Théorie des Graphes

Université Libanaise
Faculté des Sciences
License Informatique
2ème année – S3

Syllabus

1. Concepts introductifs
2. Introduction aux graphes et à leurs utilisations
3. Arbres et graphes bipartis
4. Distance et connexité
5. Matrices
6. Algorithmes sur les graphes
7. Coloration des graphes
8. Graphes Eulériens et Hamiltoniens
9. Digraphes et réseaux
10. Graphes planaires

Digraphes et réseaux

Semaine 9

Digraphes et réseaux

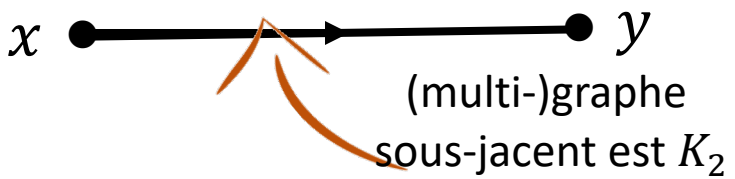
- Les **digraphes** sont similaires aux graphes sauf qu'il y a des directions sur les arêtes. Les digraphes sont utilisés pour modéliser des problèmes où la direction du flux d'une certaine quantité (informations, trafic, liquide, électrons, etc.) est importante.
- Lorsqu'il y a des limites de cette quantité qui peut s'écouler à travers une arête orientée particulière, nous obtenons un **réseau**.
- Un type spécial de digraphe n'ayant pas de cycles dirigés, appelé **digraphe d'activité**, a des poids sur les arêtes orientées indiquant la durée d'une activité donnée. Ces digraphes sont utilisés pour aider à planifier des activités individuelles qui composent un projet complexe.

Plan

- Graphes orientés
- Réseaux
- La méthode du chemin critique

Graphes orientés

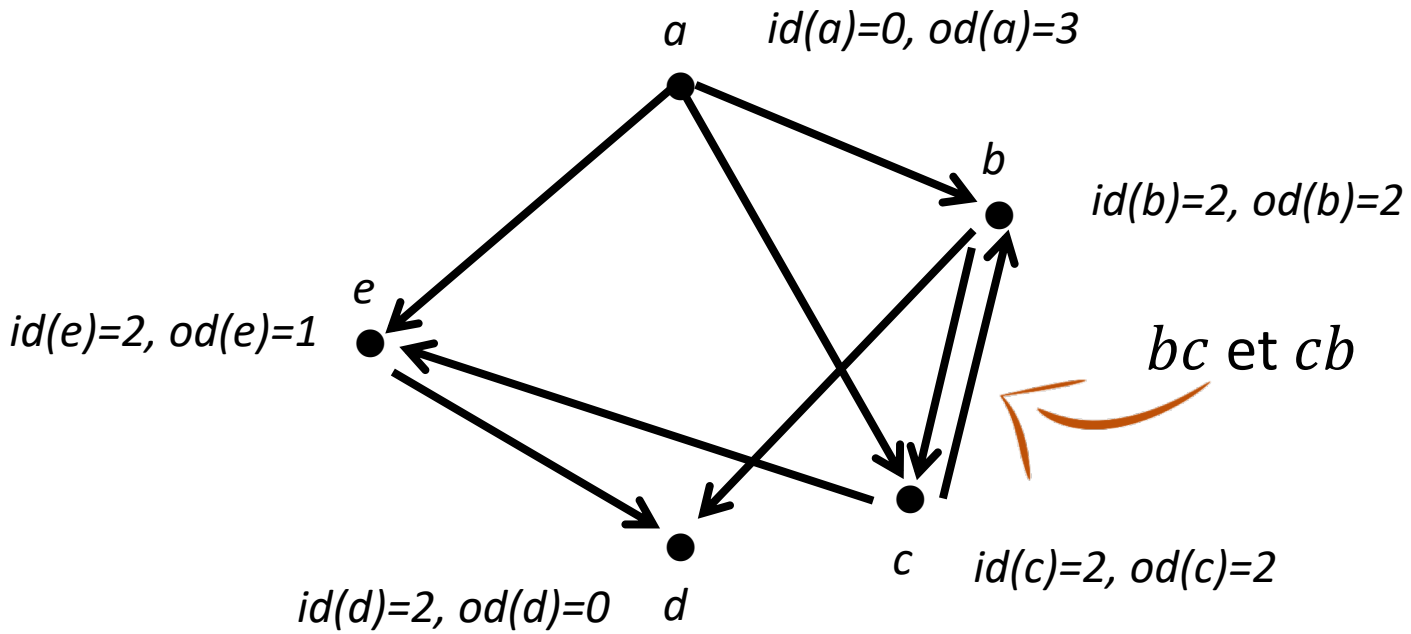
- Un **graphe orienté**, abrégé **digraphe**, D, consiste en un ensemble fini non vide de sommets $V(D)$ avec un ensemble de paires **ordonnées** de sommets distincts appelés **arcs**. Un arc xy va de x à y et est représenté par une flèche de x vers y.
- Noter que:
  - x est **adjacent à** y tandis que y est **adjacent de** x.
  - y n'est pas adjacent à x.
  - Lorsque les arcs uv et vu sont tous deux présents, ce sont des **arcs symétriques**.
- Étant donné un digraphe, le graphe avec chaque arc remplacé par une arête est appelé le **(multi-)graphe sous-jacent**.
  - Le **degré sortant od(v) (outdegree)** du sommet v est le nombre de sommets auxquels v est adjacent à.
  - Le **degré entrant id(v) (indegree)** est le nombre de sommets dont v est adjacent de.



Graphes orientés

EXEMPLE

- Dessiner le digraphe D avec $V(D) = \{a, b, c, d, e\}$ et $E(D) = \{ab, ac, ae, bc, bd, cb, ce, ed\}$. Puis indiquer le degré entrant et le degré sortant de chaque sommet. Quels arcs sont symétriques?



Graphes orientés

- Un sommet de degré entrant 0 est appelé **émetteur** (**transmitter**) car, dans un contexte de télécommunications, il envoie des informations mais n'en reçoit aucune.
- Un sommet de degré sortant 0 est appelé un **récepteur** (**receiver**) car il reçoit des informations mais n'en envoie aucune.
- Pour tout digraphe D, nous avons le résultat simple suivant qui est analogue pour les graphes.
  Théorème Si D est un digraphe avec l'ensemble de sommets $V(D) = \{v_1, v_2, \dots, v_n\}$ et ayant q arcs, alors $\sum_{i=1}^{n} id(v_i) = \sum_{i=1}^{n} od(v_i) = q$.

Graphes orientés

- Un digraphe avec la propriété que toute paire de sommets u et v a à la fois un chemin $u - v$ dirigé et un chemin $v - u$ dirigé est appelé **fortement connexe**, ou simplement, **fort**.
- Étant donné un graphe connexe, nous décrivons l'action d'attribuer une direction à chaque arête comme **orientant** le graphe.
- Si le graphe résultant est fortement connexe, nous appelons l'orientation **forte**.
- Le problème de l'obtention d'une forte orientation pour un graphe connexe est très pratique. Il est utile pour rationaliser la circulation dans une ville animée en rendant toutes les rues à sens unique.

Atteindre une forte orientation

- Il doit être clair qu'un graphe avec un pont ne peut pas être fortement orienté.
  Théorème Un graphe connexe G a une forte orientation si et seulement s'il ne contient pas de ponts; c'est-à-dire que chaque arête est contenue dans un cycle.

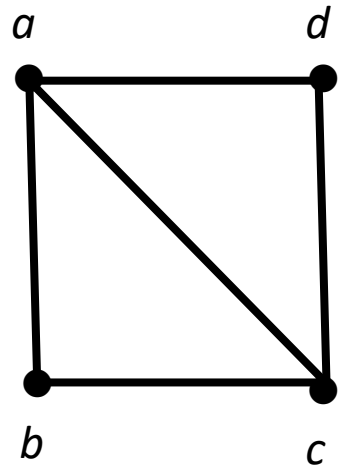
Algorithme (Orientation d'un graphe connecté sans pont)

1. Obtenir un arbre DFS T. Attribuer un nombre croissant à chaque sommet empilé.
2. Orienter chaque arête de T vers le sommet avec le nombre le plus élevé.
3. Orienter chacune des arêtes restantes de G, c'est-à-dire celles qui ne sont pas dans T, vers le sommet de plus petit nombre.

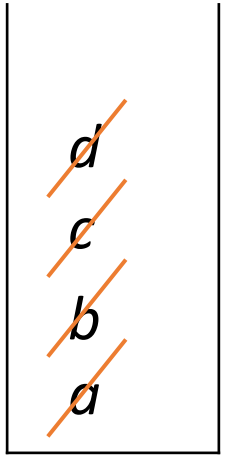
Atteindre une forte orientation

EXEMPLE

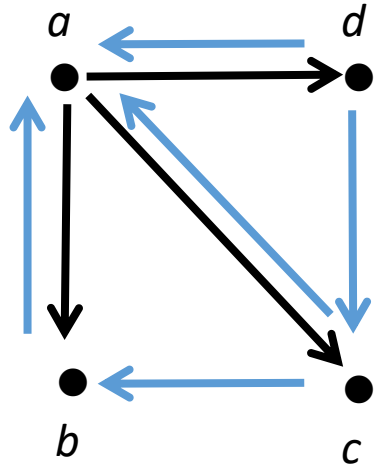
Orienter fortement le graphe suivant.



v	$N(v)$
a	b, c, d
b	a, c
c	a, b, d
d	a, c

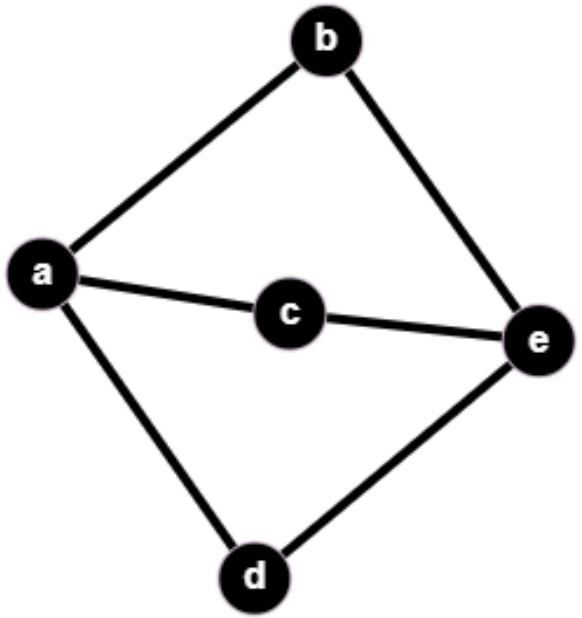


Sommet	a	b	c	d
Parent	-1	a	a	a
Numéro	1	2	3	4

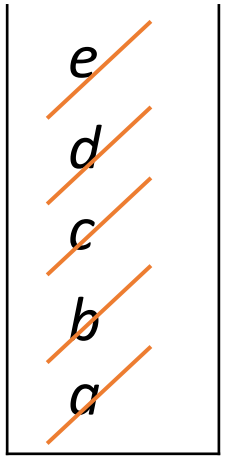


Digraphes et réseaux
Question 1

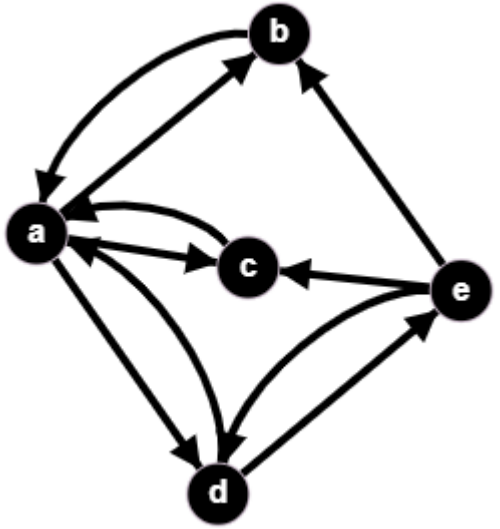
Orienter fortement le graphe suivant.



v	$N(v)$
a	*b, c, d*
b	*a, e*
c	*a, e*
d	*a, e*
e	*b, c, d*



Sommet	*a*	*b*	*c*	*d*	*e*
Parent	*-1*	*a*	*a*	*a*	*d*
Numéro	*1*	*2*	*3*	*4*	*5*



Plan

- Graphes orientés
- Réseaux
- La méthode du chemin critique

Réseaux

- Lors de l'analyse de divers problèmes de transport, de télécommunication ou de recherche opérationnelle, les digraphes sont utilisés dans lesquels des informations supplémentaires sont nécessaires en plus de la direction sur les arcs.
- Par exemple, les informations requises peuvent être la quantité de trafic qu'un arc particulier (route ou câble) peut transporter et la quantité de trafic qu'il transporte actuellement. Dans de telles situations, une structure légèrement plus complexe appelée réseau est utilisée pour modéliser et analyser le problème.

Plan

- Graphes orientés
- Réseaux
  - Distance dans les réseaux
    - Algorithme de Dijkstra
    - Algorithme Bellman-Ford
  - Flux de réseau
    - Algorithme Ford Fulkerson
    - Algorithme d'Edmonds – Karp
  - Réseaux et appariements
- La méthode du chemin critique

Distance dans les réseaux

- Considérons un digraphe pondéré où le poids $w(uv)$ sur l'arc uv désigne la distance de u à v parcourant cet arc.
- Comment trouver le meilleur itinéraire d'un sommet à un autre?
- **Algorithme de Dijkstra**: cet algorithme partage certaines similitudes avec l'algorithme de BFS et de Prim; cependant, ici un sommet v reçoit une étiquette temporaire indiquant la distance de v à la racine r. Cette distance est mise à jour lorsqu'un chemin plus court de r à v est trouvé. Le sommet cible est le sommet dont nous essayons de déterminer la distance de la racine. $N(v)$ désigne la liste de contiguïté du sommet v.

C’est quoi une bifile / file à deux extrémités (*Deque*)?

- C'est comme une file d'attente où l'ordre dans lequel les données arrivent est important.
- Les données ont une priorité.
- Les données sont mises en file d'attente en fonction de leur priorité.

Distance dans les réseaux
Algorithme de Dijkstra

1. Créer une **bifile** vide Q.
2. Créer la liste **`distance`**, définissez toutes les distances comme -1.
   Créer la liste **`parent`**.
   Mettre la priorité de la racine à 0, puis enfiler-le dans Q.
3. Itérer sur Q
   1. Si aucun élément dans Q, l'algorithme se termine
   2. S'il y a un sommet v:
      - supprimer v de Q;
      - Itérer sur ses sommets adjacents w
        - Si la distance de w est -1 **OU** si la distance de w est supérieure à la distance de v + w (vw)
        1. mettre à jour la distance de w (distance de v + $w(vw)$)
        2. enfiler w avec la priorité de sa distance ou mettre à jour sa priorité
        3. mettre à jour le parent de w (w est accessible depuis v)

Distance dans les réseaux
Algorithme de Dijkstra

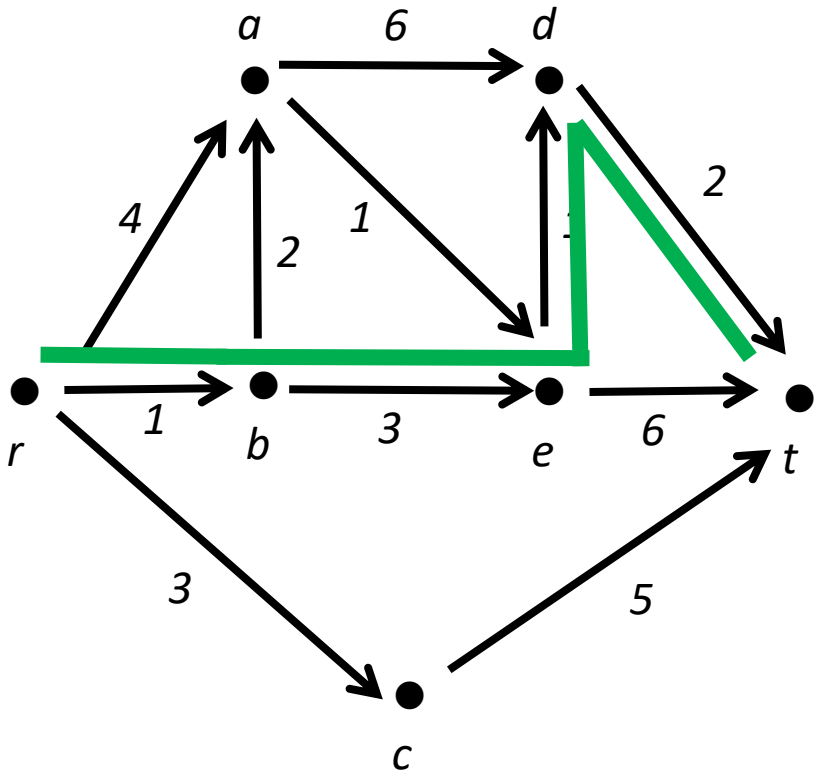
- L’algorithme de Dijkstra s’applique à la fois aux graphes orientés et non orientés.
- Dans les deux cas, il y a généralement des poids sur les arêtes; cependant, s'il n'y a pas de tels poids, nous les définissons initialement comme 1.
- La principale différence pour les graphes pondérés dirigés est que la liste d’adjacence $N(v)$ pour chaque sommet v correspond uniquement aux sommets auxquels v est adjacent à, et non à ceux dont v est adjacent de. Ainsi, nous pourrions penser à $N(v)$ comme le voisinage extérieur de v.

Distance dans les réseaux
Algorithme de Dijkstra

Utilisez l'algorithme de Dijkstra pour trouver la distance de r à t.

