

Algorithmes de graphes

Olivier Baudon
Université de Bordeaux
351, cours de la Libération, 33405 Talence Cedex, France

8 septembre 2021

Résumé

Ces notes présentent les principaux algorithmes de graphes vus dans le cadre de l'UE Algorithmique des graphes de la Licence Informatique 3^{ème} année de l'Université de Bordeaux

0.1 Parcours de graphes

0.1.1 Parcours en largeur

Algorithme 1 Parcours en largeur PL(G, s)

```
1: couleur( $s$ )  $\leftarrow$  GRIS
2:  $d(s) \leftarrow 0$ 
3: pere( $s$ )  $\leftarrow$  NIL
4:  $F \leftarrow \{s\}$ 
5: pour tout  $v \in V(G) \setminus s$  faire
6:   couleur( $v$ )  $\leftarrow$  BLANC
7:    $d(v) \leftarrow \infty$ 
8:   pere( $v$ )  $\leftarrow$  NIL
9: fin pour
10: tant que  $F$  non vide faire
11:    $v \leftarrow \text{tete}(F)$ 
12:   pour tout  $w \in \text{Adj}(v)$  faire
13:     si couleur( $w$ ) = BLANC alors
14:       couleur( $w$ )  $\leftarrow$  GRIS
15:        $d(w) \leftarrow d(v) + 1$ 
16:       pere( $w$ )  $\leftarrow$   $v$ 
17:       Enfiler( $F, w$ )
18:     fin si
19:   fin pour
20:   Defiler( $F$ )
21:   couleur( $v$ )  $\leftarrow$  NOIR
22: fin tant que
```

0.1.2 Parcours en profondeur

Algorithme 2 Parcours en profondeur PP(G)

```
1: pour tout  $v \in V(G)$  faire  
2:    $couleur(v) \leftarrow BLANC$   
3:    $pere(v) \leftarrow NIL$   
4: fin pour  
5:  $temps \leftarrow 0$   
6: pour tout  $v \in V(G)$  faire  
7:   si  $couleur(v) = BLANC$  alors  
8:      $VisiterPP(v)$   
9:   fin si  
10: fin pour
```

Algorithme 3 $VisiterPP(v)$

```
1:  $d(v) \leftarrow temps \leftarrow temps + 1$   
2:  $couleur(v) \leftarrow GRIS$   
3: pour tout  $w \in Adj(v)$  faire  
4:   si  $couleur(w) = BLANC$  alors  
5:      $pere(w) \leftarrow v$   
6:      $VisiterPP(w)$   
7:   fin si  
8: fin pour  
9:  $couleur(v) \leftarrow NOIR$   
10:  $f(v) \leftarrow temps \leftarrow temps + 1$ 
```

0.1.3 Applications du parcours en profondeur

Tri topologique

Au cours d'un parcours en profondeur, à chaque fois que l'on noircit un sommet, il est inséré en tête de liste.

A $PP(G)$, rajouter une ligne 5 bis :

$liste \leftarrow \{\}$

A $VisiterPP(v)$, rajouter
trois lignes 3.1, 3.2, 3.3

si $couleur(w) = GRIS$ **alors**
 retourner "Existence d'un circuit"
fin si

une ligne 11 :

$INSERER_TETE(liste, v)$

Composantes fortement connexes

Algorithme 4 CFC(G)

- 1: Exécuter $PP(G)$ et trier les sommets selon un ordre décroissant de f
 - 2: Calculer G^{-1}
 - 3: Exécuter $PP(G^{-1})$
 - 4: Retourner les arborescences obtenues comme composantes fortement connexes de G
-

