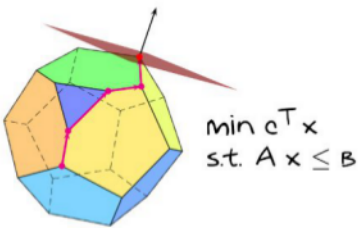
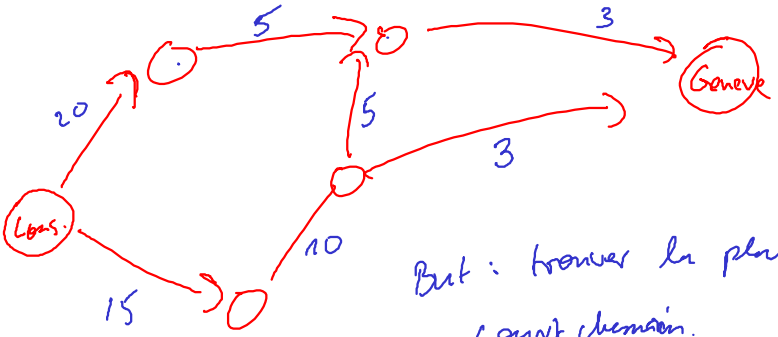
Chemins, circuits et ~~flots~~

- Graphes orientés
- Problème du plus court chemin
- Algorithme de parcours en largeur



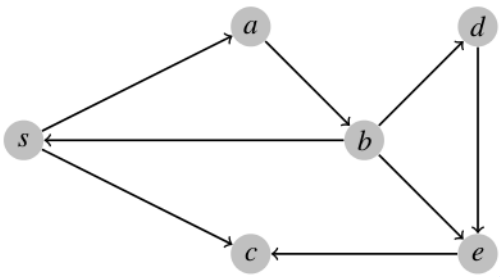
Motivation

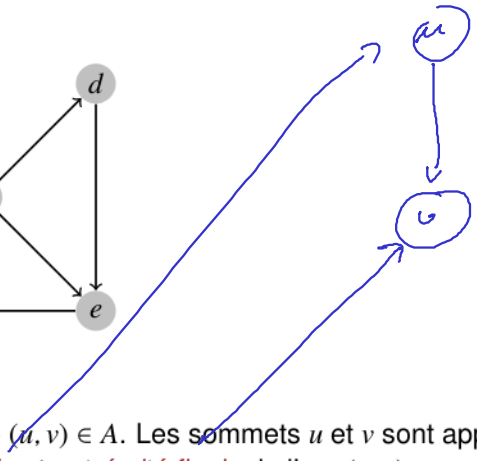


But : trouver le plus court chemin.

Graphes orientés

Un graphe *orienté* est un couple $D = (V, A)$, où V est l'ensemble fini des *sommets* ou *nœuds* et $A \subseteq (V \times V)$ est l'ensembles des *arcs* de G.

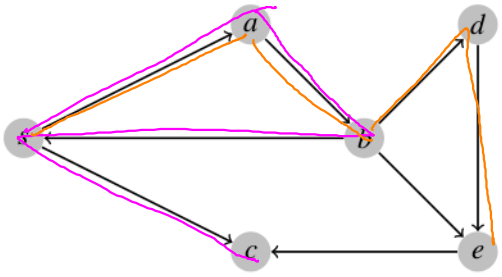




On dénote un arc par le couple $(u, v) \in A$. Les sommets u et v sont appelés respectivement *extrémité initiale* et *extrémité finale* de l'arc (u, v).

Marches et chemins

Une *marche* dans un graphe orienté est une séquence sous forme $v_0, \ldots, v_k$, où $(v_i, v_{i+1}) \in A$ pour $i = 0, \ldots, k-1$.
Une marche est un *chemin* si les sommets $v_0, \ldots, v_k$ sont tous différents. La *longueur* d'un chemin $v_0, \ldots, v_k$ est k.



s, a, b, s, c

s, a, b, d, e : chemin.

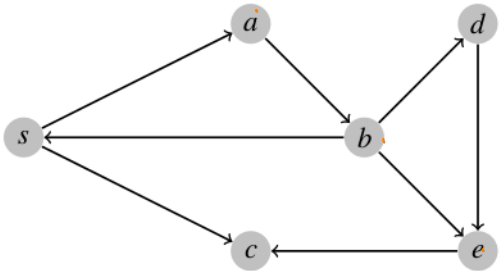
longueur : 4

Unweighted distance

non-pondéré

La *distance* $d(s,t)$ entre deux sommets $s,t \in V$ est le plus petit $k \in \mathbb{N}_0$ tel qu'il existe un chemin $s = v_0, \ldots, v_k = t$. ($k = \infty$ possible).
$d(s,t)$ est la longueur du *plus court chemin* reliant s et t.

$d(s,e) = 3$



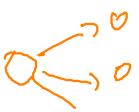
Quiz

Quelle est la plus grande longueur possible d'un chemin dans un graphe orienté $D = (V, A)$ où $|V| = n$?

- n
- $n - 1$
- $n^2 - 1$

Parmi les termes suivants, quels sont des bornes supérieures au nombre de chemins de longueur $n - 1$ dans un graphe orienté avec n sommets ?

- $n!$
- 2^n
- n



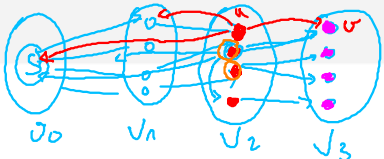
Marques de distance

Pour $i \in \mathbb{N}_0$, soit $V_i \subseteq V$ l'ensemble de sommets qui sont à distance i de s. Noter que $V_0 = \{s\}$.

Proposition

Pour $i = 1, \ldots, n-1$, l'ensemble V_i est égal à l'ensemble de sommets $v \in V \setminus (V_0 \cup \cdots \cup V_{i-1})$ tel qu'il existe un arc $(u, v) \in A$ avec $u \in V_{i-1}$.