On devrait laisser a derrière c

EXEMPLE



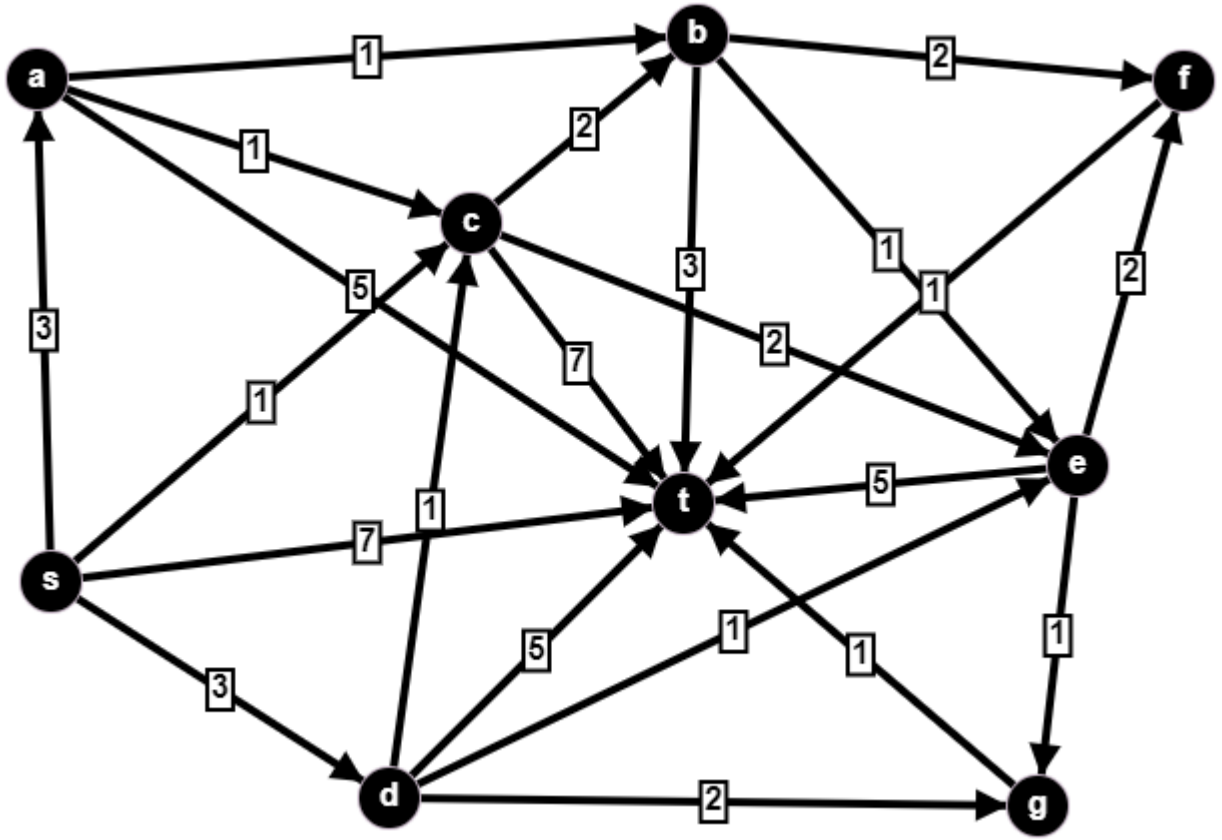
v	$N(v)$
a	d(6), e(1)
b	a(2), e(3)
c	t(5)
d	t(2)
e	d(1), t(6)
r	a(4), b(1), c(3)
t	

~~r(0)~~ ~~b(1)~~ ~~c(3)~~ ~~a(3)~~ ~~e(4)~~ ~~d(9)~~ ~~d(8)~~

Sommet	a	b	c	d	e	r	t
Distance	~~4~~	~~1~~	~~3~~	~~9~~	~~4~~	~~0~~	~~8~~
Parent	b	r	r	a	b		d

Digraphes et réseaux
Question 2

Utilisez l'algorithme de Dijkstra pour trouver la distance de s à t.



v	$N(v)$
a	b(1), c(1),t(5)
b	e(1), f(2), t(3)
c	b(2),e(2), t(7)
d	c(1),e(1),g(2),t(5)
e	f(2),g(1),t(5)
f	t(1)
g	t(1)
s	a(3), c(1),d(3),t(7)
t	

SOLUTION

---

s(0) *c(1) a(3) d(3) t(7)*

Sommet	a	b	c	d	e	f	g	s	t
Distance	3		1	3				0	7
Parent	s		s	s					s

---

c(1) *a(3) b(3) d(3) e(3) t(7)*

Sommet	a	b	c	d	e	f	g	s	t
Distance	3	3	1	3	3			0	7
Parent	s	c	s	s	c				s

---

a(3) *b(3) d(3) e(3) t(7)*

Sommet	a	b	c	d	e	f	g	s	t
Distance	3	3	1	3	3			0	7
Parent	s	c	s	s	c				s

---

b(3) *d(3) e(3) b(5) t(6)*

Sommet	a	b	c	d	e	f	g	s	t
Distance	3	3	1	3	3	5		0	6
Parent	s	c	s	s	c	b			b

---

d(3) *e(3) b(5) g(5) t(6)*

Sommet	a	b	c	d	e	f	g	s	t
Distance	3	3	1	3	3	5	5	0	6
Parent	s	c	s	s	c	b	d		b

---

e(3) *g(4) b(5) t(6)*

Sommet	a	b	c	d	e	f	g	s	t
Distance	3	3	1	3	3	5	4	0	6
Parent	s	c	s	s	c	b	e		b

v	$N(v)$
a	b(1), c(1),t(5)
b	e(1), f(2), t(3)
c	b(2),e(2), t(7)
d	c(1),e(1),g(2),t(5)
e	f(2),g(1),t(5)
f	t(1)
g	t(1)
s	a(3), c(1),d(3),t(7)
t	

SOLUTION

---

e(3) *g(4) b(5) t(6)*

Sommet	a	b	c	d	e	f	g	s	t
Distance	3	3	1	3	3	5	4	0	6
Parent	s	c	s	s	c	b	e		b

---

g(4) *b(5) t(5)*

Sommet	a	b	c	d	e	f	g	s	t
Distance	3	3	1	3	3	5	4	0	5
Parent	s	c	s	s	c	b	e		g

---

b(5) *t(5)*

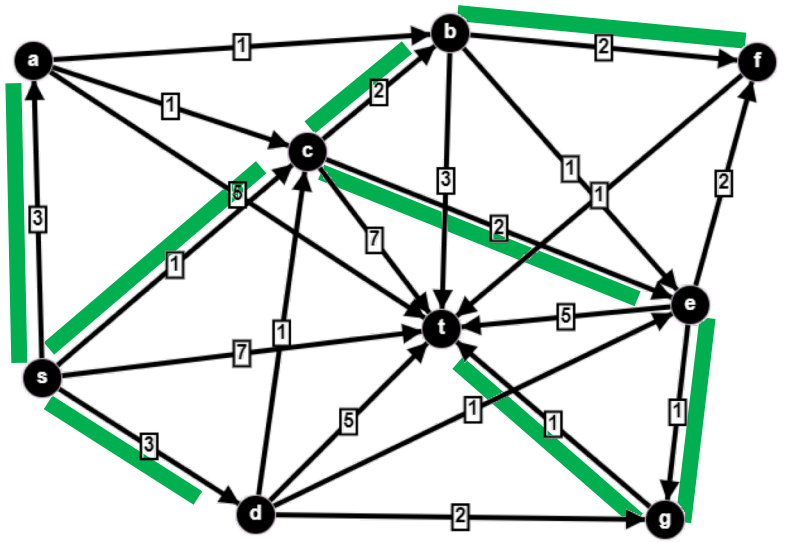
Sommet	a	b	c	d	e	f	g	s	t
Distance	3	3	1	3	3	5	4	0	5
Parent	s	c	s	s	c	b	e		g

---

t(5)

Sommet	a	b	c	d	e	f	g	s	t
Distance	3	3	1	3	3	5	4	0	5
Parent	s	c	s	s	c	b	e		g

La longueur de chemin la plus courte est de 5: s⇒c⇒e⇒g⇒t



Distance dans les réseaux
Algorithme de Dijkstra

- Il est important de noter que bien que nous ayons trouvé la distance de s à t, l'algorithme de Dijkstra trouve en fait la distance entre le sommet racine et tous les autres sommets du réseau.
- En modifiant successivement la racine, nous pouvons trouver la distance entre toutes les paires de sommets en utilisant l'algorithme de Dijkstra.

Plan

- Graphes orientés
- Réseaux
  - Distance dans les réseaux
    - Algorithme de Dijkstra
    - Algorithme Bellman-Ford
  - Flux de réseau
    - Algorithme Ford Fulkerson
    - Algorithme d'Edmonds – Karp
  - Réseaux et appariements
- La méthode du chemin critique

Distance dans les réseaux
Algorithme de Bellman-Ford

- Si le graphe a des coûts d’arêtes négatifs, alors l'algorithme de Dijkstra ne fonctionne pas.
- Une combinaison de l'algorithme de Dijkstra et d'algorithmes non pondérés résoudra le problème.

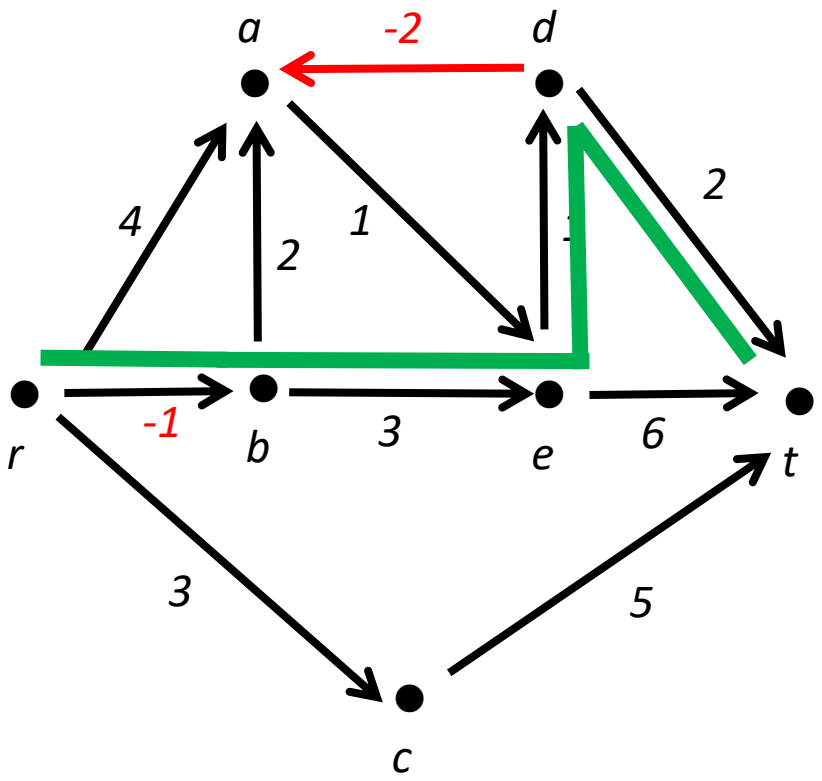
Distance dans les réseaux
Algorithme de Bellman-Ford

1. Créer une **file d'attente** vide Q.
2. Créer la liste **`distance`**, définissez toutes les distances comme $+\infty$.
   Créer la liste **`parent`**.
   Créer une liste booléenne **`Inqueue`** qui indique si un sommet est dans Q.
   Mettre la distance de la racine à 0, mettre son état *Inqueue* à Vrai, puis enfiler-le dans Q.
3. Itérer sur Q
   1. Si aucun élément dans Q, l'algorithme se termine
   2. S'il y a un sommet v:
      - supprimer v de Q;
      - mettre son état *Inqueue* à Faux
      - Itérer sur ses sommets adjacents w
        - Si la distance de w est supérieure à la distance de v + w (vw)
        1. mettre à jour la distance de w (distance de v + $w(vw)$)
        2. mettre à jour le parent de w (w est accessible depuis v)
        3. Si w n'est pas dans Q, mettre son état*Inqueue* à Vrai, puis enfiler w dans Q

Distance dans les réseaux
Algorithme de Bellman-Ford

Utilisez l'algorithme de Bellman-Ford pour trouver la distance entre r et t.

EXEMPLE



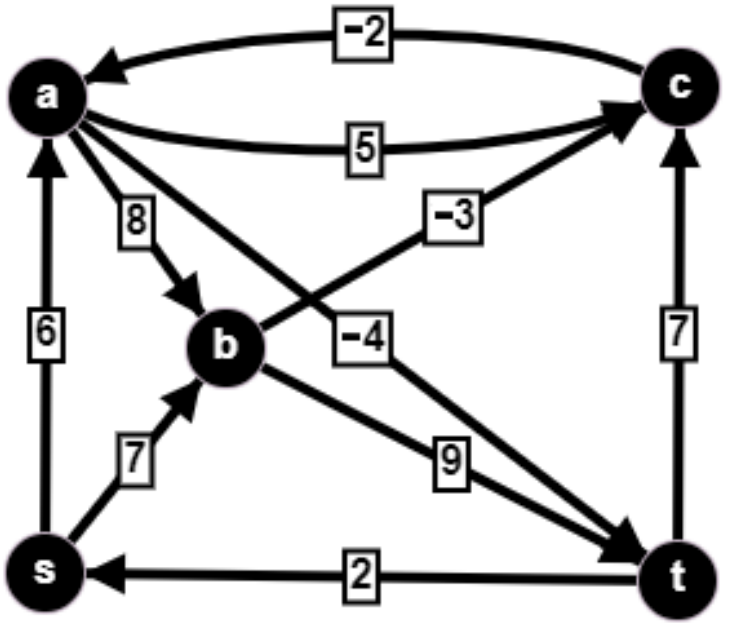
v	$N(v)$
a	e(1)
b	a(2), e(3)
c	t(5)
d	a(-2), t(2)
e	d(1), t(6)
r	a(4), b(-1), c(3)
t	

r a b c e a t d t

Sommet	a	b	c	d	e	r	t
Distance	+4	+1	+3	+3	+2	0	+8
Parent	b	r	r	e	d		d
Inqueue	0	0	0	0	0	0	0

Digraphes et réseaux
Question 3

Utilisez l'algorithme de Bellman-Ford pour trouver la distance entre s et t.



v	$N(v)$
a	b(8),c(5),t(-4)
b	c(-3), t(9)
c	a(-2)
s	a(6),b(7)
t	c(7),s(2)

SOLUTION

s a b

Vertex	a	b	c	s	t
distance	6	7	+∞	+0	+∞
Parent	s	s			
Inqueue	1	1	0	0	0

a b c t

Vertex	a	b	c	s	t
distance	6	7	11	0	2
Parent	s	s	a		a
Inqueue	0	1	1	0	1

b c t

Vertex	a	b	c	s	t
distance	6	7	4	0	2
Parent	s	s	b		a
Inqueue	0	0	1	0	1

c t a

Vertex	a	b	c	s	t
distance	2	7	4	0	2
Parent	c	s	b		a
Inqueue	1	0	0	0	1

t a

Vertex	a	b	c	s	t
distance	2	7	4	0	2
Parent	c	s	b		a
Inqueue	1	0	0	0	0

a t

Vertex	a	b	c	s	t
distance	2	7	4	0	-2
Parent	c	s	b		a
Inqueue	0	0	0	0	1

v	$N(v)$
a	b(8),c(5),t(-4)
b	c(-3), t(9)
c	a(-2)
s	a(6),b(7)
t	c(7),s(2)

SOLUTION

---

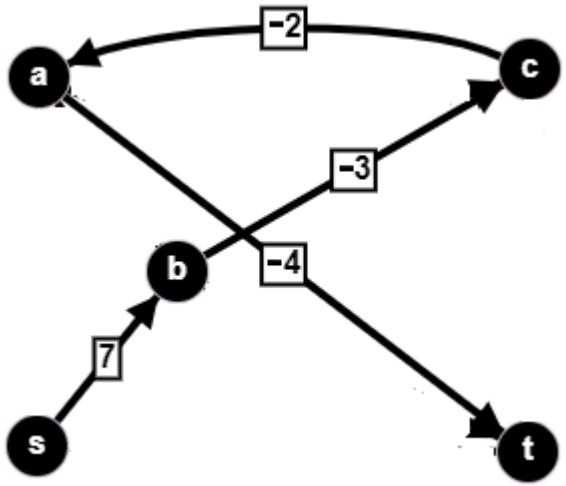
a t

Vertex	a	b	c	s	t
distance	2	7	4	0	-2
Parent	c	s	b		a
Inqueue	0	0	0	0	1

---

t

Vertex	a	b	c	s	t
distance	2	7	4	0	-2
Parent	c	s	b		a
Inqueue	0	0	0	0	0



Plan

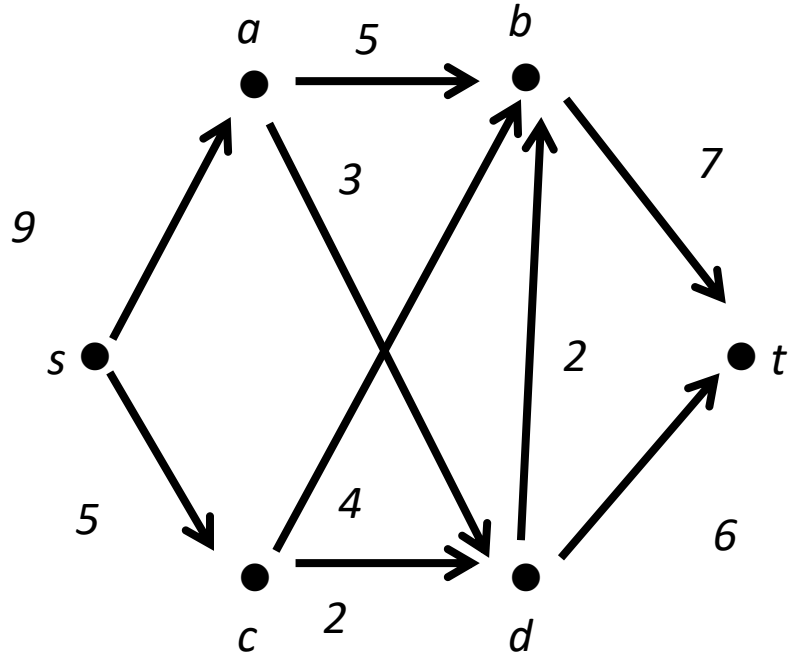
- Graphes orientés
- Réseaux
  - Distance dans les réseaux
    - Algorithme de Dijkstra
    - Algorithme Bellman-Ford
  - Flux de réseau
    - Algorithme Ford Fulkerson
    - Algorithme d'Edmonds – Karp
  - Réseaux et appariements
- La méthode du chemin critique

Flux réseau

- Un **réseau à deux terminaux** N se compose d'un digraphe connecté pondéré qui a deux sommets distincts s et t, appelés **terminaux** , ainsi qu'une fonction à valeur réelle non négative c définie sur les arcs de N.
- Le sommet s est appelé la **source** et a un degré entrant égal à zéro, c'est donc un émetteur dans le digraphe sous-jacent.
- Le sommet t est appelé le **puits** et a un degré sortant égal à zéro, il s'agit donc d'un récepteur dans le digraphe sous-jacent.
- Les sommets restants sont appelés sommets **intermédiaires** du réseau.
- En général, un réseau n'a pas besoin de terminaux, et certains peuvent en avoir plusieurs, mais nous nous concentrerons sur le problème des deux terminaux, avec une source et un puits.
- Pour chaque arc e, le poids $c(e)$ attribué est appelé la **capacité** de cet arc.