0.2 Arbre de poids minimum

0.2.1 Algorithme de Kruskal

Algorithme 5 $Kruskal(G, w)$

```
1: pour tout  $v$  de  $V(G)$  faire
2:    $composante(v) \leftarrow \{v\}$ 
3: fin pour
4: Trier  $E(G)$  dans un ordre croissant  $(e_1, \dots, e_m)$  en fonction de  $w(e)$ 
5:  $i \leftarrow 1$ 
6:  $E(T) \leftarrow \{\}$ 
7: tant que  $|E(T)| < n - 1$  faire
8:   si  $e_i = (u, v)$  et  $composante(u) \neq composante(v)$  alors
9:      $E(T) \leftarrow E(T) \cup \{e_i\}$ 
10:     $UNIFIER(composante(u), composante(v))$ 
11:   fin si
12:    $i \leftarrow i + 1$ 
13: fin tant que
14: retourner  $E(T)$ 
```

0.2.2 Algorithme de Prim

Algorithme 6 $Prim(G, w)$

```
1:  $F \leftarrow FILE\_PRIORITE(V(G), cle)$ 
2: pour tout  $v$  de  $V(G)$  faire
3:    $cle(v) \leftarrow \infty$ 
4: fin pour
5:  $cle(r) \leftarrow 0$ 
6:  $pere(r) \leftarrow NIL$ 
7: tant que  $F \neq \emptyset$  faire
8:    $u \leftarrow EXTRAIRE\_MIN(F)$ 
9:   pour tout  $v \in Adj(u)$  faire
10:    si  $v \in F$  et  $w(u, v) < cle(v)$  alors
11:       $pere(v) \leftarrow u$ 
12:       $cle(v) \leftarrow w(u, v)$ 
13:    fin si
14:   fin pour
15: fin tant que
```

0.3 Plus court chemin

0.3.1 Plus court chemin à partir d'un sommet

Dijkstra

G : graphe orienté

$w : E(G) \rightarrow \mathbb{R}^+$

s : source de G

Algorithme 7 Dijkstra(G, w, s)

```
1: pour tout  $v$  de  $V(G)$  faire
2:    $d(v) \leftarrow \infty$ 
3:    $pere(v) \leftarrow NIL$ 
4:    $couleur(v) \leftarrow BLANC$ 
5: fin pour
6:  $d(s) \leftarrow 0$ 
7:  $F \leftarrow FILE\_PRIORITE(\{s\}, d)$ 
8: tant que  $F \neq \emptyset$  faire
9:    $pivot \leftarrow EXTRAIRE\_MIN(F)$ 
10:  pour tout  $e = (pivot, v)$  arc sortant de  $pivot$  faire
11:    si  $couleur(v) = BLANC$  alors
12:      si  $d(v) = \infty$  alors
13:        INSERER( $F, v$ )
14:      fin si
15:      si  $d[v] > d[pivot] + w(e)$  alors
16:         $d[v] \leftarrow d[pivot] + w(e)$ 
17:         $pere[v] \leftarrow pivot$ 
18:      fin si
19:    fin si
20:  fin pour
21:   $couleur[pivot] \leftarrow NOIR$ 
22: fin tant que
```

Bellman

G : graphe orienté sans circuit

$w : E(G) \rightarrow \mathbb{R}$

s : source de G

Algorithme 8 Bellman-court(G, w, s)

```
1: TriTopologique(G)
2: Soit  $v_1, \dots, v_n$  l'ordre topologique calculé à l'étape 1
3: pour  $i$  de 1 à  $n$  faire
4:    $d[v_i] \leftarrow \infty$ 
5:    $pere[v_i] \leftarrow NIL$ 
6: fin pour
7:  $d[s] \leftarrow 0$ 
8: Soit  $j$  l'indice de  $s$  dans l'ordre topologique calculé à l'étape 1
9: pour  $i$  de  $j$  à  $n - 1$  faire
10:  pour  $u \in Adj(v_i)$  faire
11:    si  $d[u] > d[v_i] + w(v_i, u)$  alors
12:       $d[u] \leftarrow d[v_i] + w[v_i, u]$ 
13:       $pere[u] \leftarrow v_i$ 
14:    fin si
15:  fin pour
16: fin pour
```