Breadth-First-Search



L'algorithme maintient à jour les tableaux

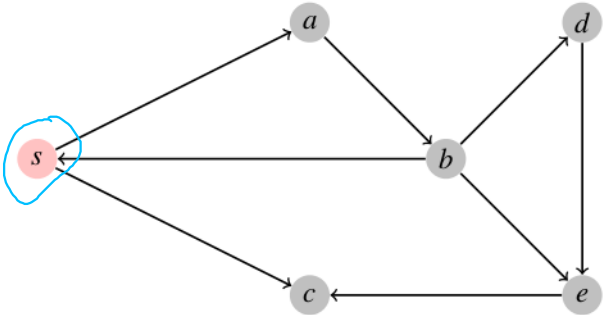
$$D[v_1 = s, v_2, \ldots, v_n]$$
$$\pi[v_1 = s, v_2, \ldots, v_n]$$

et une *queue* Q qui contient seulement s au début.

```
while Q ≠ ∅
    u := head(Q)
    for each (u, v) ∈ δ⁺(u)
        if (D[v] = ∞)
            π[v] := u
            D[v] := D[u] + 1
            enqueue(Q, v)
    dequeue(Q)
```

Exemple

while $Q \neq \emptyset$
  $u := head(Q)$
  for each $(u, v) \in \delta^+(u)$
    if $(D[v] = \infty)$
      $\pi[v] := u$
      $D[v] := D[u] + 1$
      $enqueue(Q, v)$
  $dequeue(Q)$

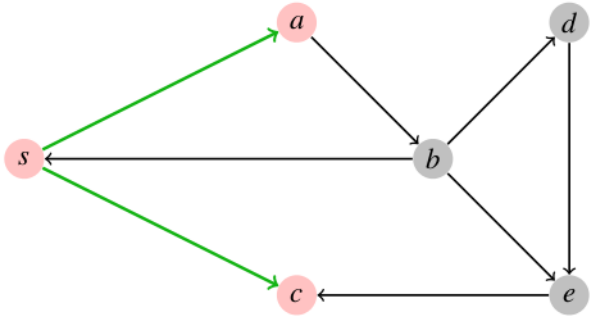


$Q = [s]$

$Q = [a, c]$

Exemple

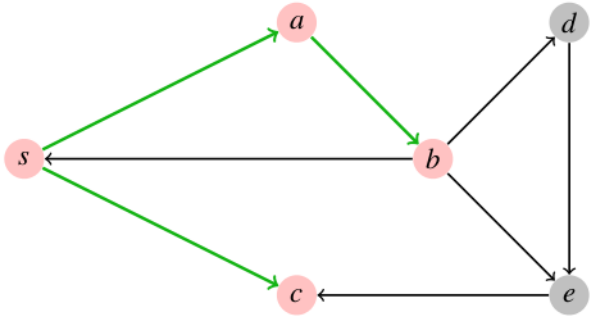
while $Q \neq \emptyset$
  $u := head(Q)$
  for each $(u, v) \in \delta^+(u)$
    if $(D[v] = \infty)$
      $\pi[v] := u$
      $D[v] := D[u] + 1$
      $enqueue(Q, v)$
  $dequeue(Q)$



$Q = [a, c]$ [c, b]

Exemple

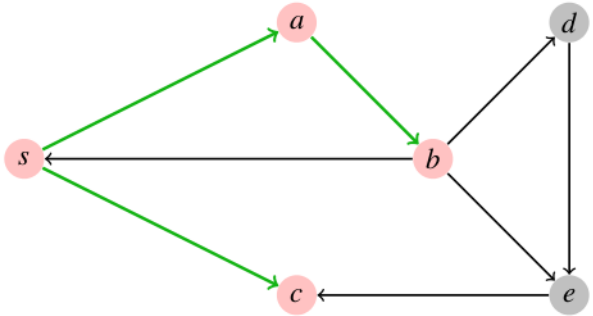
while $Q \neq \emptyset$
    $u := head(Q)$
    for each $(u, v) \in \delta^+(u)$
        if $(D[v] = \infty)$
            $\pi[v] := u$
            $D[v] := D[u] + 1$
            $enqueue(Q, v)$
    $dequeue(Q)$



$Q = [c, b]$

Exemple

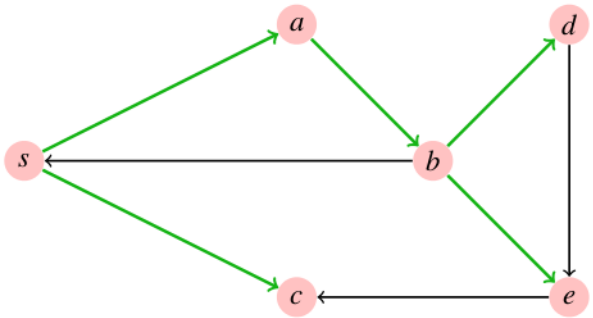
while $Q \neq \emptyset$
    $u := head(Q)$
    for each $(u, v) \in \delta^+(u)$
        if $(D[v] = \infty)$
            $\pi[v] := u$
            $D[v] := D[u] + 1$
            $enqueue(Q, v)$
    $dequeue(Q)$



$Q = [b]$

Exemple

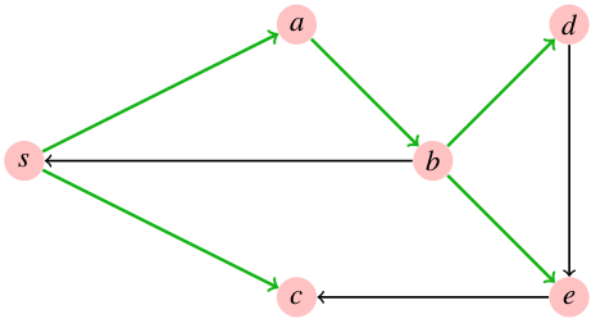
while $Q \neq \emptyset$
    $u := head(Q)$
    for each $(u, v) \in \delta^+(u)$
        if $(D[v] = \infty)$
            $\pi[v] := u$
            $D[v] := D[u] + 1$
            $enqueue(Q, v)$
    $dequeue(Q)$



$Q = [d, e]$

Exemple

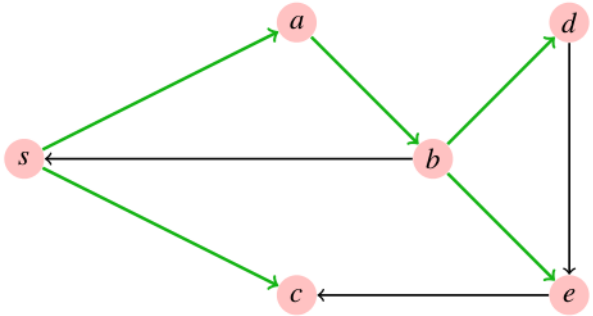
while $Q \neq \emptyset$
    $u := head(Q)$
    for each $(u, v) \in \delta^+(u)$
        if $(D[v] = \infty)$
            $\pi[v] := u$
            $D[v] := D[u] + 1$
            $enqueue(Q, v)$
    $dequeue(Q)$