Flux réseau

- Dans un problème de transport, les sommets a, b, c, et d représentent les points auxquels les articles seront transférés d'un mécanisme d'expédition à un autre. Vous pouvez penser que les articles sont transférés d'un camion à une voie ferrée ou à un navire.
- Dans un contexte de télécommunications, vous pouvez considérer les sommets comme des points d'où les informations sont transférées (par exemple, un réseau local vers un réseau étendu ou une ligne de jonction).



Flux réseau

- Un **flux** (légal) dans un réseau est une affectation d'un nombre réel non négatif à chaque arc de afin que les deux conditions suivantes soient remplies:

1. pour chaque arc e, $0 < f(e) < c(e)$
2. (**conservation du flot**) pour chaque sommet v autre que s et t, $\sum_{uv \in N} f(uv) - \sum_{vx \in N} f(vx) = 0$.

Flux réseau

- Le problème général d'intérêt dans les problèmes de réseau est de **maximiser** le flux de marchandises de la source s vers le puits t avec des flux légaux le long des arcs du réseau N.
- Le **flot** F(débit total) dans un réseau est défini comme étant le débit sortant de la source s ou, de manière équivalente, le débit entrant dans le puits t; C'est,

$$F = \sum_{su \in N} f(su) = \sum_{xt \in N} f(xt)$$

- Nous voulons donc maximiser F.

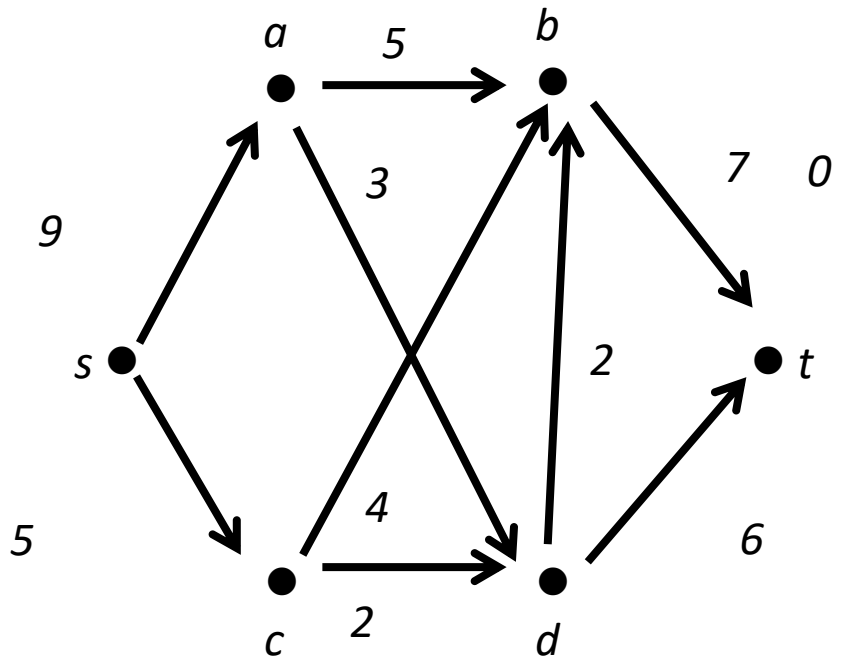
Flux réseau

- Le **mou** (slack) $sl(e)$ de l'arc est la quantité par laquelle la capacité dépasse le débit, donc $sl(e) = c(e) - f(e)$.
- $sl(e)$ représente donc la capacité restante disponible en l'arc e.
- Si $sl(e) = 0$, alors l'arc est **saturé** .
- De toute évidence, si chaque arc menant au puits est saturé, alors nous avons atteint un flot maximal.
  De même, si chaque arc sortant de la source est saturé, alors nous avons atteint un flot maximal.
- Cependant, aucune de ces deux situations n'est nécessaire, comme nous le verrons.

Flux réseau

Essayez d'obtenir un flot maximal.

EXEMPLE

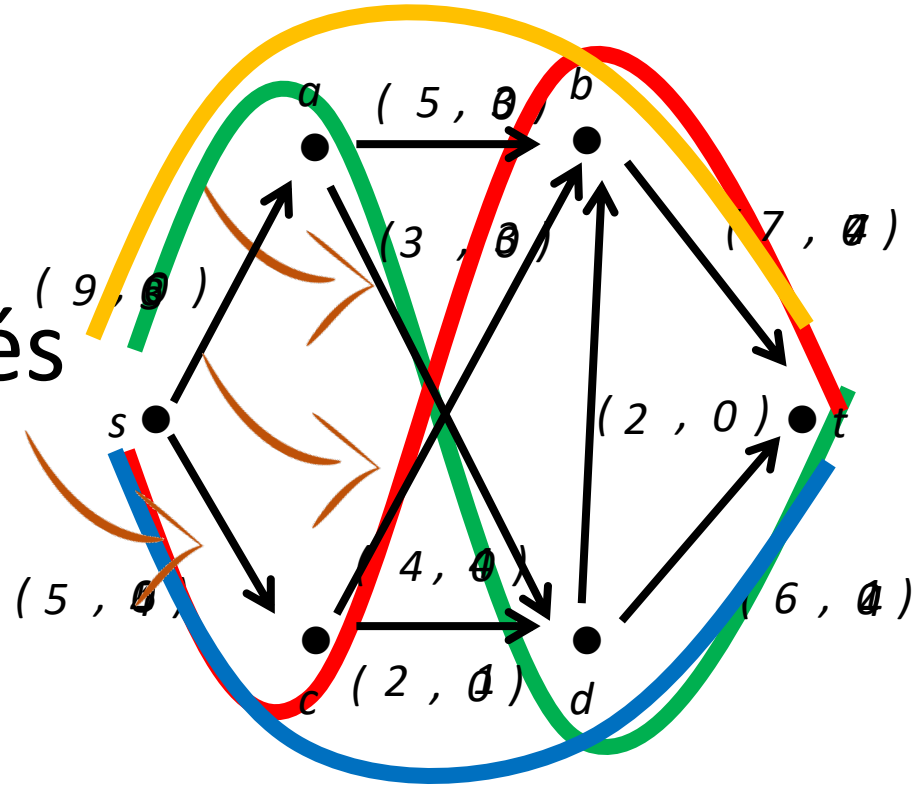


Flux réseau

EXEMPLE

Essayez d'obtenir un flot maximal.

Arcs saturés



Nous commencerons par trouver successivement les chemins de s à t au le long desquels nous pouvons augmenter le flot. Pour un tel chemin, nous pourrons augmenter le flot par le minimum des mous des arcs de ce chemin.

Le chemin s, c, b, t. Mou minimal est 4. Envoyer un flux de 4 le long du chemin.

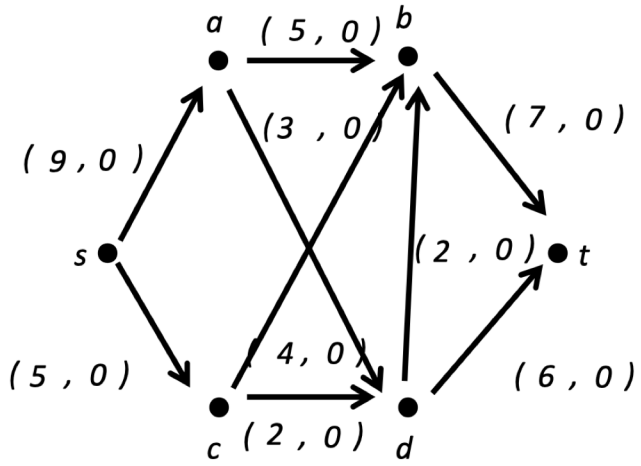
Le chemin s, c, d, t. Mou minimal est 1. Envoyer un flux de 1 le long du chemin.

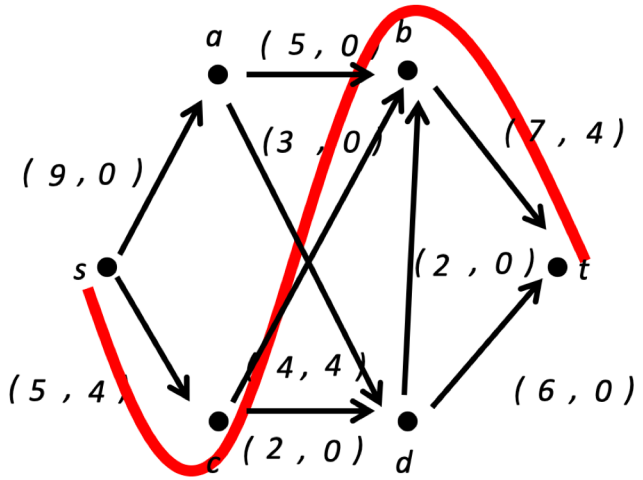
Le chemin s, a, d, t. Mou minimal est 3. Envoyer un flux de 3 le long du chemin.

Le chemin s, a, b, t. Mou minimal est 3. Envoyer un flux de 3 le long du chemin.

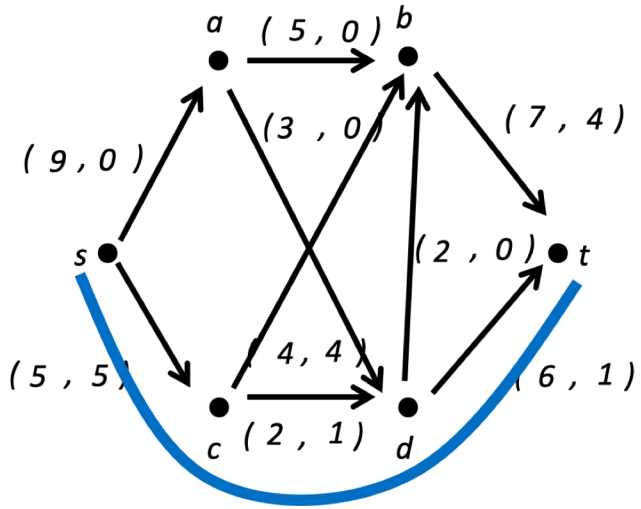
Il n'y a pas de chemin de s à t qui puisse être augmenté. Avons-nous flot maximal? Flot total 11

NON!

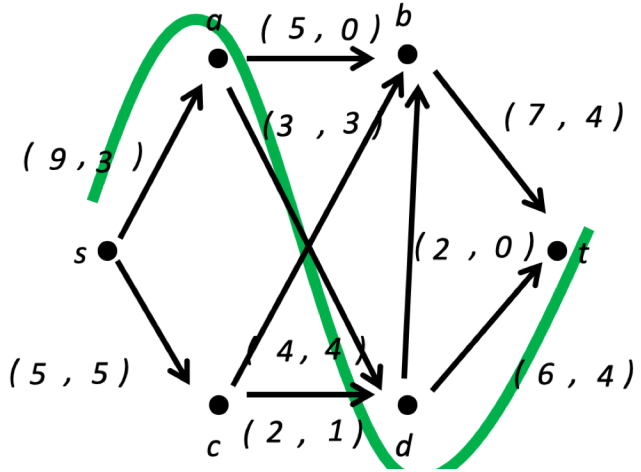




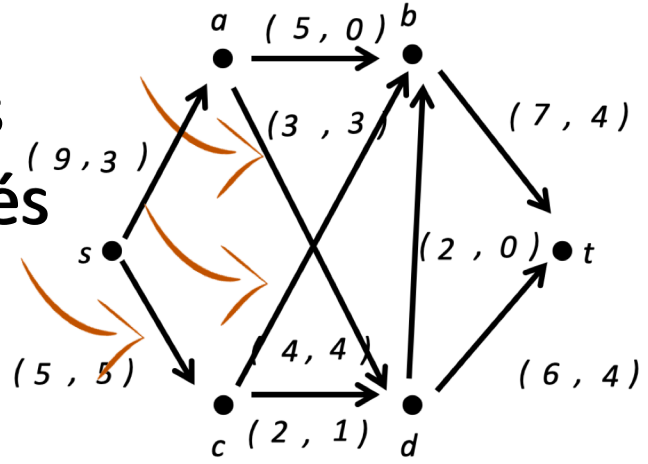
Le chemin s, c, b, t. Mou minimal est 4.
Envoyer un flux de 4 le long du chemin.



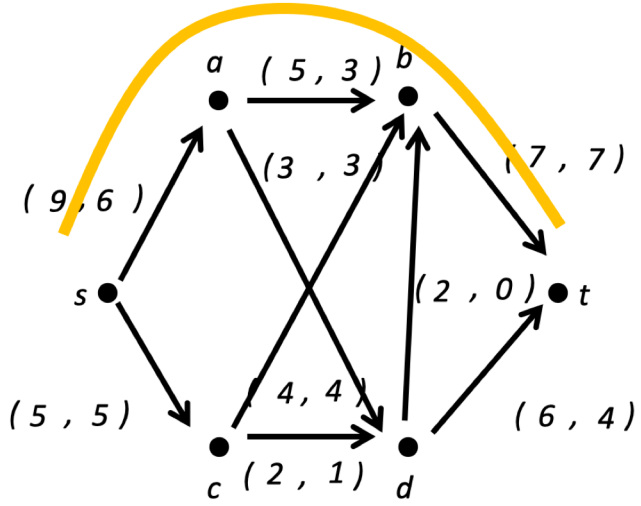
Le chemin s, c, d, t. Mou minimal est 1.
Envoyer un flux de 1 le long du chemin.



Arcs saturés



Le chemin s, a, d, t. Mou minimal est 3.
Envoyer un flux de 3 le long du chemin.

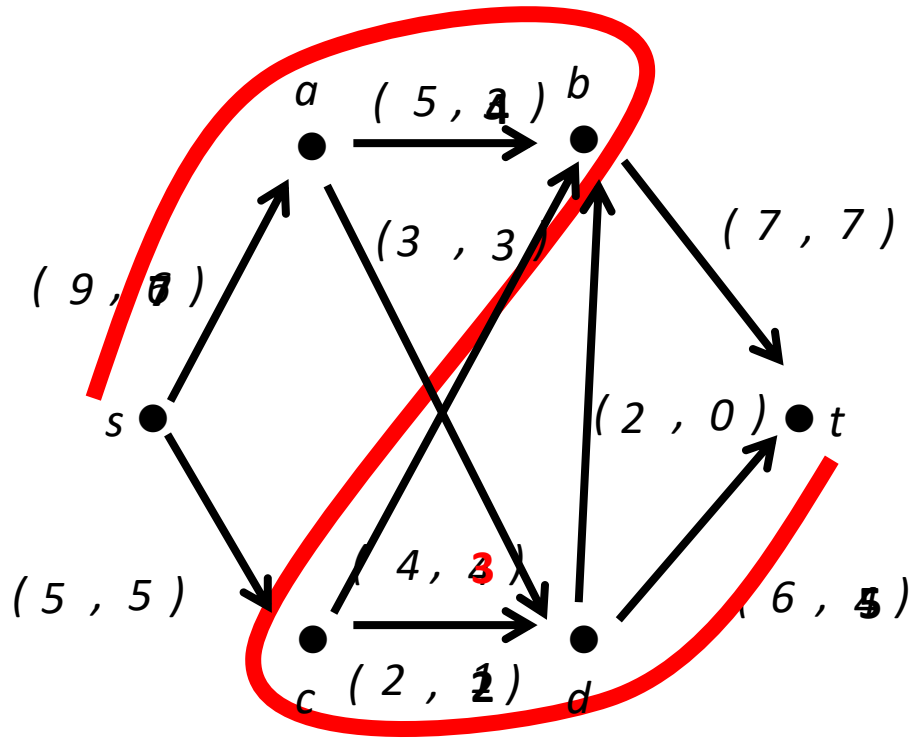


Le chemin s, a, b, t. Mou minimal est 3.
Envoyer un flux de 3 le long du chemin.

Flux réseau

EXEMPLE

Essayez d'obtenir un flot maximal.



Comment pouvons-nous alors augmenter le débit?

Utilisez des demi-trajets (déplacez-vous dans la mauvaise direction !!) en diminuant le débit dans l'arc.

Combien de flux supplémentaire pouvons-nous déplacer le long d'un demi-chemin?

Nous pouvons déplacer le minimum des valeurs de mou pour les arcs avant et les valeurs de flux pour les arcs inverses.