Algorithme 9 Bellman-long(G, w, s)

```
1: pour tout  $v$  de  $V(G)$  faire
2:    $d[v] \leftarrow \infty$ 
3:    $pere[v] \leftarrow NIL$ 
4:    $npred[v] \leftarrow deg^-[v]$ 
5:   si  $npred(v) = 0$  alors
6:     INSERER_FILE(F, v)
7:   fin si
8: fin pour
9:  $d[s] \leftarrow 0$ 
10: tant que F non vide faire
11:   $u \leftarrow TETE\_FILE(F)$ 
12:  DEFILER(F)
13:  pour  $v \in Adj(u)$  faire
14:    si  $d[v] > d[u] + w(u, v)$  alors
15:       $d[v] \leftarrow d[u] + w[u, v]$ 
16:       $pere[v] \leftarrow u$ 
17:    fin si
18:     $npred(v) \leftarrow npred[v] - 1$ 
19:    si  $npred(v) = 0$  alors
20:      INSERER_FILE(F, v)
21:    fin si
22:  fin pour
23: fin tant que
```

Algorithme de Ford

G : graphe orienté

$w : E(G) \rightarrow \mathbb{R}$

s : source de G

L'algorithme renvoie vrai si le graphe G est sans circuit négatif.

Algorithme 10 Ford(G, w, s)

```
1: pour tout  $v$  de  $V(G)$  faire
2:    $d[v] \leftarrow \infty$ 
3:    $pere[v] \leftarrow NIL$ 
4: fin pour
5:  $d[s] \leftarrow 0$ 
6: pour  $i$  de 1 à  $n - 1$  faire
7:   pour tout arc  $e = (u, v) \in E(G)$  faire
8:     si  $d[v] > d[u] + w(u, v)$  alors
9:        $d[v] \leftarrow d[u] + w[u, v]$ 
10:       $pere[v] \leftarrow u$ 
11:    fin si
12:  fin pour
13: fin pour
14: pour tout arc  $e = (u, v) \in E(G)$  faire
15:   si  $d[v] > d[u] + w(u, v)$  alors
16:     retourner FAUX
17:   fin si
18: fin pour
19: retourner VRAI
```

0.3.2 Plus courts chemins entre toutes paires de sommets

Algorithme de Floyd

G est un graphe orienté, munis d'une fonction de poids sur les arcs w .

Les sommets sont numérotés de 1 à n .

$W_{i,j}$ contiendra la plus courte distance entre le sommet i et le sommet j ,

$Pred_{i,j}$ le sommet prédécesseur de j sur un plus court chemin de i à j .

Algorithme 11 Floyd(G, w)

```
1: pour  $i$  de 1 à  $n$  faire
2:   pour  $j$  de 1 à  $n$  faire
3:     si  $i = j$  alors
4:        $W(i, j) \leftarrow 0$ 
5:        $Pred(i, j) \leftarrow NIL$ 
6:     sinon si  $ij \in E(G)$  alors
7:        $W(i, j) \leftarrow w(i, j)$ 
8:        $Pred(i, j) \leftarrow i$ 
9:     sinon
10:       $W(i, j) \leftarrow \infty$ 
11:       $Pred(i, j) \leftarrow NIL$ 
12:    fin si
13:  fin pour
14: fin pour
15: pour  $k$  de 1 à  $n$  faire
16:   pour  $i$  de 1 à  $n$  faire
17:    pour  $j$  de 1 à  $n$  faire
18:     si  $W(i, k) + W(k, j) < W(i, j)$  alors
19:        $W(i, j) \leftarrow W(i, k) + W(k, j)$ 
20:        $Pred(i, j) \leftarrow Pred(k, j)$ 
21:     fin si
22:   fin pour
23: fin pour
24: fin pour
```

Algorithme de Warshall

L'algorithme de Warshall est une adaptation de l'algorithme de Floyd au calcul de la fermeture transitive d'un graphe orienté. $W_{i,j}$ vaudra 1 s'il existe un chemin de i à j dans le graphe G , 0 sinon.