$Q = [e]$

Exemple

```
while Q ≠ ∅
    u := head(Q)
    for each (u, v) ∈ δ+(u)
        if (D[v] = ∞)
            π[v] := u
            D[v] := D[u] + 1
            enqueue(Q, v)
    dequeue(Q)
```



$Q = []$

Analyse

Théorème

L'algorithme de parcours en largeur (breadth-first-search) se déroule en temps $O(|V| + |A|)$. Il est alors un algorithme en temps linéaire.

Analyse

Théorème

L'algorithme de parcours en largeur (breadth-first-search) se déroule en temps $O(|V| + |A|)$. Il est alors un algorithme en temps linéaire.

```
while Q ≠ ∅
    u := head(Q)
    for tout v ∈ δ⁺(u)
        if (D[v] = ∞)
            π[v] := u
            D[v] := D[u] + 1
            enqueue(Q, v)
    dequeue(Q)
```

Itération u : Au plus $c_1 \cdot |\delta^+(u)| + c_2$ opérations élémentaires.

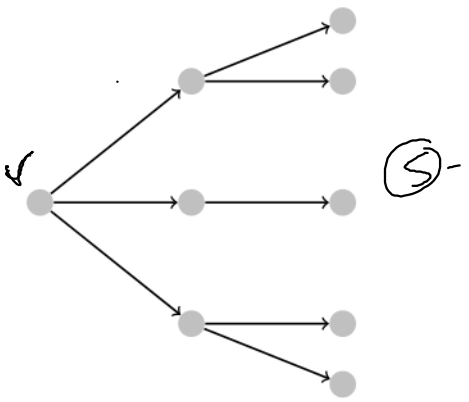
op. élementaires constant.

$$\sum_{u \in V} c_1 \cdot \delta^+(u) + c_2$$

$$= O(|A| + |V|)$$

Arbre orienté

Un *arbre orienté* est un graphe orienté $T = (V, A)$ avec $|A| = |V| - 1$ et dans lequel il y a un sommet $r \in T$ tel qu'il existe un chemin de r à tous les autres sommets de T.



$V' \subseteq V$

V' est découvert par S dans BFS.

$|V'| - 1$

$E' = \{(\pi(v), v) : v \in V' \setminus \{s\}\}$

$\Rightarrow T = (V', E')$ arbre.

Chemins dans un arbre

Lemme

Soit $T = (V, A)$ un graphe orienté tel que chaque sommet soit accessible depuis r.
Alors T est un arbre si et seulement si $\delta^-(r) = \emptyset$ et pour tout $v \in V \setminus \{r\}$ on a $|\delta^-(v)| = 1$.

L'arbre des plus courts chemins

Lemma

Considérons les tableaux D et π quand l'algorithme de parcours en largeur a terminé. Le graphe $T = (V', A')$ où $V' = \{v \in V : D[v] < \infty\}$ et $A' = \{(\pi(v), v) : 1 \leqslant D[v] < \infty\}$ est un arbre.

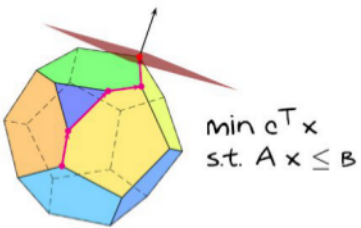
Proof.

- Clairement, $|A'| = |V'| - 1$.
- Pour chaque $i \in \{1, \ldots, n-1\}$, en retraçants les marques π, on arrive finalement à s.

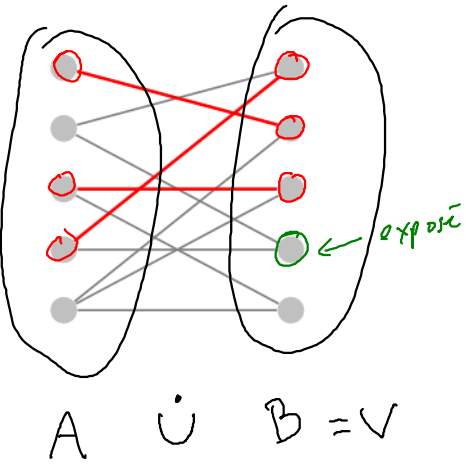
□

Chemins, circuits et flots

- Couplages de cardinalité maximale dans un graphe biparti
- Chemins d'augmentation
- Un algorithme en temps $O(m \cdot n)$



Sommets exposés ou couplés



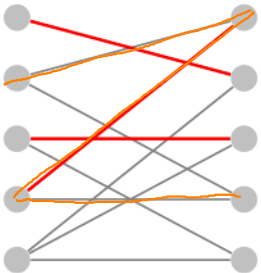
Soit $G = (V, E)$ un graphe non-orienté biparti.
On veut trouver le couplage de cardinalité maximale.

Soit $M \subseteq E$ un couplage.

- Un sommet qui est l'extremité d'une arête dans M est dit *couplé*.
- Un sommet qui n'est pas couplé est dit *exposé*.

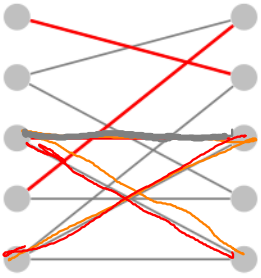
Chemins alternants

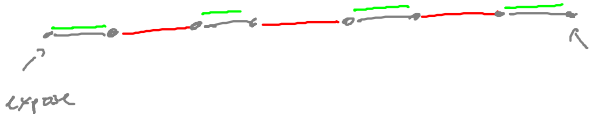
Un chemin alternant par rapport à un couplage M est un chemin qui alterne des arêtes dans M avec des arêtes dans $E \setminus M$.