Total flot 12

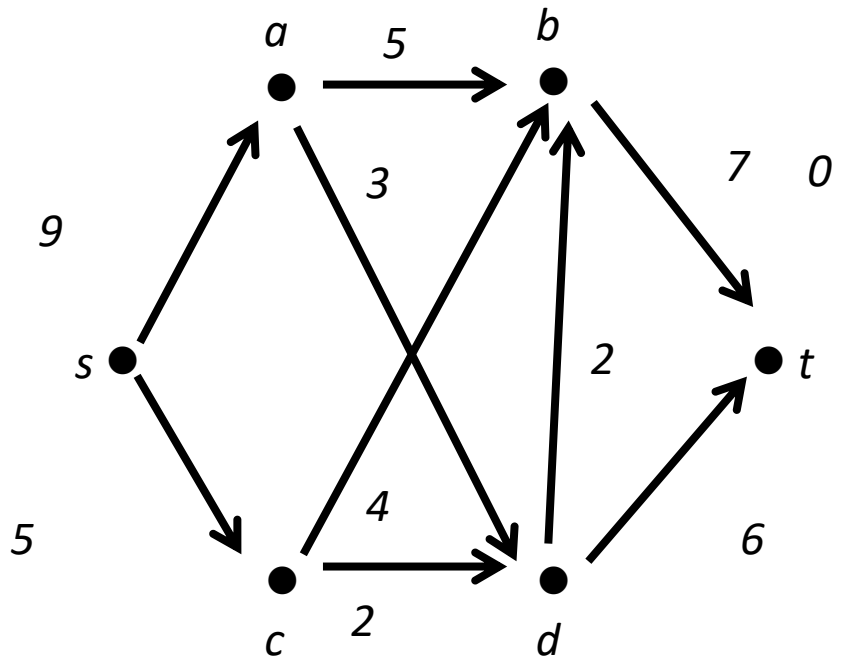
Coupes minimales / Coupes minimum

- Pour atteindre un flot maximal dans un réseau, nous utiliserons un théorème important et puissant dû à **Ford et Fulkerson** appelé le théorème de coupe min - flot max.
- Il utilise le concept d'une coupe minimum.
- Pour un réseau avec source s et un puits t, une **coupe** est un ensemble S d'arcs tels que chaque chemin dirigé de s à t passe par au moins un des arcs de S. Une autre façon de dire cela est que si les arcs de étaient supprimés de N, il n'y aurait pas de chemin dirigé de s vers t.
- La **valeur de la coupe** est la somme des capacités des arcs dans la coupe.
  - Une **coupe minimum** pour un réseau N est une coupe qui a la plus petite valeur de coupe possible.
  - Une **coupe minimale** S est une coupe telle que pour tout $e \in S$, $S - e$ n'est pas une coupe. Ainsi, si un arc de est supprimé, nous n'avons plus de coupe. On peut dire qu'il n'y a pas d'arcs redondants dans une coupe minimale.

Coupes minimales / Coupes minimum

EXEMPLE

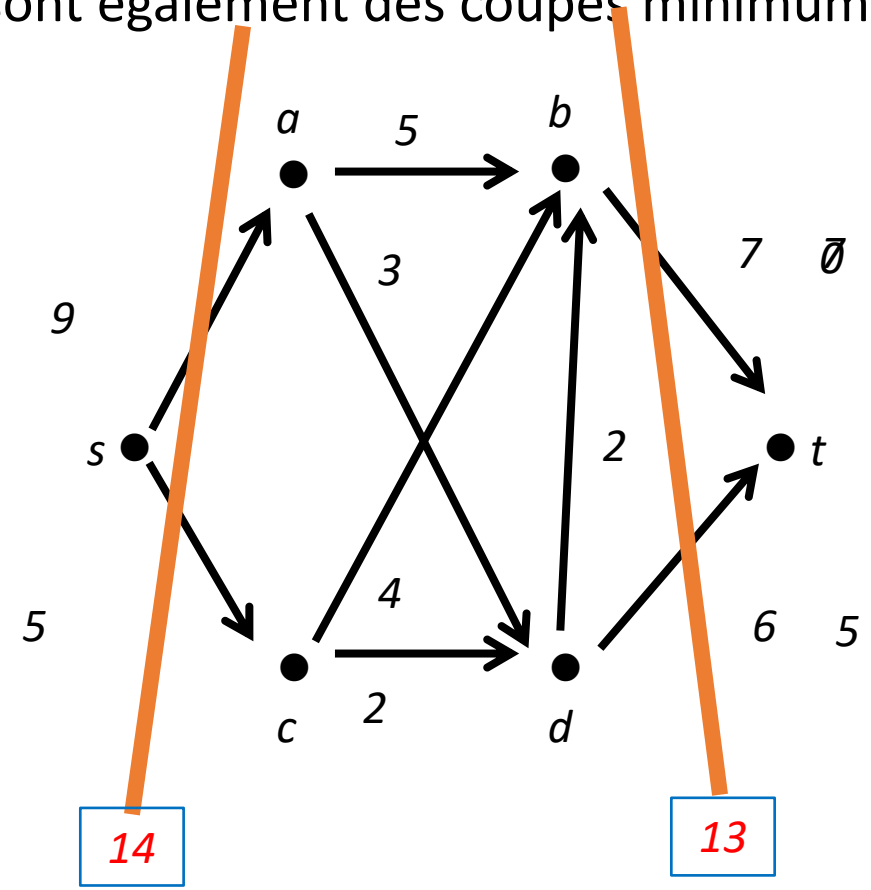
Trouvez toutes les coupes minimales pour le réseau suivant. Lesquelles de ces coupes sont également des coupes minimum?



Coupes minimales / Coupes minimum

EXEMPLE

Trouvez toutes les coupes minimales pour le réseau suivant. Lesquelles de ces coupes sont également des coupes minimum?

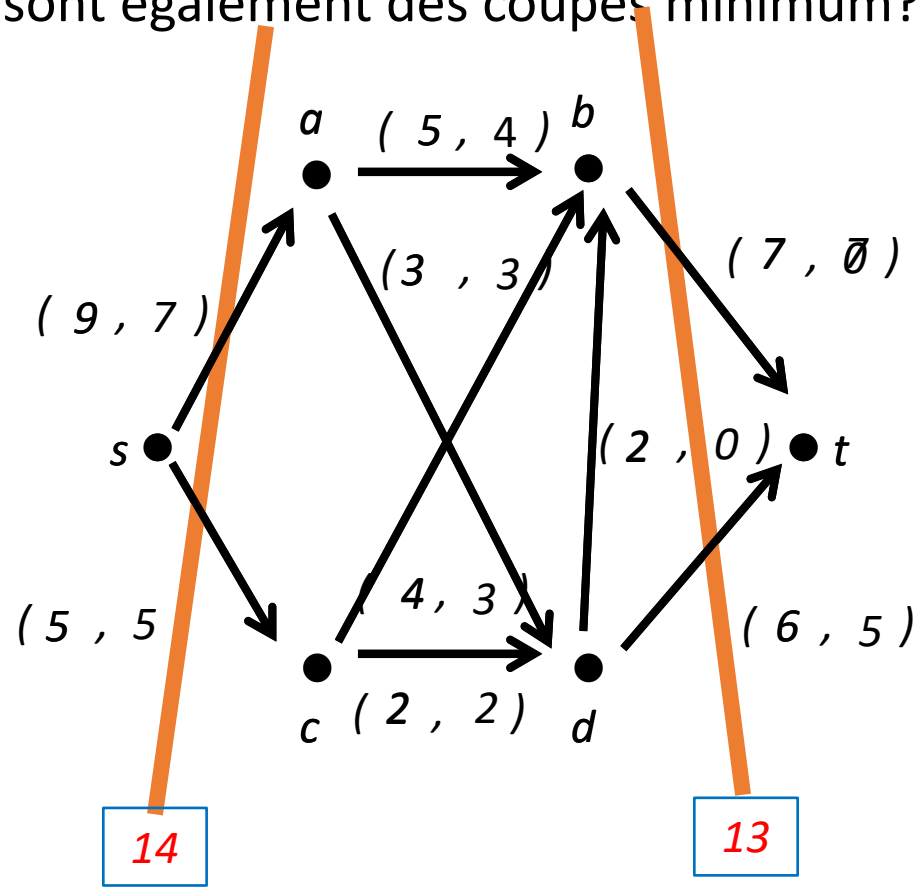


$\{sa, sc\}$	14
$\{bt, dt\}$	13
$\{sa, cb, cd\}$	15
$\{sa, cb, db, dt\}$	21
$\{sa, cd, bt\}$	18
$\{sa, ab, ad\}$	13
$\{sc, ab, db, dt\}$	18
$\{sc, ad, bt\}$	15
$\{ab, ad, cb, cd\}$	14
$\{ab, cb, db, dt\}$	17
$\{ad, cd, bt\}$	12

Coupes minimales / Coupes minimum

EXEMPLE

Trouvez toutes les coupes minimales pour le réseau suivant. Lesquelles de ces coupes sont également des coupes minimum?



$\{sa, sc\}$	14
$\{bt, dt\}$	13
$\{sa, cb, cd\}$	15
$\{sa, cb, db, dt\}$	21
$\{sa, cd, bt\}$	18
$\{sa, ab, ad\}$	13
$\{sc, ab, db, dt\}$	18
$\{sc, ad, bt\}$	15
$\{ab, ad, cb, cd\}$	14
$\{ab, cb, db, dt\}$	17
$\{ad, cd, bt\}$	12

Théorème Coupe-Min Flot-Max

- Toute coupe S divise l'ensemble de sommets de N en deux ensembles X et Y, où $s \in X$ et $t \in Y$. L'ensemble X se compose de tous les sommets pouvant être atteints par s dans $N - S$, et l'ensemble Y se compose de tous les autres sommets. Ainsi, chaque arc de S est incident avec un sommet dans X et un sommet dans Y.
- Soit $\boldsymbol{C(S)}$ la valeur de coupe de S.
- **Théorème** Si F est le flot total dans un réseau N et S est n'importe quelle coupe, alors $F \leq C(S)$.
- **Corollaire** Pour tout réseau N, le flot maximum est inférieur ou égal à la valeur de la coupe minimum.

Ford et Fulkerson ont montré que:

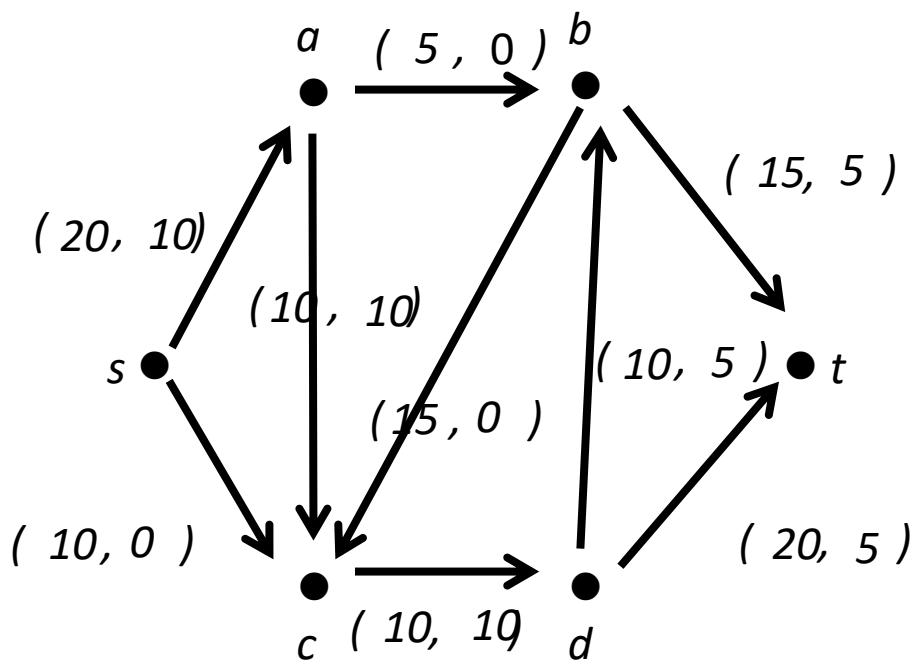
- **Théorème** (**Max-Flow Min-Cut Theorem**) Pour tout réseau N, la valeur d'un flot maximum en N est égale à la valeur de coupe d'une coupe minimum.

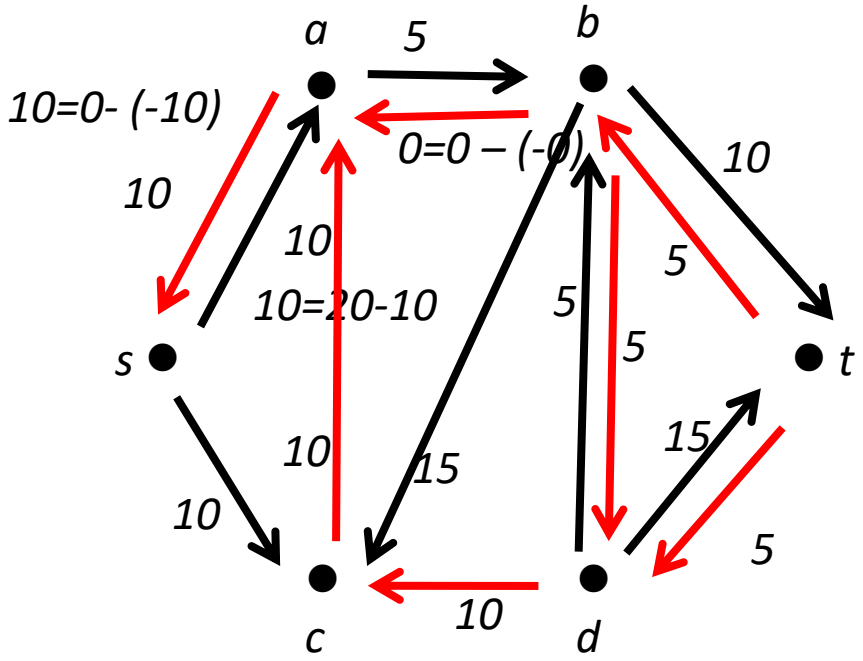
Graphe résiduel

- Étant donné un flux f dans le graphe G, le graphe résiduel G_f est le graphe orienté avec toutes les arêtes de mou strictement positif ($sl(e) = c(e) - f(e)$), chacun marqué par son mou.
- Remarque: cela peut inclure les arêtes arrière du graphe d'origine G. Si, $uv \notin G$, $c(uv) = 0$ mais $f(uv) = -f(vu)$

Graphe résiduel

$$sl(e) = c(e) - f(e)$$



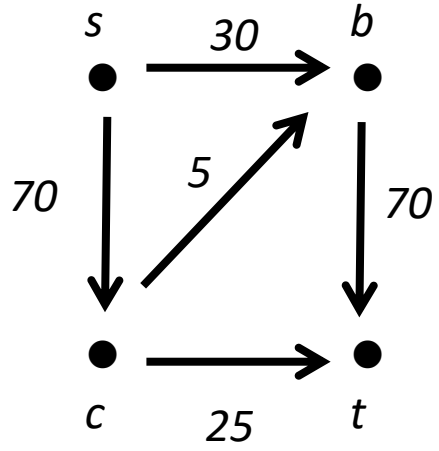


Méthode de Ford Fulkerson

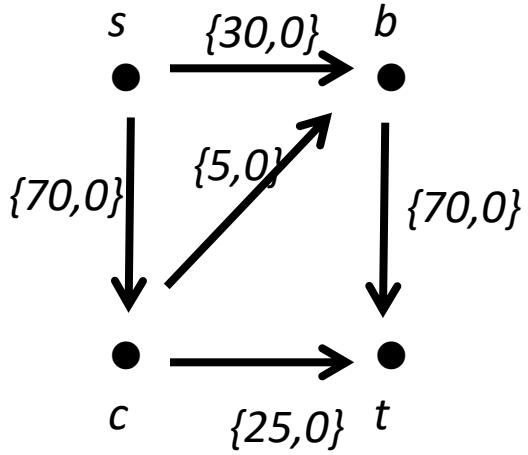
- Algorithme itératif simple: envoie un flux f sur un chemin p chaque fois qu'il existe un **chemin augmentant** p de s vers t.
- Chemin augmentant p *:* un chemin de la source s au puits t qui passe par des arêtes pondérées positives dans le **graphe résiduel**.