Algorithme 12 Warshall(G, w)

```
1: pour  $i$  de 1 à  $n$  faire
2:   pour  $j$  de 1 à  $n$  faire
3:     si  $i = j$  ||  $ij \in E(G)$  alors
4:        $W(i, j) \leftarrow 1$ 
5:     sinon
6:        $W(i, j) \leftarrow 0$ 
7:     fin si
8:   fin pour
9: fin pour
10: pour  $k$  de 1 à  $n$  faire
11:   pour  $i$  de 1 à  $n$  faire
12:     pour  $j$  de 1 à  $n$  faire
13:        $W(i, j) \leftarrow (W(i, k) \& W(k, j)) \vee W(i, j)$ 
14:     fin pour
15:   fin pour
16: fin pour
```

0.4 Flots

0.4.1 Algorithme de Ford et Fulkerson

G : graphe orienté

$c : E(G) \rightarrow \mathbb{R}^+$

s : source de G

t : puit de G

On considère un arc (t, s) de capacité $c(t, s)$ infinie. La valeur du flot sur cet arc sera la "valeur du flot de s à t ".

On note respectivement par $I(e)$ et $T(e)$ l'extrémité initiale et l'extrémité terminale d'un arc e .

Algorithme 13 FlotMax(G, c, s, t)

```
1: pour tout  $e$  de  $E(G)$  faire
2:    $f[e] \leftarrow 0$ 
3: fin pour
4: répéter
5:   Marquage( $G, c, f, s, t$ )
6:   si  $t \in Y$  alors
7:      $v \leftarrow t$ 
8:      $C^+ \leftarrow \{(t, s)\}$ 
9:      $C^- \leftarrow \emptyset$ 
10:    tant que  $v \neq s$  faire
11:       $e \leftarrow A[v]$ 
12:      si  $v = T[e]$  alors
13:         $C^+ \leftarrow C^+ \cup \{e\}$ 
14:         $v \leftarrow I[e]$ 
15:      sinon
16:         $C^- \leftarrow C^- \cup \{e\}$ 
17:         $v \leftarrow T[e]$ 
18:      fin si
19:    fin tant que
20:  fin si
21:  pour tout  $e \in C^+$  faire
22:     $f(e) \leftarrow f(e) + \delta[t]$ 
23:  fin pour
24:  pour tout  $e \in C^-$  faire
25:     $f(e) \leftarrow f(e) - \delta[t]$ 
26:  fin pour
27: jusqu'à  $t \notin Y$ 
```

Algorithme 14 Marquage(G, c, f, s, t)

```
1:  $Y \leftarrow \{s\}$ 
2:  $\delta(s) \leftarrow +\infty$ 
3:  $Max \leftarrow \text{faux}$ 
4: tant que  $t \notin Y$  et  $Max = \text{faux}$  faire
5:   si il existe  $e = (u, v)$  avec  $u \in Y, v \notin Y, f(e) < c(e)$  alors
6:      $Y \leftarrow Y \cup \{v\}$ 
7:      $A[v] \leftarrow e$ 
8:      $\delta[v] \leftarrow \min(\delta[u], c(e) - f(e))$ 
9:   sinon
10:    si il existe  $e = (u, v)$  avec  $v \in Y, u \notin Y, f(e) > 0$  alors
11:       $Y \leftarrow Y \cup \{u\}$ 
12:       $A[u] \leftarrow e$ 
13:       $\delta[u] \leftarrow \min(\delta[v], f(e))$ 
14:    sinon
15:       $Max \leftarrow \text{vrai}$ 
16:    fin si
17:  fin si
18: fin tant que
```