Chemins d'augmentation

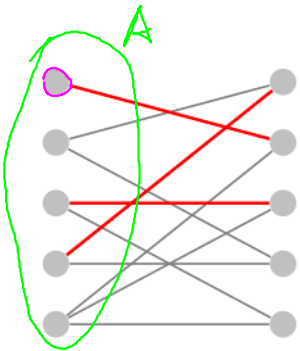
Un chemin alternant qui commence et termine à des sommets exposés est dit *chemin d'augmentation*.



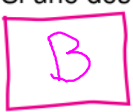


Chemins d'augmentation

Un chemin alternant qui commence et termine à des sommets exposés est dit *chemin d'augmentation*.



Si une des extremités d'un chemin d'augmentation est dans A, alors l'autre est dans



Un critère de cardinalité maximale

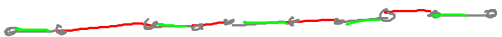
Théorème

Un couplage M d'un graphe (pas forcement biparti) est de cardinalité maximale si et seulement s'il n'existe pas de chemins d'augmentation par rapport à M.

=> "

Supposé qu'il existe

P :



E(P)

$M' = (M \setminus E(P)) \cup (E(P) \setminus M)$

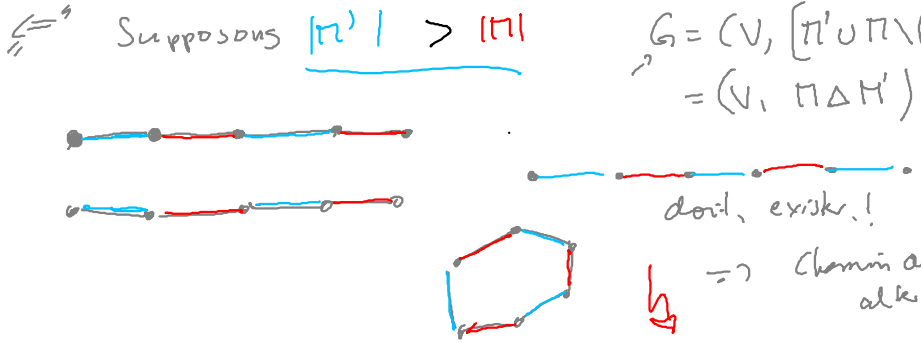
$|M'| = |M| + 1$

Alors M n'est pas max.

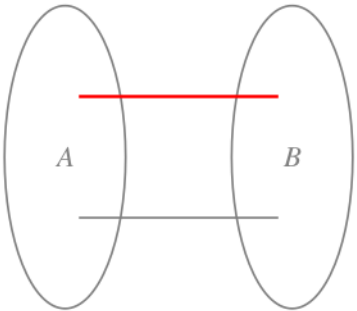
Un critère de cardinalité maximale

Théorème

Un couplage M d'un graphe (pas forcement biparti) est de cardinalité maximale si et seulement s'il n'existe pas de chemins d'augmentation par rapport à M.

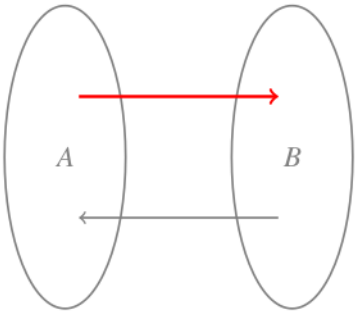


Calculer des chemins d'augmentation



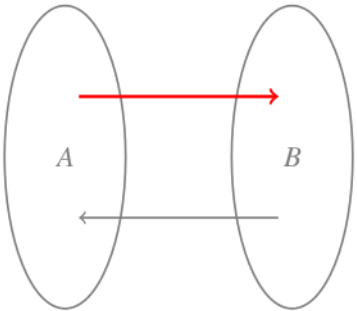
- Transfomer $G = (A + B, E)$ dans un graphe orienté $D = (V, A)$ comme suit.

Calculer des chemins d'augmentation



- Transfomer $G = (A + B, E)$ dans un graphe orienté $D = (V, A)$ comme suit.
- Orienter une arête du couplage de A à B.
- Orienter une arête dans $E \setminus M$ de B à A.

Calculer des chemins d'augmentation

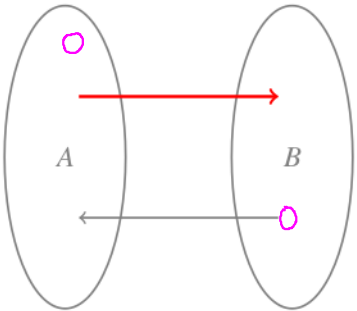


- Transfomer $G = (A + B, E)$ dans un graphe orienté $D = (V, A)$ comme suit.
- Orienter une arête du couplage de A à B.
- Orienter une arête dans $E \setminus M$ de B à A.

Quiz : Supposons v est exposé et v est dans A,

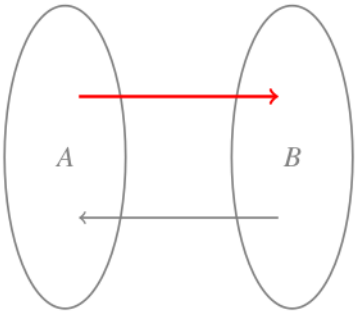
qu'est $|\delta^+(v)|$?

Calculer des chemins d'augmentation



- Transfomer $G = (A + B, E)$ dans un graphe orienté $D = (V, A)$ comme suit.
- Orienter une arête du couplage de A à B.
- Orienter une arête dans $E \setminus M$ de B à A.
- Trouver un chemin dans ce graphe orienté entre deux sommets exposés.

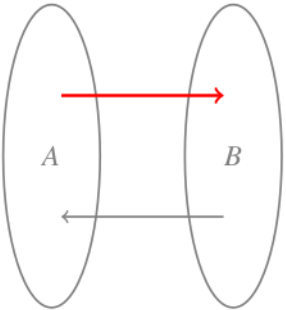
Calculer des chemins d'augmentation



- Transfomer $G = (A + B, E)$ dans un graphe orienté $D = (V, A)$ comme suit.
- Orienter une arête du couplage de A à B.
- Orienter une arête dans $E \setminus M$ de B à A.
- Trouver un chemin dans ce graphe orienté entre deux sommets exposés.