Méthode de Ford Fulkerson
Exemple d'exécution

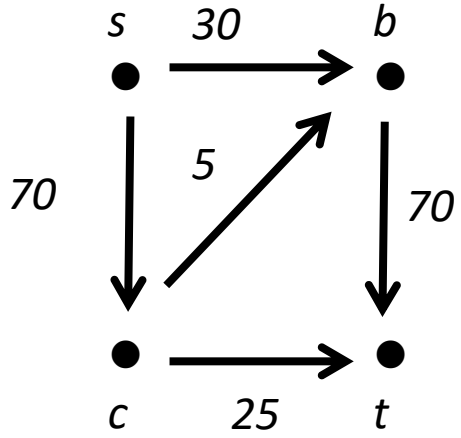
chaque arête a le même poids que le graphe d'origine



graph G



Flot f
Commencez par 0

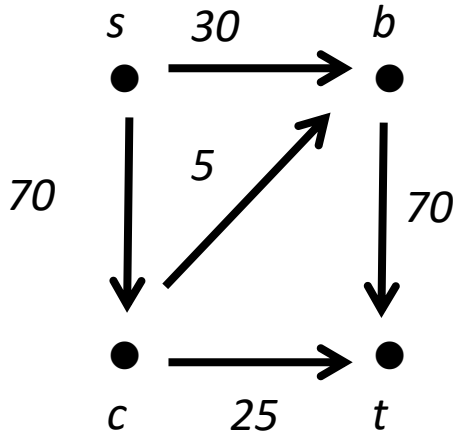


graphe résiduel

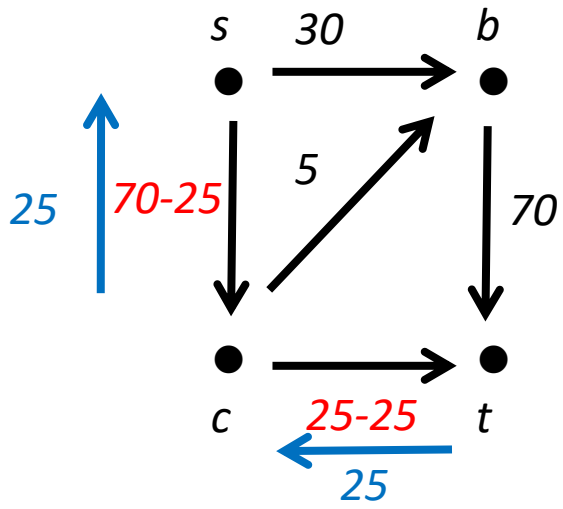
Méthode de Ford Fulkerson
Exemple d'exécution

chemin augmentant: s → c → t

Flot = 25



graphe résiduel

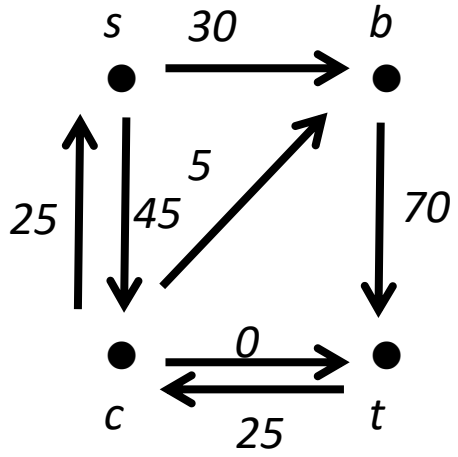


nouveau graphe résiduel

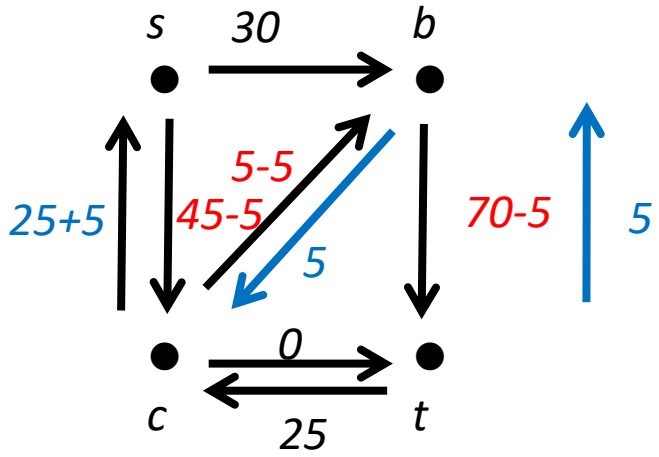
Méthode de Ford Fulkerson
Exemple d'exécution

chemin augmentant : s → c → b→ t

Flot = 5



graphe résiduel

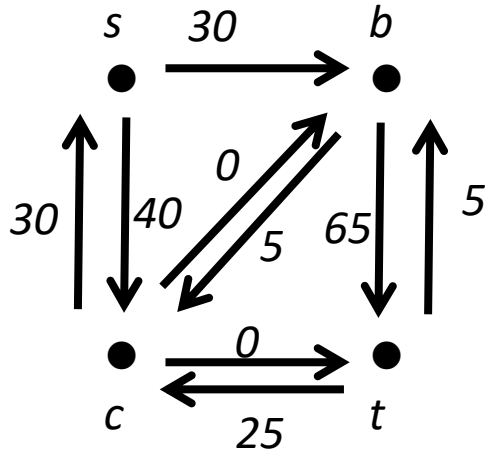


nouveau graphe résiduel

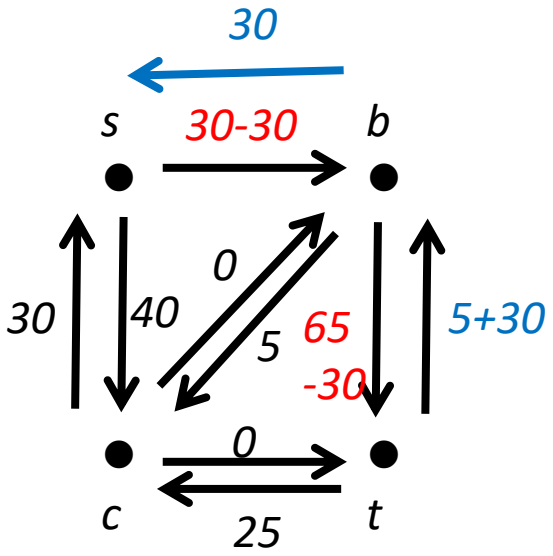
Méthode de Ford Fulkerson
Exemple d'exécution

chemin augmentant : s → b → t

Flot = 30

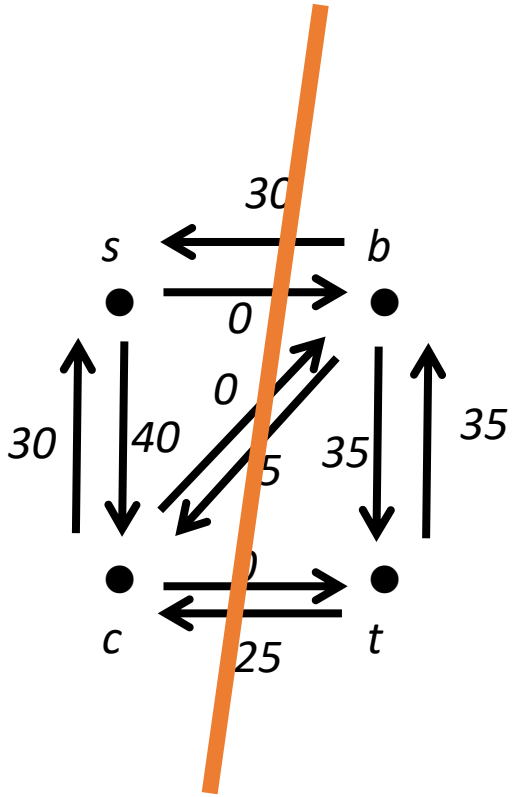


graphe résiduel



nouveau graphe résiduel

Méthode de Ford Fulkerson
Exemple d'exécution



graphe résiduel

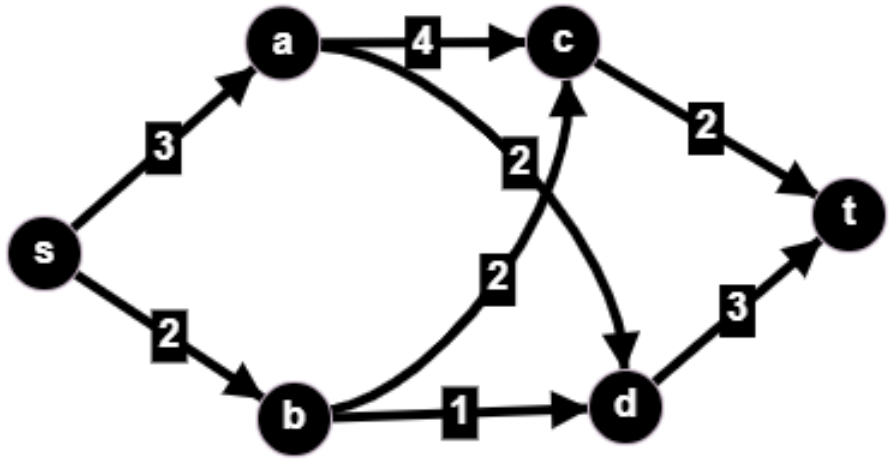
plus de chemins augmentants

Flot Max Total = somme des flots précédents

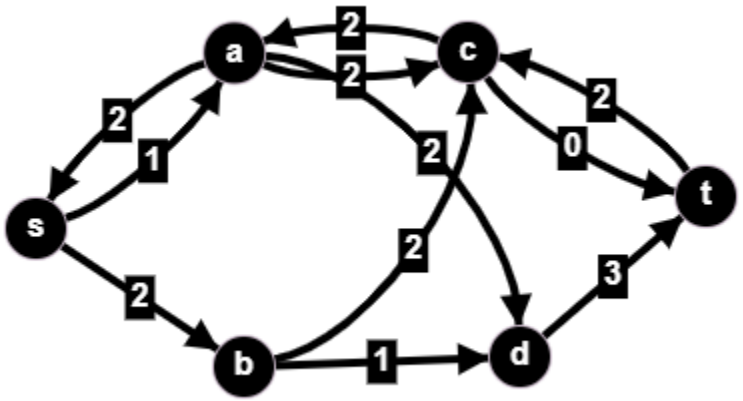
25 + 5 + 30 = 60

Digraphes et réseaux
Question 4

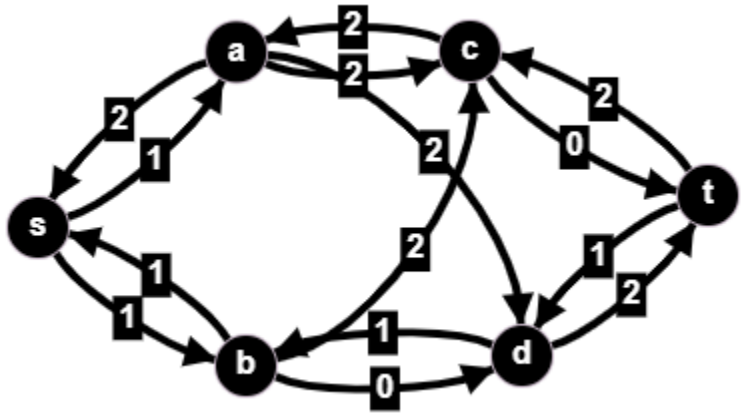
Utilisez la méthode Ford-Fulkerson pour obtenir un flot maximum de s à t.



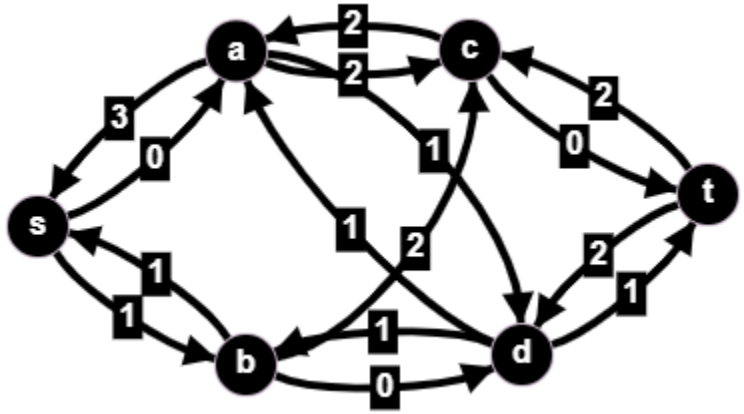
sact , flot =2



sbdt , flot =1

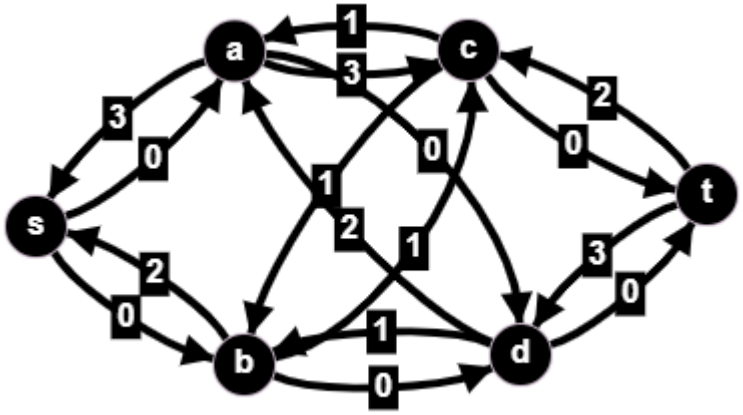


sadt , flot =1



SOLUTION

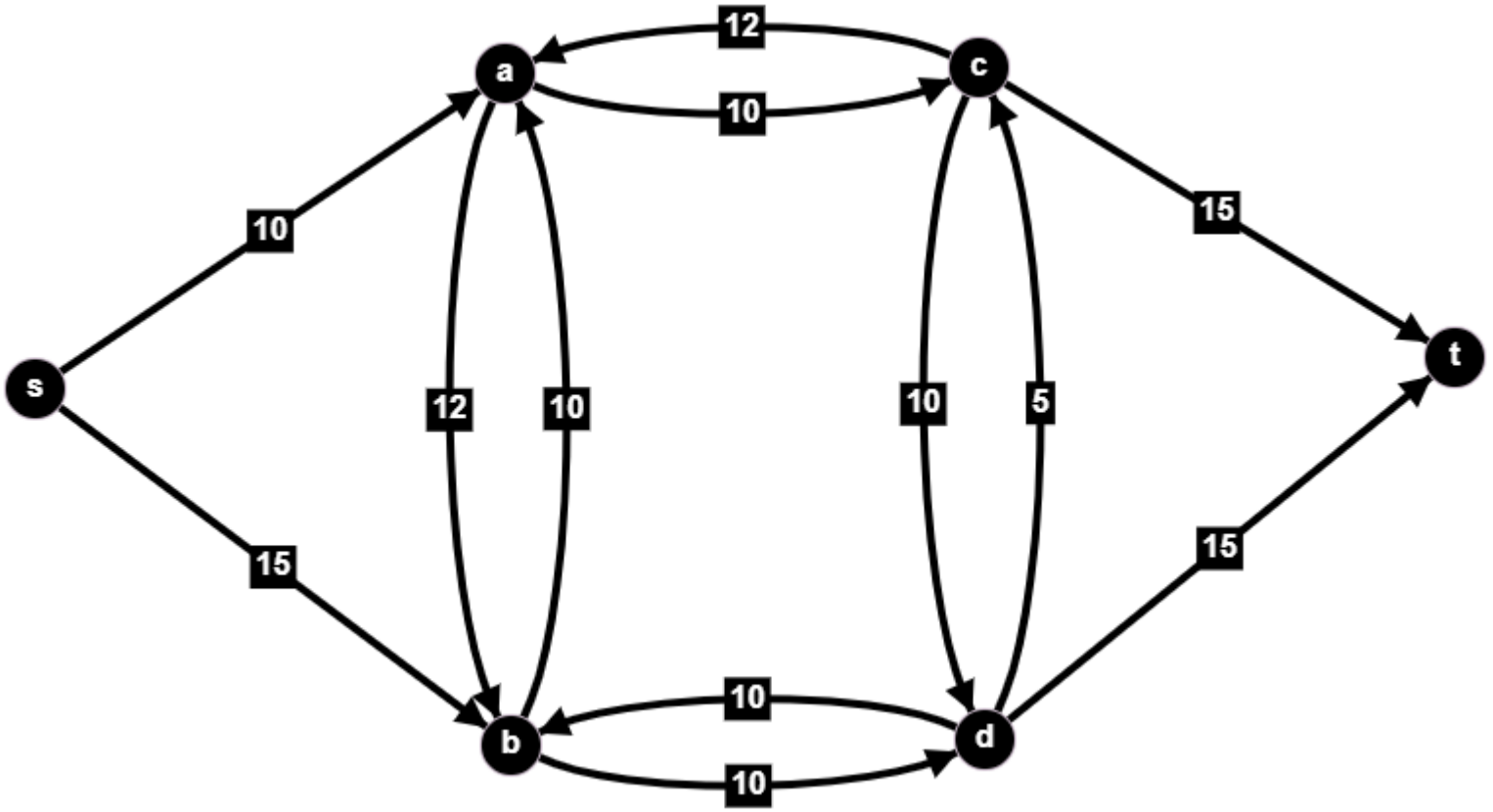
sbcadt , flot =1



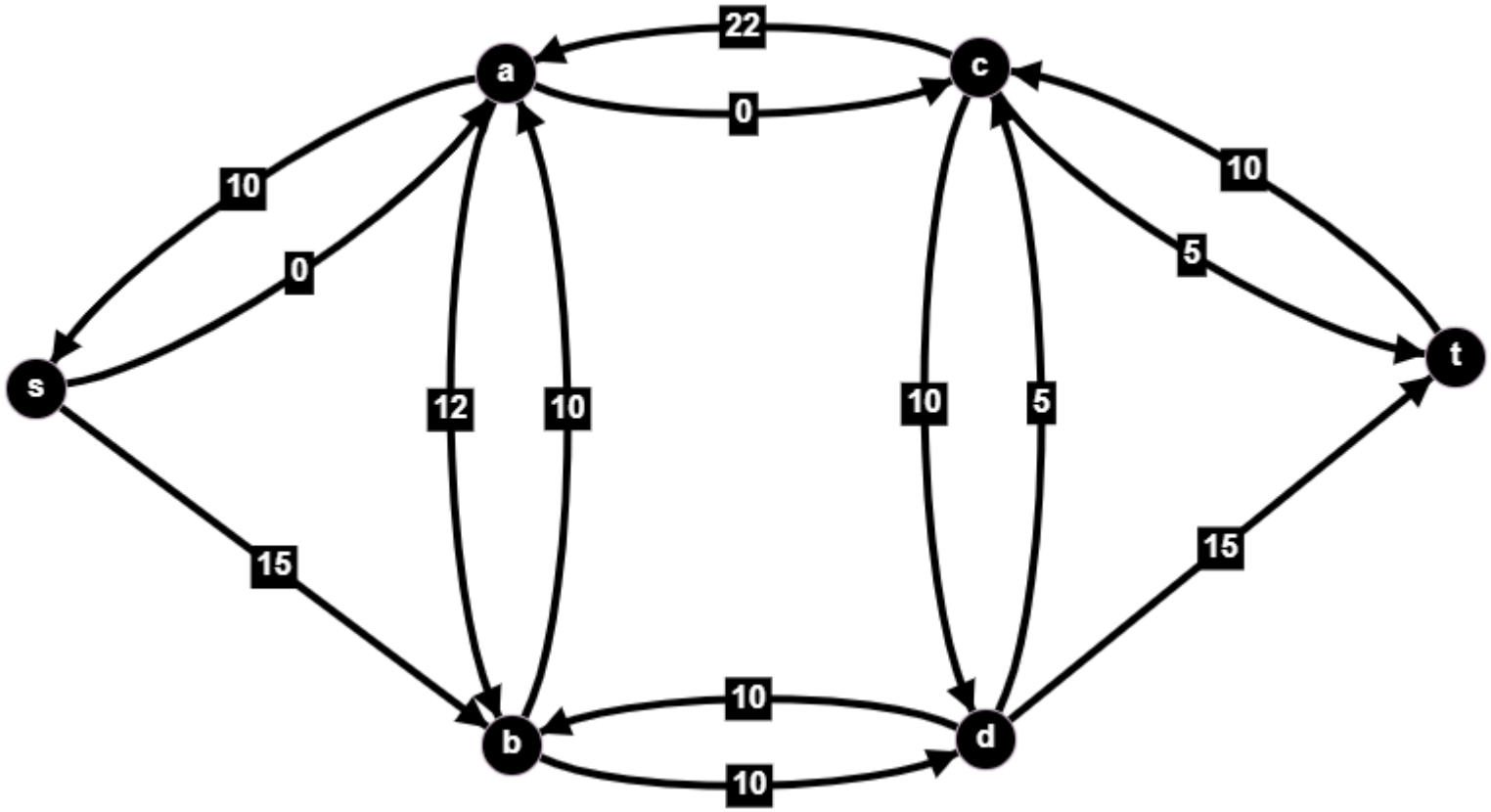
Total flot =5

Digraphes et réseaux
Question 5

Utilisez la méthode Ford-Fulkerson pour obtenir un flot maximum de s à t.

