2	3	11	1	4	5
Fin	13		8	7	12	9	6	5



Algorithme du parcours en profondeur

Étape 17.

La phase d'appels récursifs sur le voisinage du sommet b est terminée. Le sommet b devient noir. Le temps passe à 15. L'appel récursif se termine. L'algorithme se termine.

Sommet	a	b	c	d	e	f	g	h
Couleur	N	N	N	N	B	N	N	N
Père	b	-	f	c	a	b	d	g
Debut	10	0	2	3	11	1	4	5
Fin	13	14	8	7	12	9	6	5