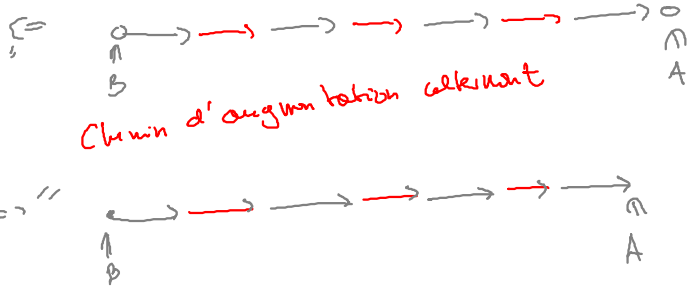
Quiz : Un tel chemin commence avec un sommet exposé dans B et termine avec un sommet exposé dans A

Calculer des chemins d'augmentation (cont.)

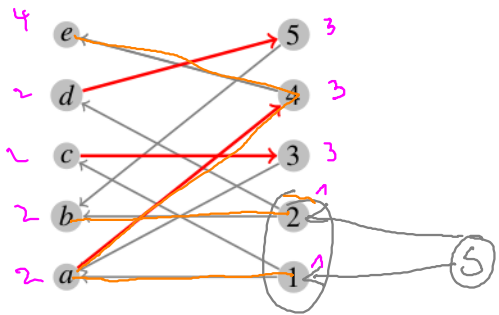


Théorème

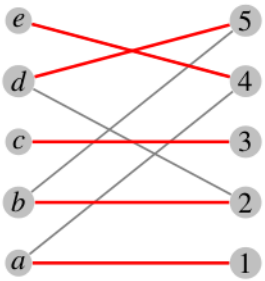
Il existe un chemin d'augmentation pour M dans G si et seulement s'il existe un chemin d'un sommet exposé dans B à un sommet exposé de A dans le graphe orienté D.



User BFS pour trouver des chemins d'augmentation



User BFS pour trouver des chemins d'augmentation



Algorithme pour un couplage de cardinalité maximale dans un graphe biparti

$M = \emptyset$

while $\exists$ chemin d'augmentation pour M

    Update M

return M

Augmente : $O(|V|)$ fois

$O(|E|)$

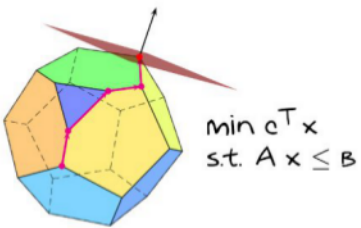
Présupposition: G n'a pas de sommet isolé ($\Rightarrow |E| \geqslant |V|/2$).

Théorème

Un couplage de cardinalité maximale dans un graphe biparti $G = (V, E)$ peut être calculé en temps $O(|V| \cdot |E|)$

Chemins, circuits et flots

- Graphes orientés pondérés
- Le plus court chemin
- Algorithme de Bellman-Ford

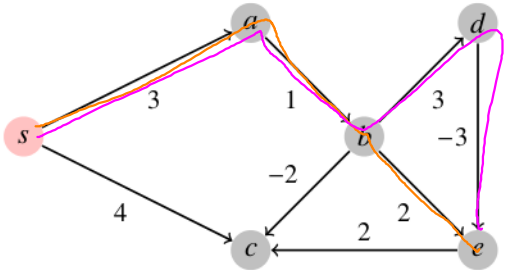


Graphes orientés pondérés

Soit $D = (V, A)$ un graphe pondéré (sans boucles). Soit $\ell : A \to \mathbb{R}$ la *longueur* des arcs.
La *longueur* d'une marche $W = v_0, \ldots, v_k$ est la somme de longueur de ses arcs :

$$\ell(W) = \sum_{i=1}^{k} \ell(v_{i-1}, v_i).$$

La *distance* entre deux sommets s et t est la longueur d'un *plus court chemin* de s à t.



Problème des plus courts chemins

Problème des plus courts chemins (seul sommet source)

Donné un graphe orienté avec longueur des arcs et un sommet source s, calculer $d(s, v)$ pour tout $v \in V$.

Problème des plus courts chemins

Problème des plus courts chemins (seul sommet source)

Donné un graphe orienté avec longueur des arcs et un sommet source s, calculer $d(s, v)$ pour tout $v \in V$.

- est NP-complet en général.

Problème des plus courts chemins (seul sommet source)

Donné un graphe orienté avec longueur des arcs et un sommet source s, calculer $d(s, v)$ pour tout $v \in V$.

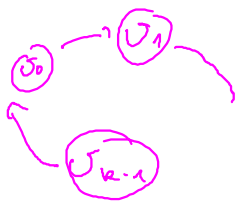
- est NP-complet en général.
- peut être résolu en temps polynômial s'il n'existe pas de circuit de poids total négatif.

Problème des plus courts chemins (seul sommet source)

Donné un graphe orienté avec longueur des arcs et un sommet source s, calculer $d(s, v)$ pour tout $v \in V$.

- est NP-complet en général.
- peut être résolu en temps polynômial s'il n'existe pas de circuit de poids total négatif.

Un *circuit* est une marche $v_0, v_1, \ldots, v_k$ avec $v_0 = v_k$.



L'algorithme de Bellman-Ford

Un algorithme pour calculer des plus courtes *marches*.

Donné : $D = (V, A)$ (sans boucles), $\ell : A \to \mathbb{R}$ et un sommet source $s \in V$

But : Calculer les distances des plus courts chemin de s à tous autres sommets.

Présupposition : Chaque sommet est accessible depuis s

L'algorithme de Bellman-Ford (cont.)

Pour $k \geqslant 0$ et $t \in V$:

$d_k(t) =$ longueur minimale d'une marche s–t traversant au plus k arcs. ($k = \infty$ possible)

L'algorithme de Bellman-Ford (cont.)

Pour $k \geqslant 0$ et $t \in V$:

$d_k(t) =$ longueur minimale d'une marche s–t traversant au plus k arcs. ($k = \infty$ possible)

$d_o(s) = 0, \quad d_0(t) = \infty, t \neq s$

L'algorithme de Bellman-Ford (cont.)

Pour $k \geqslant 0$ et $t \in V$:

$d_k(t) =$ longueur minimale d'une marche s–t traversant au plus k arcs. ($k = \infty$ possible)

Supposons que $d_i(t)$ est déjà connu pour $i \leqslant k$ et chaque $t \in V$.

$d_{k+1}(t)$

L'algorithme de Bellman-Ford (cont.)