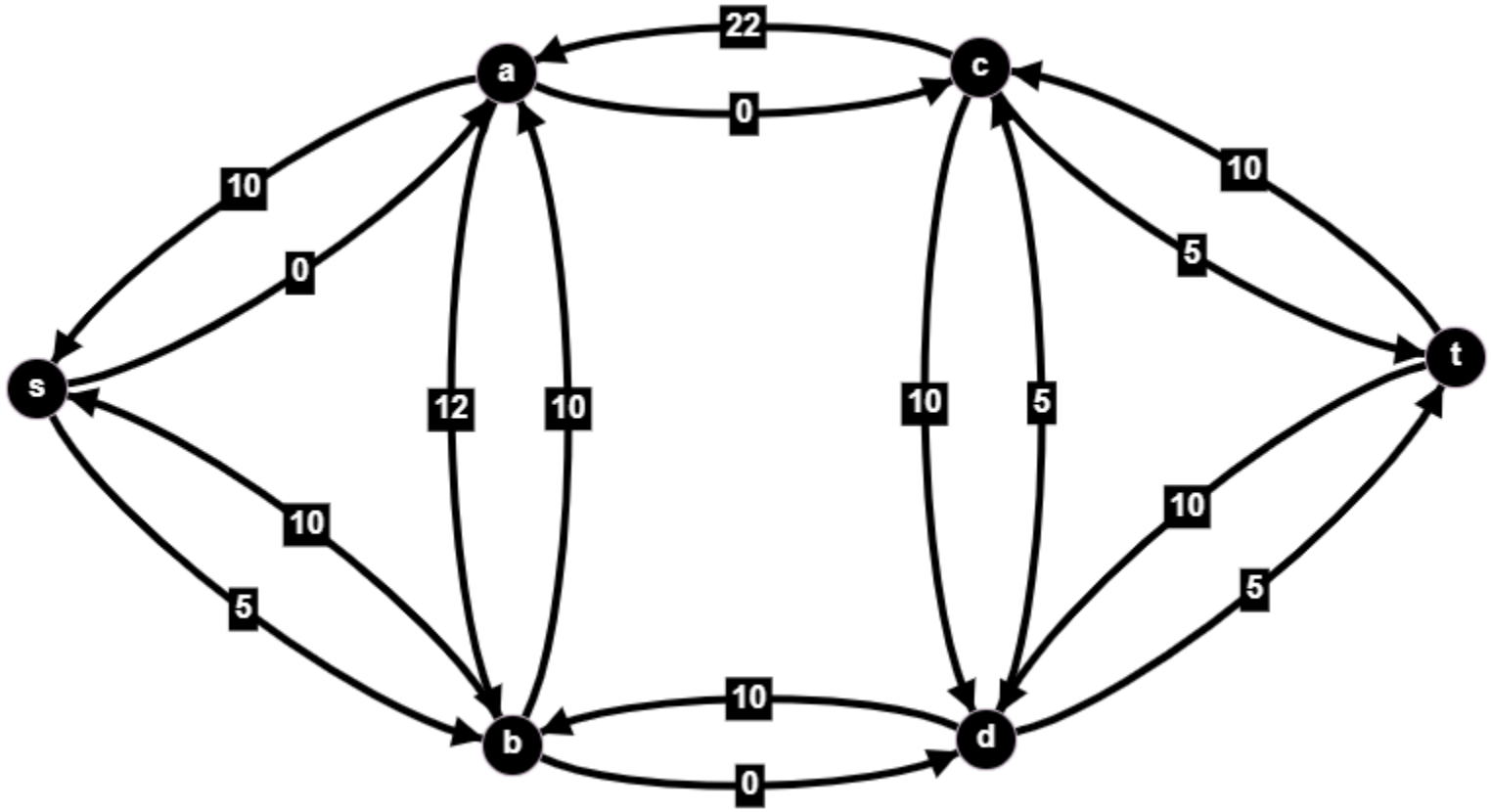


sact , flot =10



SOLUTION

sbdt , flot =10

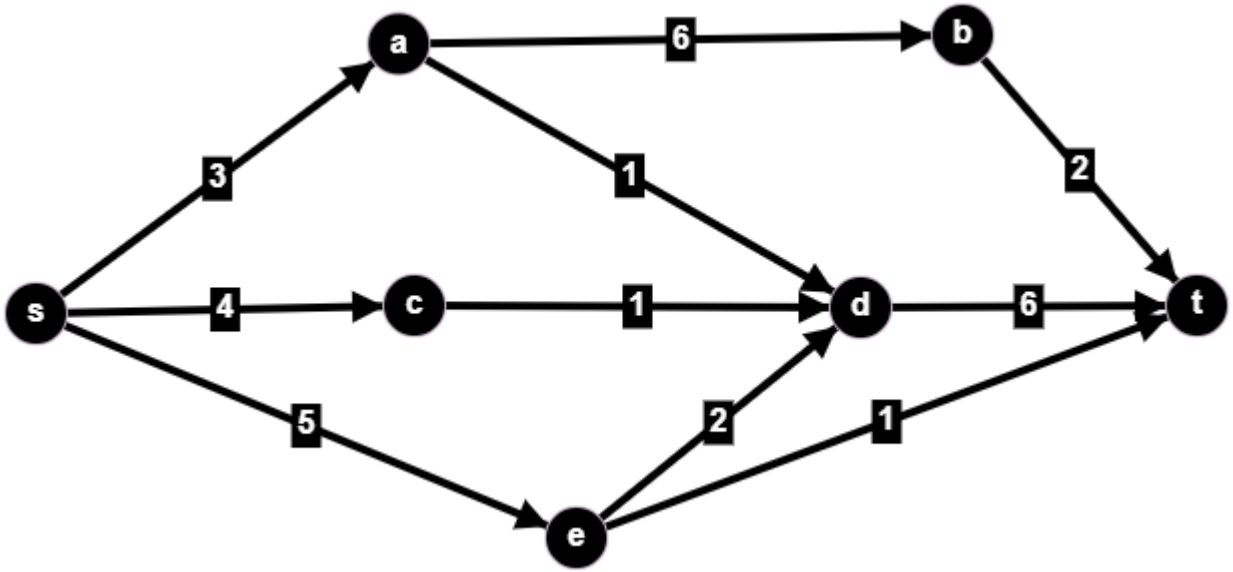


SOLUTION

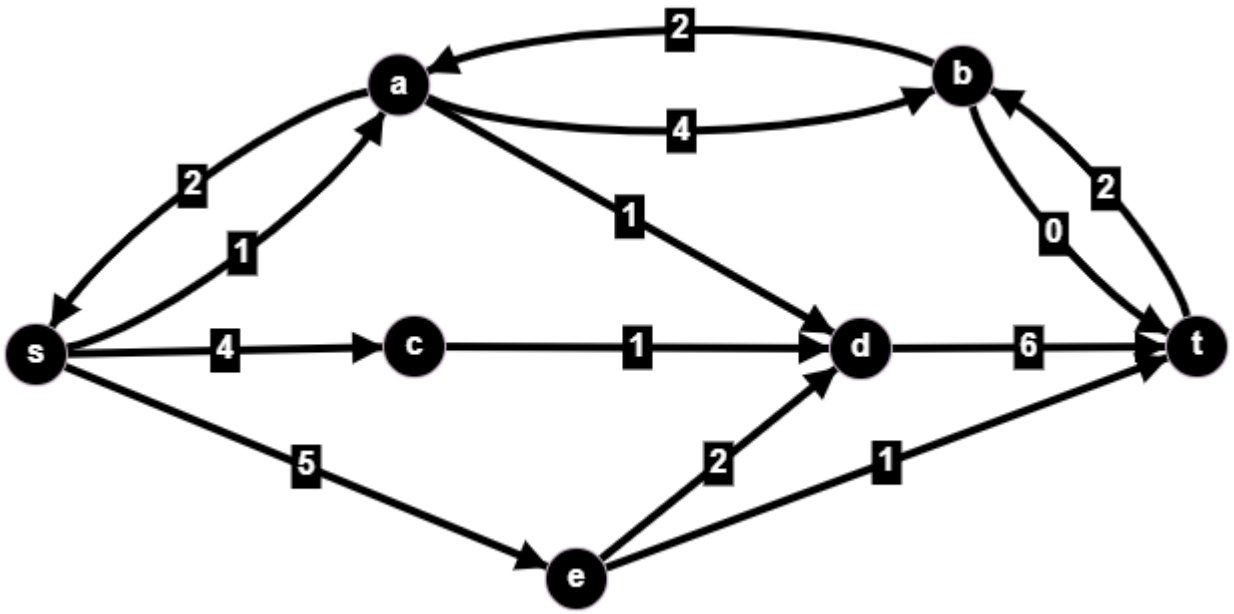
Total flot =20

Digraphes et réseaux
Question 6

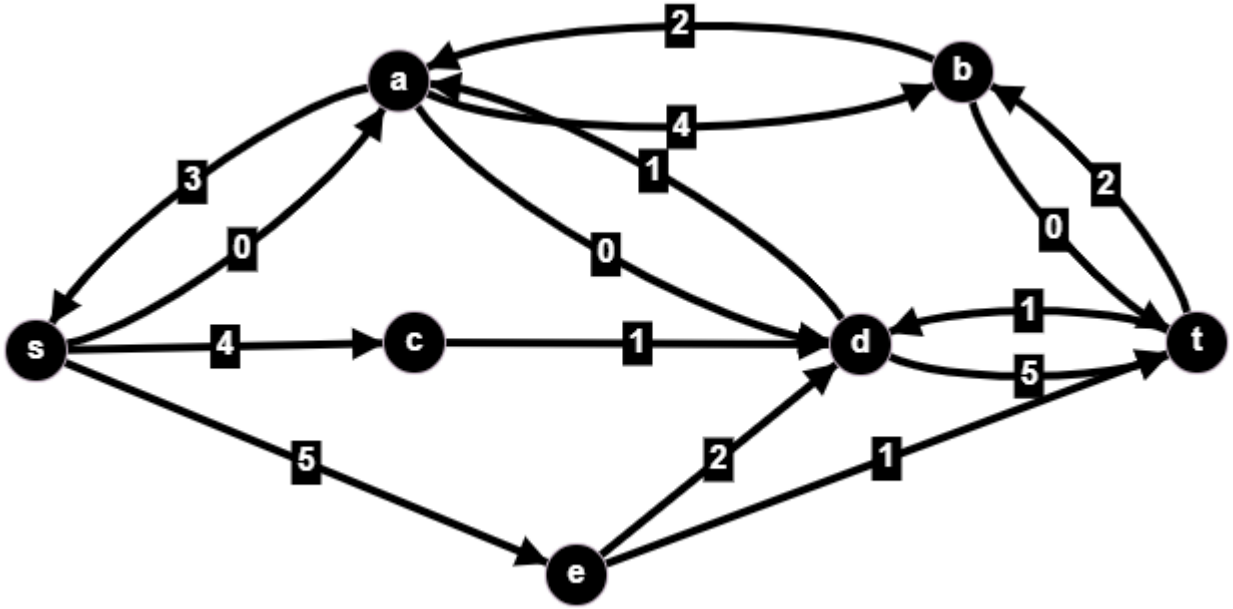
Utilisez la méthode Ford-Fulkerson pour obtenir un flot maximum de s à t.



sabt , flot =2

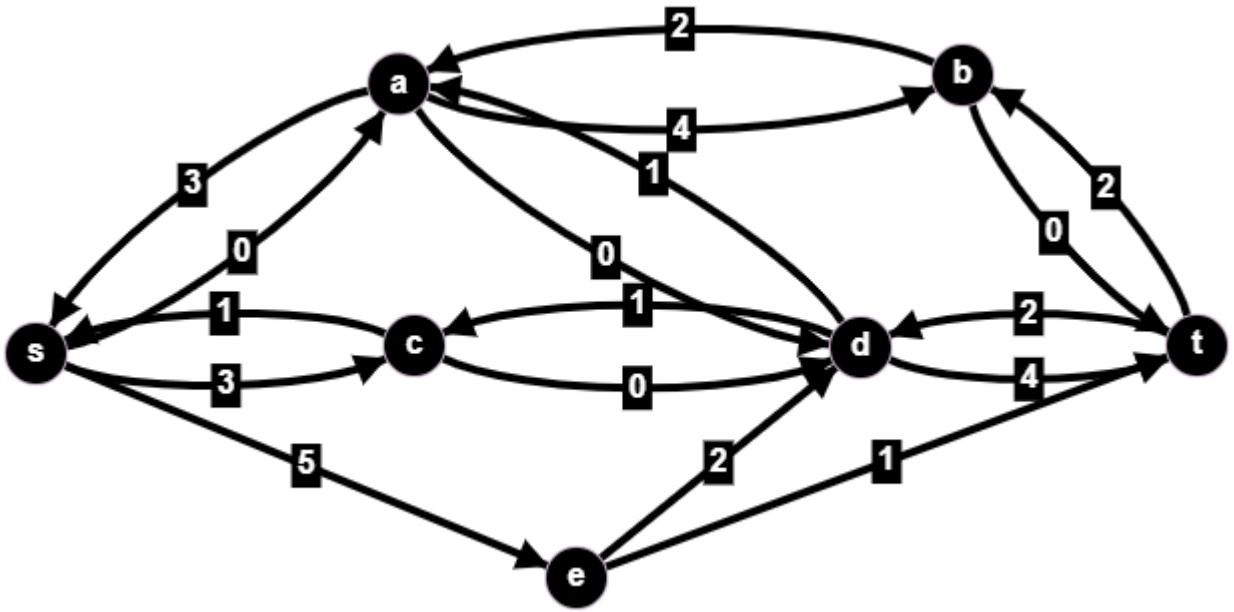


sadt , flot =1

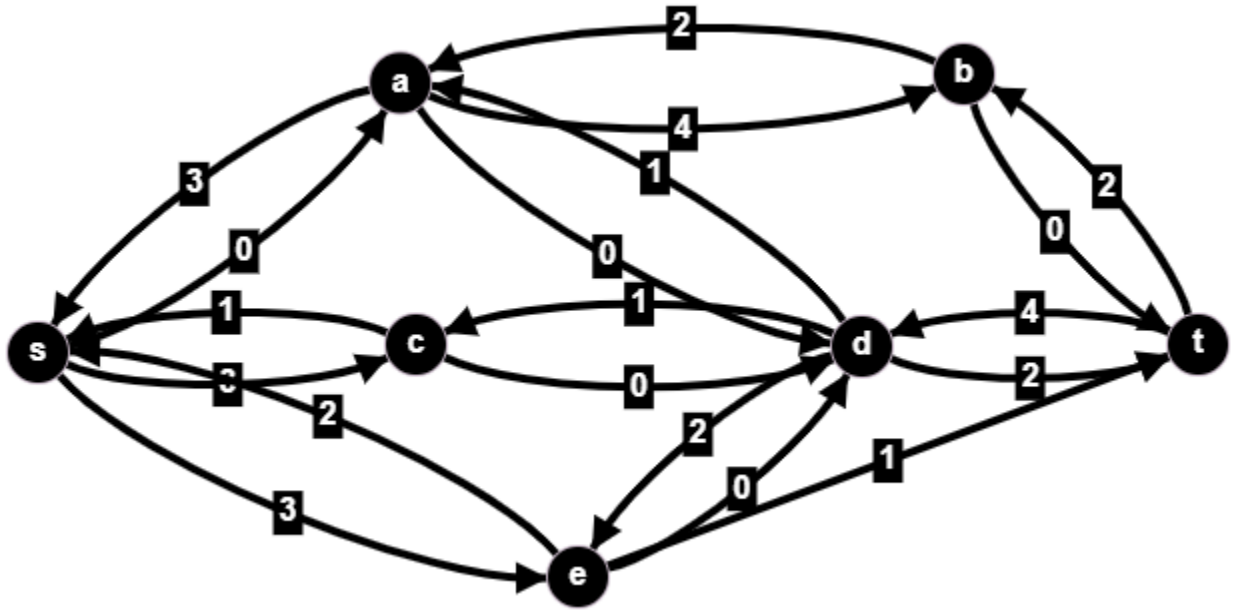


SOLUTION

scdt , flot =1

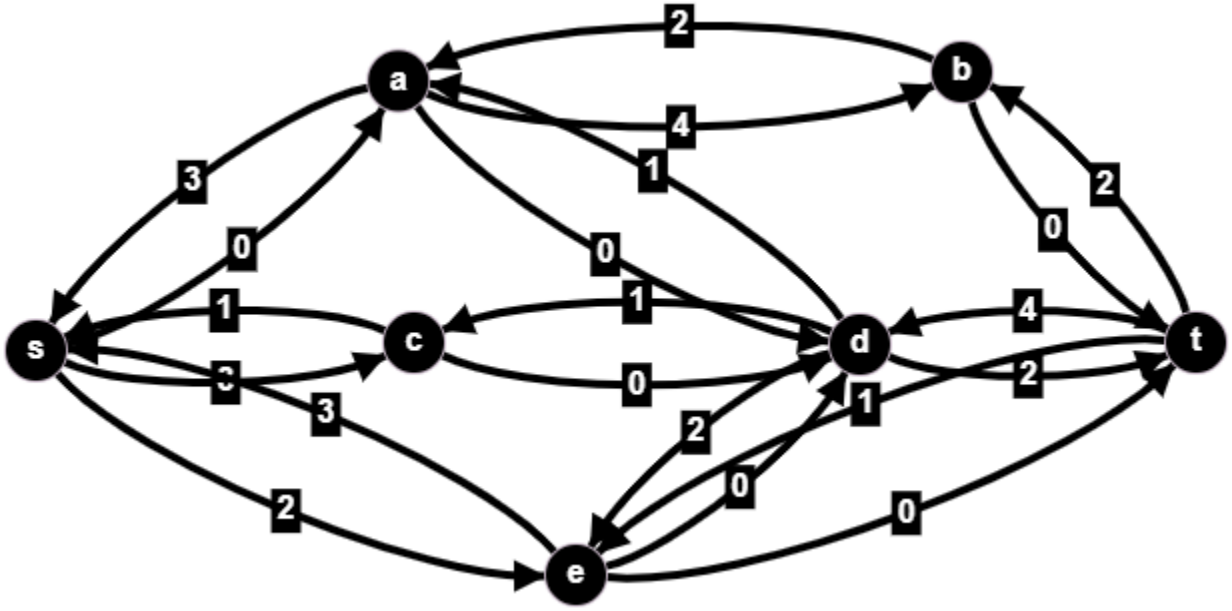


sedt , flot =2



SOLUTION

set , flot =1



SOLUTION

Total flot =7

Plan

- Graphes orientés
- Réseaux
  - Distance dans les réseaux
    - Algorithme de Dijkstra
    - Algorithme Bellman-Ford
  - Flux de réseau
    - Algorithme Ford Fulkerson
    - Algorithme d'Edmonds – Karp
  - Réseaux et appariements
- La méthode du chemin critique