Pour $k \geqslant 0$ et $t \in V$:

$d_k(t) =$ longueur minimale d'une marche s–t traversant au plus k arcs. ($k = \infty$ possible)

Supposons que $d_i(t)$ est déjà connu pour $i \leqslant k$ et chaque $t \in V$.

Maintenant : Calculer $d_{k+1}(t)$ pour tout $t \in V$.

L'algorithme de Bellman-Ford (cont.)

Pour $k \geqslant 0$ et $t \in V$:

$d_k(t) =$ longueur minimale d'une marche s–t traversant au plus k arcs. ($k = \infty$ possible)

Supposons que $d_i(t)$ est déjà connu pour $i \leqslant k$ et chaque $t \in V$.

Maintenant : Calculer $d_{k+1}(t)$ pour tout $t \in V$.

L'algorithme de Bellman-Ford (cont.)

Pour $k \geqslant 0$ et $t \in V$:

$d_k(t) =$ longueur minimale d'une marche s–t traversant au plus k arcs. ($k = \infty$ possible)

Supposons que $d_i(t)$ est déjà connu pour $i \leqslant k$ et chaque $t \in V$.

Maintenant : Calculer $d_{k+1}(t)$ pour tout $t \in V$.

Premier cas : La plus courte marche traversant $k+1$ arcs au plus traverse exactement $k+1$ arcs.

$d_{k+1}(t) = d_k(u) + \ell(u,t)$

L'algorithme de Bellman-Ford (cont.)

Pour $k \geqslant 0$ et $t \in V$:

$d_k(t) =$ longueur minimale d'une marche s–t traversant au plus k arcs. ($k = \infty$ possible)

Supposons que $d_i(t)$ est déjà connu pour $i \leqslant k$ et chaque $t \in V$.

Maintenant : Calculer $d_{k+1}(t)$ pour tout $t \in V$.

∃ une plus

Deuxième cas : ~~La plus~~ courte marche traversant $k+1$ arcs au plus traverse k arcs au plus.

$d_{k+1}(t) = d_k(t)$, 3. cas. ∃ pas de marche de s at traverse $\leq k+1$ arcs.

$d_{k+1}(t) = d_k(t) = \infty$

L'algorithme de Bellman-Ford (cont.)

$$d_o(s) = 0, \quad d_0(t) = \infty, t \neq s$$

$$k \geqslant 0,\ t \in V : \ d_{k+1}(t) = \min\{d_k(t), \min_{(u,t)\in A} \{d_k(u) + \ell(u,t)\}\}$$

L'algorithme de Bellman-Ford (cont.)

$$d_o(s) = 0, \quad d_0(t) = \infty, t \neq s$$

$$k \geqslant 0,\ t \in V :\ d_{k+1}(t) = \min\{d_k(t),\ \min_{(u,t)\in A}\{d_k(u) + \ell(u,t)\}.$$

Méthode pour calculer les valeurs $d_{k+1}(t)$ *en supposant* que les valeurs $d_k(t)$ sont pré-déterminées :

for each $t \in V$:

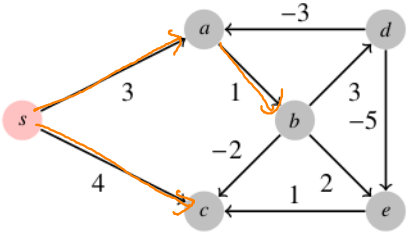
$d_{k+1}(t) := d_k(t)$

for each $(u,t) \in A$

if: $d_k(u) + \ell(u,t) < d_{k+1}(t)$

$d_{k+1}(t) := d_k(u) + \ell(u,t)$

Exemple



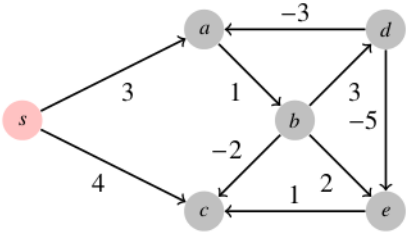
for each $t \in V$:
$d_{k+1}(t) := d_k(t)$

for each $(u, t) \in A$
if: $d_k(u) + \ell(u, t) < d_{k+1}(t)$
$d_{k+1}(t) := d_k(u) + \ell(u, t)$

$d_0(s) + 3 < d_1(a) = \infty$

0	3	∞	4	∞	∞	d_1
0	∞	∞	∞	∞	∞	d_0
s	a	b	c	d	e	

Exemple

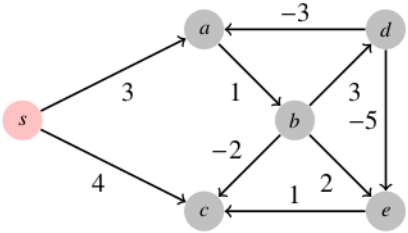


for each $t \in V$:
$d_{k+1}(t) := d_k(t)$

for each $(u, t) \in A$
if: $d_k(u) + \ell(u, t) < d_{k+1}(t)$
$d_{k+1}(t) := d_k(u) + \ell(u, t)$

0	3	∞	4	∞	∞	d_1
0	∞	∞	∞	∞	∞	d_0
s	a	b	c	d	e	

Exemple

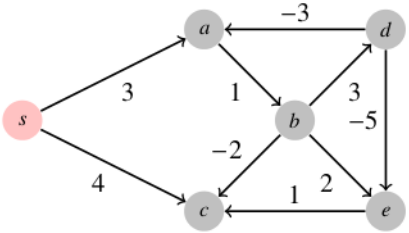


for each $t \in V$:
$d_{k+1}(t) := d_k(t)$

for each $(u, t) \in A$
if: $d_k(u) + \ell(u, t) < d_{k+1}(t)$
$d_{k+1}(t) := d_k(u) + \ell(u, t)$

0	3	4	4	∞	∞	d_2
0	3	∞	4	∞	∞	d_1
0	∞	∞	∞	∞	∞	d_0
s	a	b	c	d	e	

Exemple

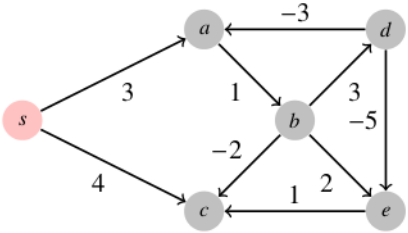


for each $t \in V$:
$d_{k+1}(t) := d_k(t)$

for each $(u, t) \in A$
if: $d_k(u) + \ell(u, t) < d_{k+1}(t)$
$d_{k+1}(t) := d_k(u) + \ell(u, t)$