Recherche d'un semi-trajet augmentant le flot

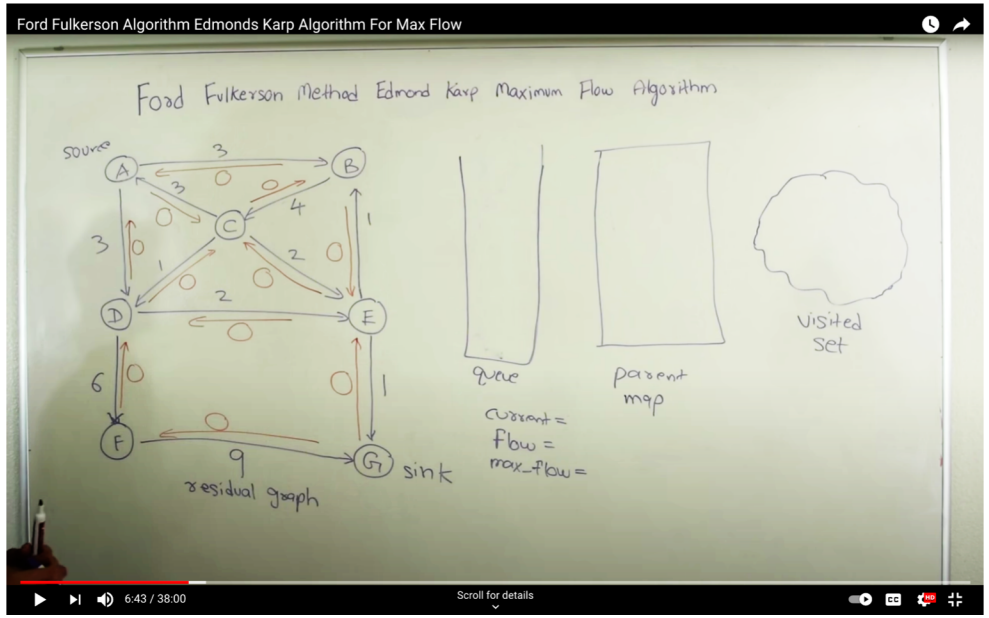
- L'algorithme original de Ford et Fulkerson a géré sans problème des réseaux avec des capacités entières ou même rationnelles. Cependant, ils ont montré que si les capacités étaient irrationnelles, c'est-à-dire dans $\mathbb{R} - \mathbb{Q}$, que leur algorithme pourrait prendre un nombre infini d'étapes.
- *Edmond et Karp ont* corrigé le problème en utilisant BFS.
  - Spécialisation de l'algorithme de Ford-Fulkerson, où les chemins augmentants sont des plus courts chemins (en nombre d'arcs) dans le graphe résiduel (on utilise un parcours en largeur)

Algorithme d'Edmond et Karp

- Créer le graphe résiduel dirigé
- Réglez le débit maximum sur 0.
- tant qu'il existe un chemin augmentant p de s à t de la forme ($s \rightarrow$ … $\rightarrow i \rightarrow j \rightarrow \cdots t$)
  {Ici la méthode de recherche de p est améliorée en utilisant BFS.}
  1. Trouvez f le poids minimal d'arête le long du chemin p
  2. Diminuez le poids des arêtes avant (par exemple $i \rightarrow j$) le long du chemin p de f
  3. Augmentez le poids des arêtes arrière (par exemple $j \rightarrow i$) le long du chemin p de f
  4. Incrementer le flot maximum de f
- Afficher le flot maximum

Algorithme d'Edmond et Karp
Recherche du chemin augmentant

- Exemple 2: sur Wikipedia
  - https://fr.wikipedia.org/wiki/Algorithme_d%27Edmonds-Karp

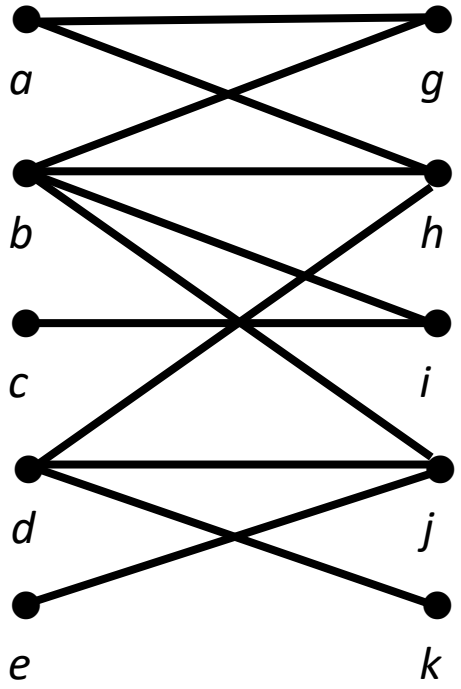


Plan

- Graphes orientés
- Réseaux
  - Distance dans les réseaux
    - Algorithme de Dijkstra
    - Algorithme Bellman-Ford
  - Flux de réseau
    - Algorithme Ford Fulkerson
    - Algorithme d'Edmonds – Karp
  - Réseaux et appariements
- La méthode du chemin critique

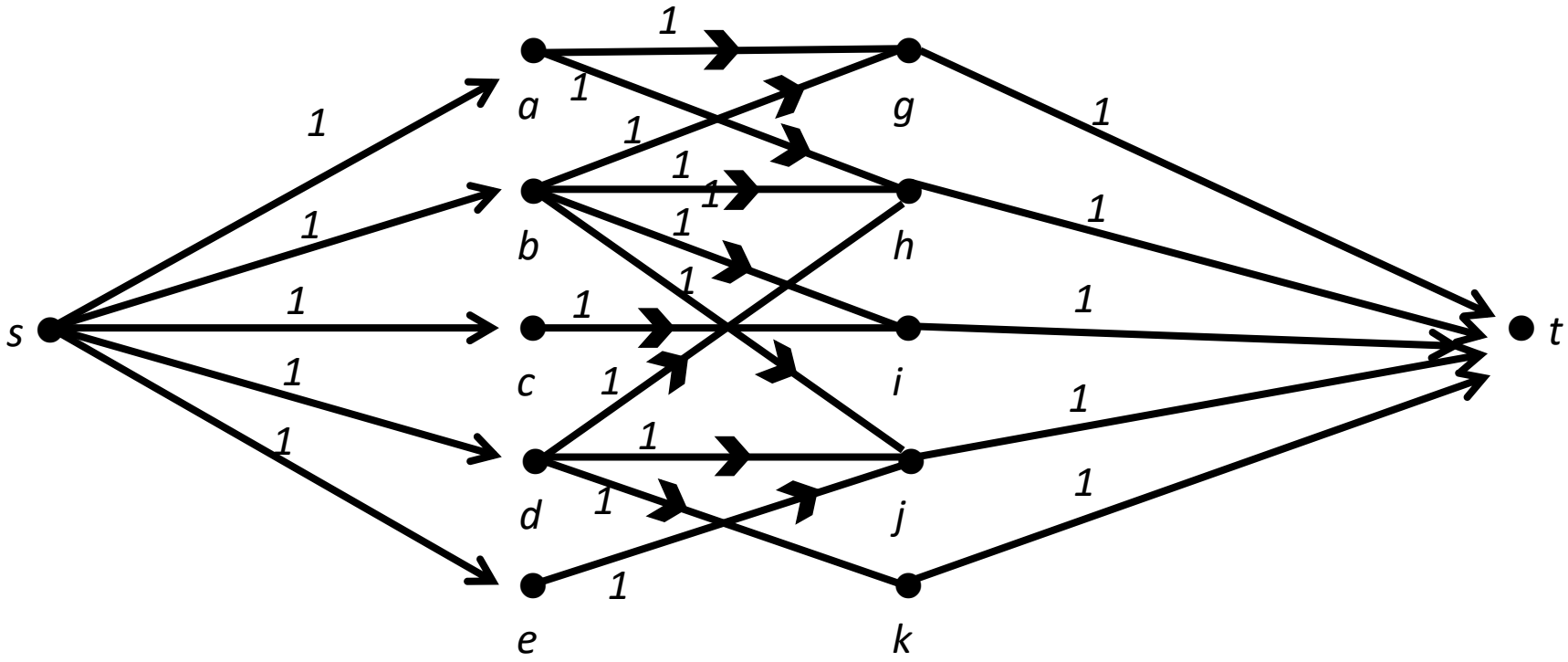
Réseaux et appariements

- Les flux réseau peuvent être utilisés pour résoudre facilement des problèmes d'appariement bipartis.



Réseaux et appariements

- Les flux réseau peuvent être utilisés pour résoudre facilement des problèmes d'appariement bipartis.



Réseaux et appariements

- Étant donné un graphe biparti G avec des parties A et B, supposons que nous souhaitons obtenir une correspondance maximale.
- Nous pouvons le faire comme suit:
  - Diriger tous les bords de A à B.
  - Créez un réseau N à partir de G en introduisant deux nouveaux sommets s et t.
  - Placez les arcs de s à chaque sommet et placez les arcs de chaque sommet vers t.
  - Ensuite, mettez la capacité 1 sur chaque arc.
  - Maintenant, appliquez l'algorithme.
  - Lorsqu'un flot maximal est trouvé, les arcs saturés de A à B correspondent à une correspondance maximale en G.

Plan

- Graphes orientés
- Réseaux
- La méthode du chemin critique

La méthode du chemin critique

- Chaque grand projet se compose de nombreuses activités qui doivent être planifiées pour se dérouler dans un certain ordre pour une réalisation rapide et réussie du projet.
- Le chef de projet fait des estimations de la durée de chaque activité individuelle et détermine une relation de priorité pour les activités.
- Par exemple, sur un projet de construction, nous installons généralement le câblage électrique avant d'installer le panneau mural. Les durées d'activité et les relations de précédence entre les activités peuvent être modélisées à l'aide d'un certain **digraphe acyclique pondéré**. À l'aide de ce digraphe, le responsable peut déterminer le calendrier approprié pour les activités.

Digraphes d'activité

- Supposons que vous envisagiez d'ouvrir un réfrigérateur en libre-service. Les différentes activités que vous devez réaliser au cours du projet sont décrites dans le tableau suivant.

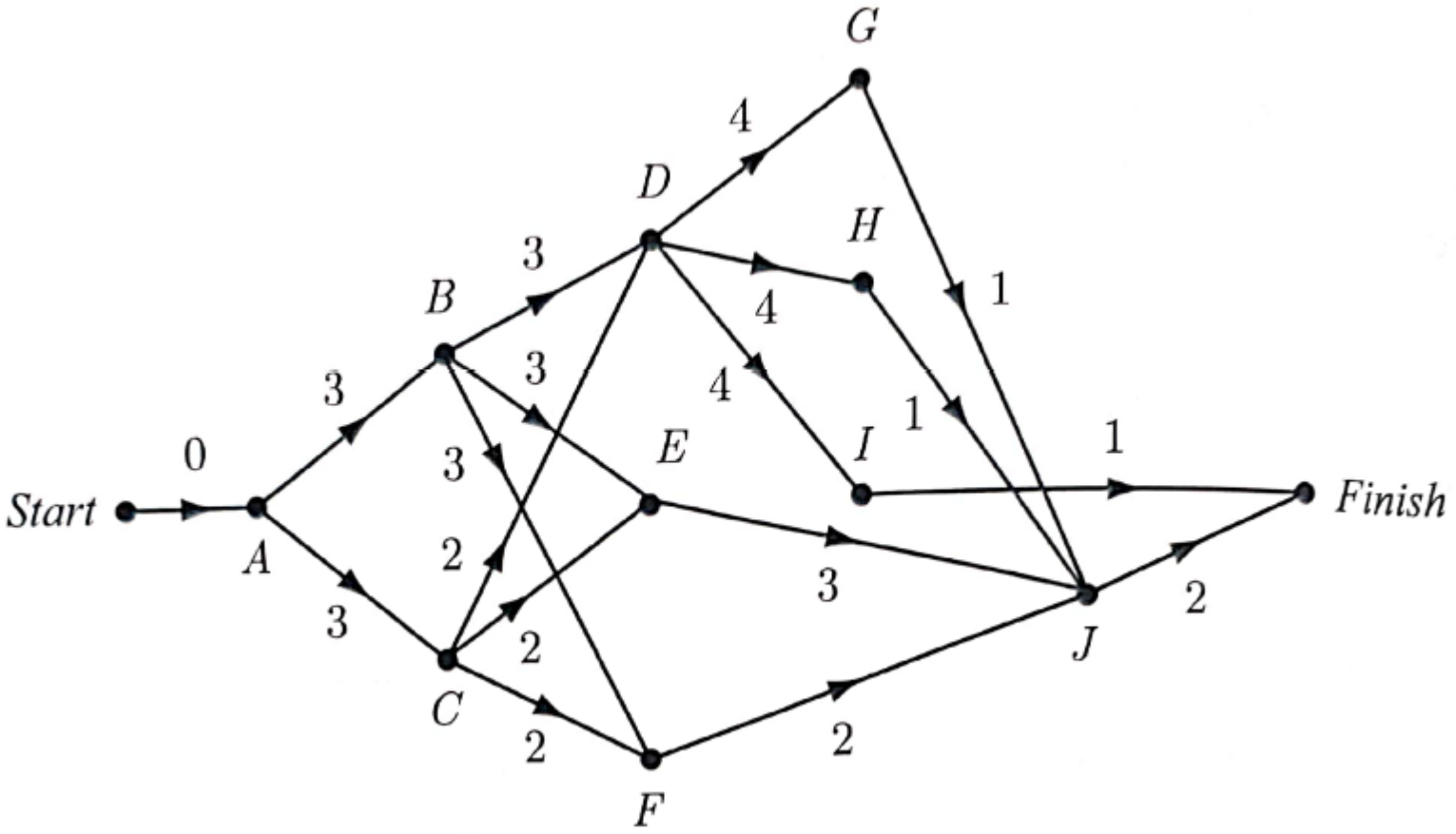
Activité	Description	Prédécesseurs immédiats	Durée en jours
Start	-	-	
A	Rencontre avec un agent immobilier	Start	3
B	Rencontrer un avocat	A	3
C	Rencontre avec un comptable	A	2
D	Obtenir une hypothèque auprès de la banque	B, C	4
E	Obtenir des réfrigérateurs	B, C	3
F	Obtenir des distributeurs automatiques	B, C	2
G	Branchez l'électricité	D	1
H	Brancher le gaz	D	1
I	Obtenir une licence de ville	D	1
J	Configurer l'équipement	E, F, G, H	2
Finish		I, J	0

Digraphes d'activité

- Le **digraphe d'activité** est construit comme suit. Chaque activité correspond à un sommet. Il y a un arc de sommet i en sommet j si l'activité i est un **prédécesseur immédiat** de l'activité j. Dans ce cas, le sommet est appelé un **successeur** du sommet. Notez que $start$ n'a pas de prédécesseurs immédiats et $finish$ n'a pas de successeurs. Un arc de i à j a un poids égal à la durée de l'activité. Nous devons mentionner qu'un digraphe d'activité est en fait un **réseau** , mais que le terme *digraphe d'*activité est la terminologie standard.