0	3	4	2	7	6	d_3
0	3	4	4	∞	∞	d_2
0	3	∞	4	∞	∞	d_1
0	∞	∞	∞	∞	∞	d_0
s	a	b	c	d	e	

Exemple

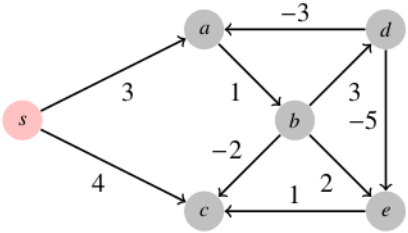


for each $t \in V$:
$d_{k+1}(t) := d_k(t)$

for each $(u, t) \in A$
if: $d_k(u) + \ell(u, t) < d_{k+1}(t)$
$d_{k+1}(t) := d_k(u) + \ell(u, t)$

0	3	4	2	7	2	d_4
0	3	4	2	7	6	d_3
0	3	4	4	∞	∞	d_2
0	3	∞	4	∞	∞	d_1
0	∞	∞	∞	∞	∞	d_0
s	a	b	c	d	e	

Exemple

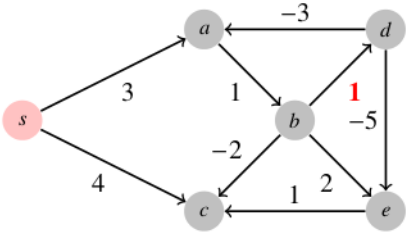


for each $t \in V$:
$d_{k+1}(t) := d_k(t)$

for each $(u, t) \in A$
if: $d_k(u) + \ell(u, t) < d_{k+1}(t)$
$d_{k+1}(t) := d_k(u) + \ell(u, t)$

0	3	4	2	7	2	d_5
0	3	4	2	7	2	d_4
0	3	4	2	7	6	d_3
0	3	4	4	∞	∞	d_2
0	3	∞	4	∞	∞	d_1
0	∞	∞	∞	∞	∞	d_0
s	a	b	c	d	e	

Exemple (cont.)

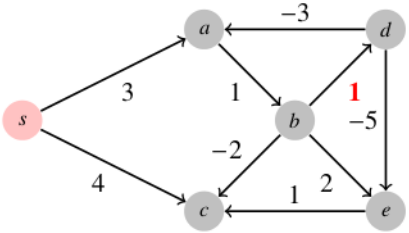


for each $t \in V$:
$d_{k+1}(t) := d_k(t)$

for each $(u, t) \in A$
if: $d_k(u) + \ell(u, t) < d_{k+1}(t)$
$d_{k+1}(t) := d_k(u) + \ell(u, t)$

0	∞	∞	∞	∞	∞	d_0
s	a	b	c	d	e	node \ d_k

Exemple (cont.)

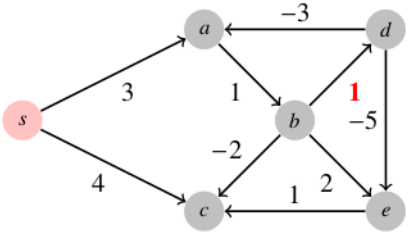


for each $t \in V$:
$d_{k+1}(t) := d_k(t)$

for each $(u, t) \in A$
if: $d_k(u) + \ell(u, t) < d_{k+1}(t)$
$d_{k+1}(t) := d_k(u) + \ell(u, t)$

0	3	∞	4	∞	∞	d_1
0	∞	∞	∞	∞	∞	d_0
s	a	b	c	d	e	node \ d_k

Exemple (cont.)

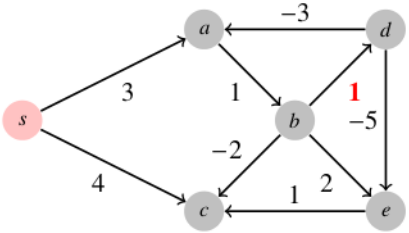


for each $t \in V$:
$d_{k+1}(t) := d_k(t)$

for each $(u, t) \in A$
if: $d_k(u) + \ell(u, t) < d_{k+1}(t)$
$d_{k+1}(t) := d_k(u) + \ell(u, t)$

0	3	4	4	∞	∞	d_2
0	3	∞	4	∞	∞	d_1
0	∞	∞	∞	∞	∞	d_0
s	a	b	c	d	e	node \ d_k

Exemple (cont.)

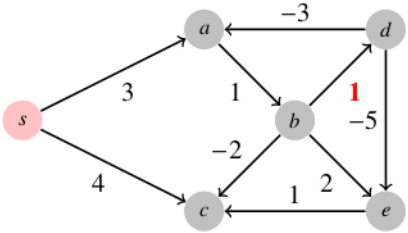


for each $t \in V$:
$d_{k+1}(t) := d_k(t)$

for each $(u, t) \in A$
if: $d_k(u) + \ell(u, t) < d_{k+1}(t)$
$d_{k+1}(t) := d_k(u) + \ell(u, t)$

0	3	4	2	5	6	d_3
0	3	4	4	∞	∞	d_2
0	3	∞	4	∞	∞	d_1
0	∞	∞	∞	∞	∞	d_0
s	a	b	c	d	e	node \ d_k

Exemple (cont.)

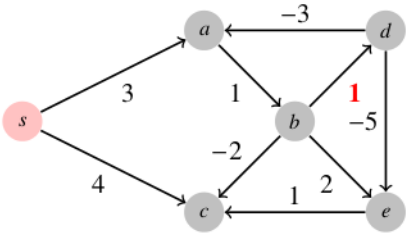


for each $t \in V$:
$d_{k+1}(t) := d_k(t)$

for each $(u, t) \in A$
if: $d_k(u) + \ell(u, t) < d_{k+1}(t)$
$d_{k+1}(t) := d_k(u) + \ell(u, t)$

0	2	4	2	5	0	d_4
0	3	4	2	5	6	d_3
0	3	4	4	∞	∞	d_2
0	3	∞	4	∞	∞	d_1
0	∞	∞	∞	∞	∞	d_0
s	a	b	c	d	e	node \ d_k

Exemple (cont.)