Digraphes d'activité

- Le digraphe d'activité du projet est:



Digraphes d'activité

- Pour la colonne "durée en jours " du tableau, les nombres totalisent 22, ce qui signifie que le projet pourrait certainement être achevé en 22 jours.
- Cependant, comme certaines activités peuvent être gérées simultanément, le projet peut être terminé un peu plus rapidement que cela.
- Pour achever le projet le plus rapidement possible, nous avons besoin de savoir quel est **le moment le plus tôt possible, que chaque activité peut commencer et se terminer**.
- Pour plus de flexibilité dans la planification, il est également utile de savoir quelle est **la dernière date à laquelle une activité donnée peut démarrer sans pour autant retarder le projet**.
- Certaines activités sont généralement **critiques** en ce sens que si elles commencent tard, l'achèvement du projet sera retardé.

La méthode du chemin critique
Critical Path Method (CPM)

1. Identifiez les activités, leurs prédécesseurs immédiats et leur durée.
2. Dessinez le digraphe d'activité.
3. Utilisez le digraphe d'activité pour déterminer la **première heure de début** (**earliest start time**) (EST) et la **première heure de fin** (**earliest finish time**) (EFT) pour chaque activité.
4. Calculez le temps mort pour chaque activité.
5. Identifiez toutes les activités critiques et les chemins critiques.

La méthode du chemin critique
Calcul de EST et EFT

1. $EST(start) = 0$
2. Pour $v \neq start$,
   $EST(v) = \max\{EFT(w) : w$ est un prédécesseur immédiat de $v\}$.
3. $EFT(u) = EST(u) +$ durée(u) for all u.
4. $EFT(finish) =$ **temps d'achèvement (completion time)** du projet.

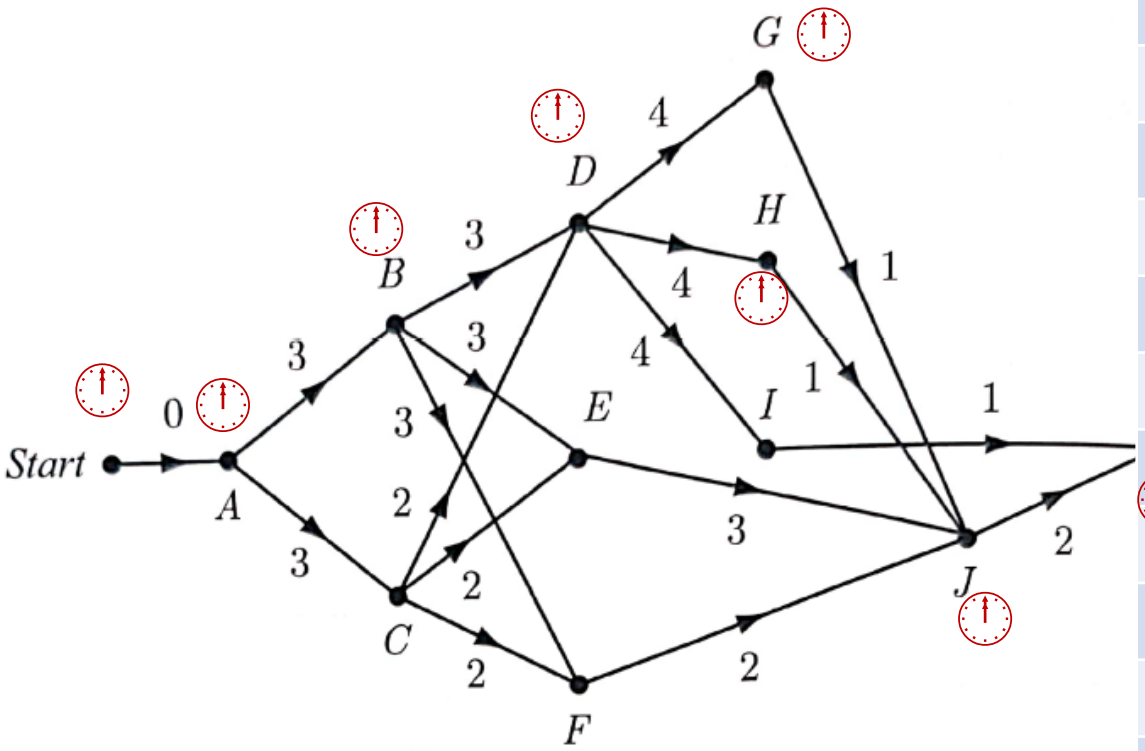
La méthode du chemin critique
Calcul de LST et LFT

1. Déterminez le temps d'achèvement du projet.
2. $LFT(finish) =$ temps d'achèvement.
3. Pour $v \neq finish$,
   $LFT(v) = \min\{LST(w) : w$ est le successeur de $v\}$.
4. $LST(u) = LFT(u) -$ durée(u) pour tout u.

La méthode du chemin critique

- Le **temps mort** (**slack time**)d'une activité est la quantité dont nous pouvons retarder le début de cette activité sans affecter le temps d'achèvement du projet. Il se calcule facilement comme suit: $slack{-}time\ (v) = LST\ (v)$ - $EST\ (v)$.
- Une **activité critique** est une activité dont le temps mort est nul. Tout retard dans le démarrage d'une telle activité retardera l'achèvement du projet. D'un autre côté, un projet avec un temps mort positif n'est pas critique car son heure de démarrage est quelque peu flexible.
- Un **chemin critique** dans un digraphe d'activité est un chemin dirigé du début à la fin, dont toutes les activités sont critiques (avec un temps mort = 0).
- Un digraphe d'activité a toujours au moins un chemin critique, mais il peut en avoir plusieurs.

La méthode du chemin critique



Activity	Pred.	Duration	EST	EFT	LST	LFT	Slack Time
Start	-	0	0	0	0	0	0
A	Start	3	0	3	0	3	0
B	A	3	3	6	3	6	0
C	A	2	3	5	4	6	1
D	B, C	4	6	10	6	10	0
E	B, C	3	6	9	8	11	2
F	B, C	2	6	8	9	11	3
G	D	1	10	11	10	11	0
H	D	1	10	11	10	11	0
I	D	1	10	11	12	13	2
J	E, F, G, H	2	11	13	11	13	0
Finish	I, J	0	13	**13**	13	**13**	0

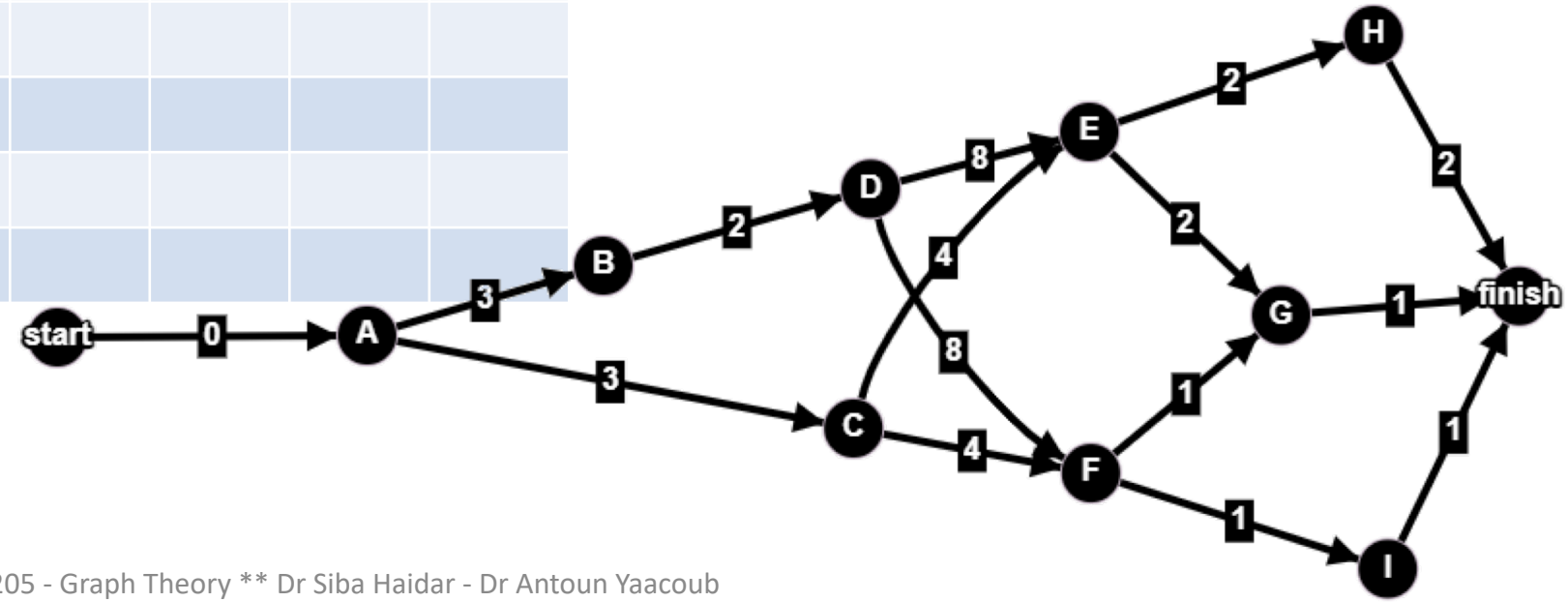
Digraphes et réseaux
Question 7

En utilisant les données du tableau

1. Construisez le digraphe d'activité.
2. Pour chaque activité, déterminez EST, EFT, LST, LFT et le temps mort.
3. Déterminez toutes les activités critiques.
4. Trouvez tous les chemins critiques dans le digraphe d'activité.

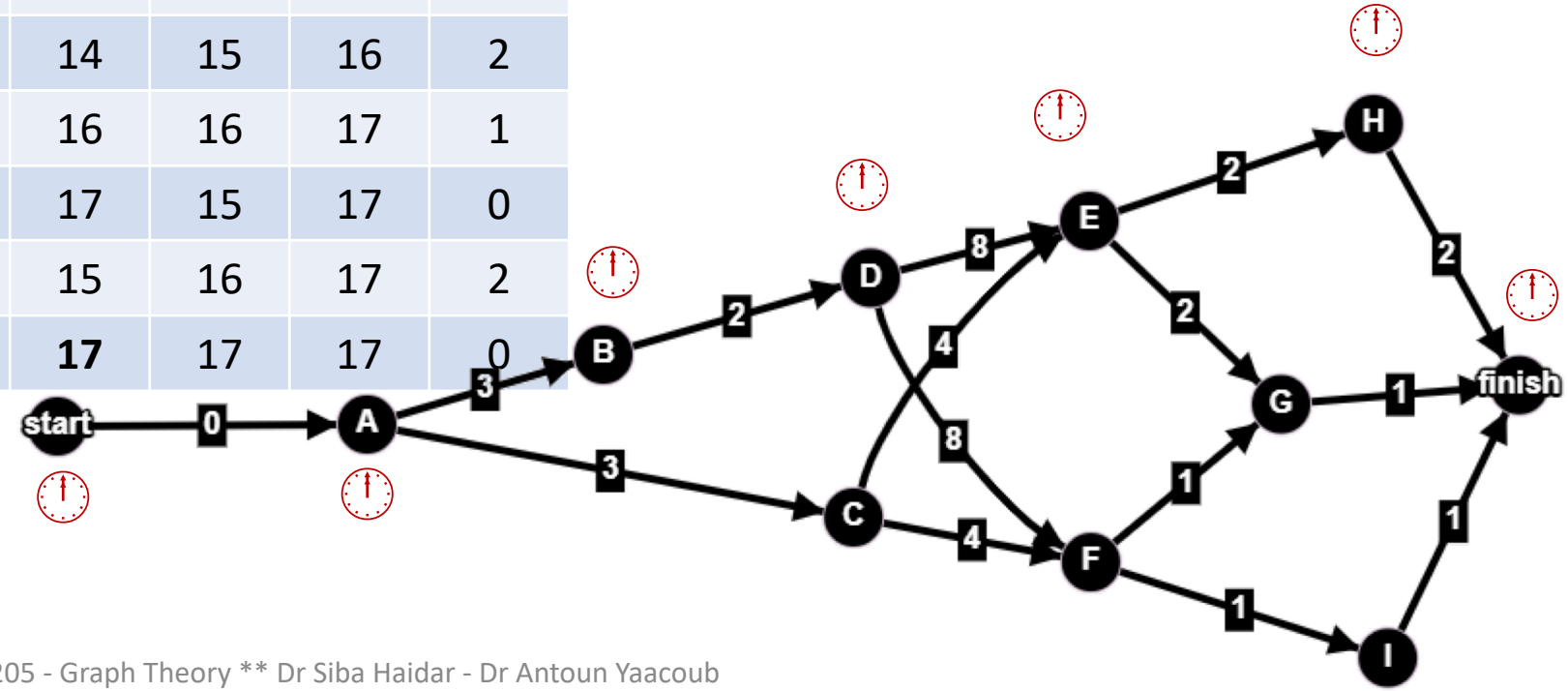
Activity	Immediate Predecessors	Duration in Weeks
Start	-	0
A	Start	3
B	A	2
C	A	4
D	B	8
E	C, D	2
F	C, D	1
G	E, F	1
H	E	2
I	F	1
Finish	G, H, I	0

Activity	Predecessors	Duration	EST	EFT	LST	LFT	Slack Time
Start	-	0					
A	Start	3					
B	A	2					
C	A	4					
D	B	8					
E	C, D	2					
F	C, D	1					
G	E, F	1					
H	E	2					
I	F	1					
Finish	G, H, I	0					



Activity	Predecessors	Duration	EST	EFT	LST	LFT	Slack Time
Start	-	0	0	0	0	0	0
A	Start	3	0	3	0	3	0
B	A	2	3	5	3	5	0
C	A	4	3	7	9	13	6
D	B	8	5	13	5	13	0
E	C, D	2	13	15	13	15	0
F	C, D	1	13	14	15	16	2
G	E, F	1	15	16	16	17	1
H	E	2	15	17	15	17	0
I	F	1	14	15	16	17	2
Finish	G, H, I	0	17	**17**	17	17	0

SOLUTION



Plan

- Graphes orientés
- Réseaux
- La méthode du chemin critique
- Biographies

Delbert Ray **Fulkerson**

- 14 août 1924 - 10 janvier 1976
- Mathématicien américain, coauteur de l'algorithme de Ford-Fulkerson.
- On a donné son nom au prix Fulkerson, qui récompense tous les trois ans des articles dans le domaine des mathématiques discrètes.

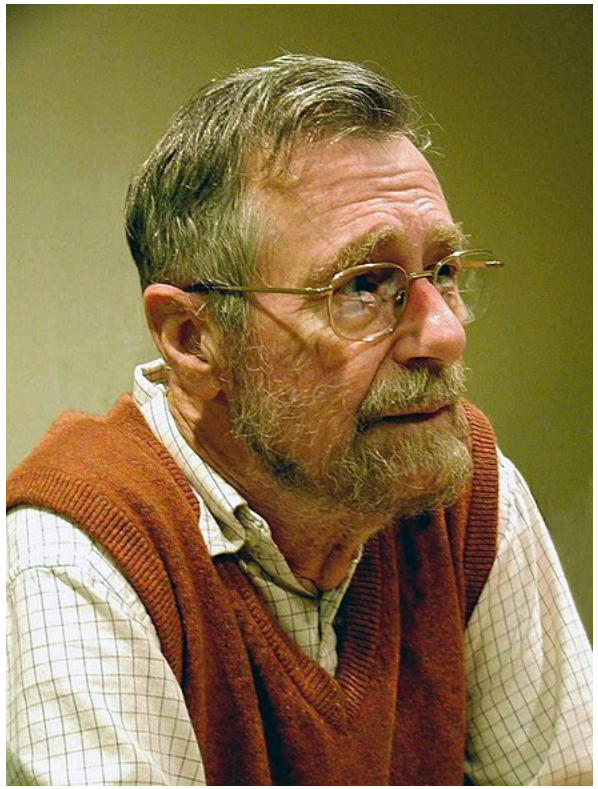
Lester Randolph **Ford** junior

- Né le 23 septembre 1927 à Houston et mort le 26 février 2017
- Mathématicien americain spécialiste des problèmes des réseaux de transport.
- Il est le fils du mathématicien Lester R. Ford senior.
- Il est connu pour sa contribution au problème de flot maximum : le théorème flot-max/coupe-min sur le problème de flot maximum et l'algorithme de Ford-Fulkerson pour le résoudre paraissent dans des rapports techniques en 1954 resp. 1955 et dans un périodique public en 1956 resp. 19572 ,3 ,4.
- Ford a également conçu, avec Richard Bellman et Samuel End, un algorithme pour déterminer les plus courts chemins dans un graphe dont les arcs peuvent avoir des poids négatifs. Cet algorithme s'appelle maintenant l'algorithme de Bellman-Ford.

Richard Ernest **Bellman**

- Né le 29 août 1920 à Brooklyn et mort le 19 mars 1984 à Los Angeles.
- Mathématicien américain.
- Il étudia les mathématiques appliquées.
- Célèbre pour diverses contributions dans plusieurs domaines des mathématiques, il est surtout l'inventeur de la programmation dynamique en 1953.

Edsger Wybe **Dijkstra**



- Né à Rotterdam le 11 mai 1930 et mort à Nuenen le 6 août 2002
- Mathématicien et informaticien néerlandais du xxe siècle.
- Il reçoit en 1972 le prix Turing pour ses contributions sur la science et l'art des langages de programmation et au langage Algol.
- Juste avant sa mort, en 2002, il reçoit le prix PoDC de l'article influent, pour ses travaux sur l'autostabilisation.
- L'année suivant sa mort, le prix sera renommé en son honneur prix Dijkstra.

Jack R. **Edmonds**

- Né le 5 avril 1934 (86 ans)
- Mathématicien et informaticien théoricien canadien, considéré comme l'un des contributeurs les plus importants dans le domaine de l'optimisation combinatoire.

Richard Manning **Karp**

- Né le 3 janvier 1935 à Boston dans le Massachusetts
- Chercheur américain connu notamment pour ses recherches en optimisation combinatoire et théorie de la complexité.
- Il a reçu le prix Turing en 1985 pour ses travaux.