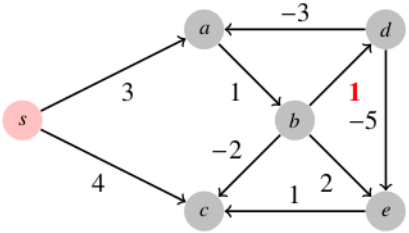


for each $t \in V$:
$d_{k+1}(t) := d_k(t)$

for each $(u, t) \in A$
if: $d_k(u) + \ell(u, t) < d_{k+1}(t)$
$d_{k+1}(t) := d_k(u) + \ell(u, t)$

0	2	3	2	5	0	d_5
0	2	4	2	5	0	d_4
0	3	4	2	5	6	d_3
0	3	4	4	∞	∞	d_2
0	3	∞	4	∞	∞	d_1
0	∞	∞	∞	∞	∞	d_0
s	a	b	c	d	e	node \ d_k

Exemple (cont.)



for each $t \in V$:
$d_{k+1}(t) := d_k(t)$

for each $(u, t) \in A$
if: $d_k(u) + \ell(u, t) < d_{k+1}(t)$
$d_{k+1}(t) := d_k(u) + \ell(u, t)$

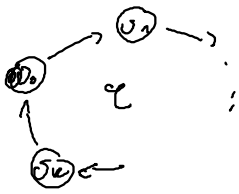
0	2	3	1	4	0	d_6
0	2	3	2	5	0	d_5
0	2	4	2	5	0	d_4
0	3	4	2	5	6	d_3
0	3	4	4	∞	∞	d_2
0	3	∞	4	∞	∞	d_1
0	∞	∞	∞	∞	∞	d_0
s	a	b	c	d	e	node \ d_k

Circuits absorbants

Théorème

Soient $D = (V, A)$, $s \in V$, $\ell : A \to \mathbb{R}$. Alors on a $d_n = d_{n-1}$ où $n = |V|$ ssi D n'a pas de circuit de longueur totale négative qui est accessible depuis s.

$\Rightarrow$



$d(v_i) < \infty$ (accessible)

$$0 = \sum_{(u,v)\in\mathcal{C}} \left(\underbrace{d_n(s)}_{\leq d_{n-1}(u) + \ell(u,v)} - d_n(u) \right)$$

$d_{n-1}(u)$

$$\leq \sum_{(u,v)\in\mathcal{C}} \ell(u,v)$$

Circuits absorbants

Théorème

Soient $D = (V, A)$, $s \in V$, $\ell : A \to \mathbb{R}$. Alors on a $d_n = d_{n-1}$ où $n = |V|$ ssi D n'a pas de circuit de longueur totale négative qui est accessible depuis s.

$\Leftarrow$

Supposons $d_n(t) < d_{n-1}(t) < \infty$

Parcours plus court utilisant n arcs.

$\Leftarrow$ utilise $< n$ arcs plus longs

Des plus courts chemins

Théorème

négatif

Donné $D = (V, A)$, $s \in V$, $\ell : A \to \mathbb{R}$, et supposons qu'il n'existe pas de circuit ~~absorbant~~ qui est accessible depuis s. Alors pour chaque $t \in V$ $d_{n-1}(t)$ est la distance de s à t.

une marche plus courte utilisant $\leq s$ ou plus $(n-1)$ arc.
et nombre d'arc minimal.

est un chemin.

⊛

Calculer des plus courts chemins

Calculer les valeurs $d_{k+1}(t)$ et le prédécesseur $\pi_{k+1}(t)$ *en supposant* que les valeurs $d_k(t)$ et $\pi_{k+1}(t)$ sont prédéterminées :

for each $t \in V$:

$d_{k+1}(t) := d_k(t)$

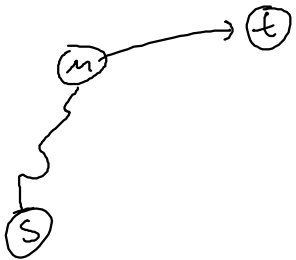
$\pi_{k+1}(t) := \pi_k(t)$

for each $(u, t) \in A$

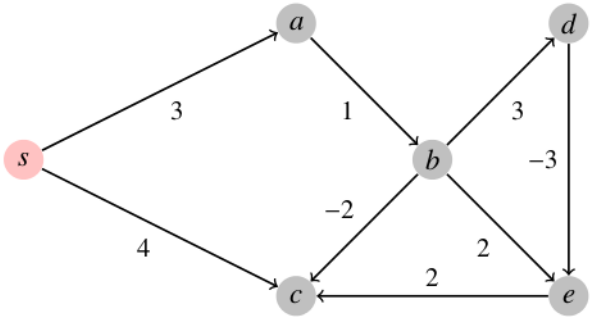
if: $d_k(u) + \ell(u, t) < d_{k+1}(t)$

$d_{k+1}(t) := d_k(u) + \ell(u, t)$

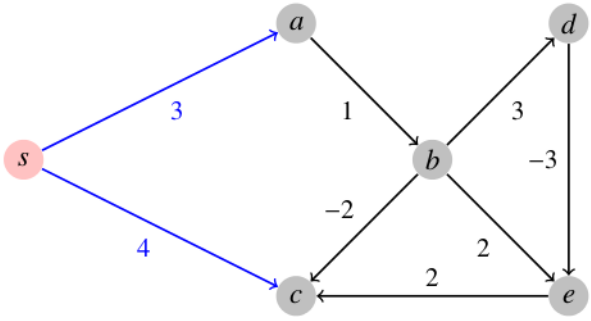
$\pi_{k+1}(t) :=$ u



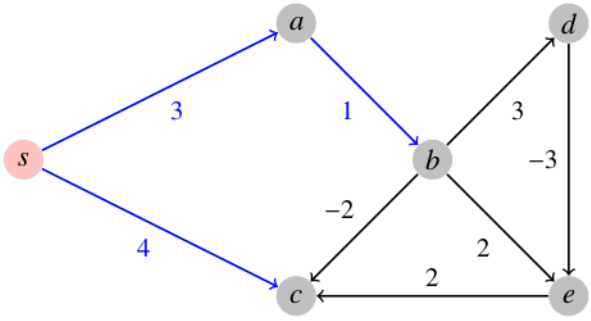
Exemple



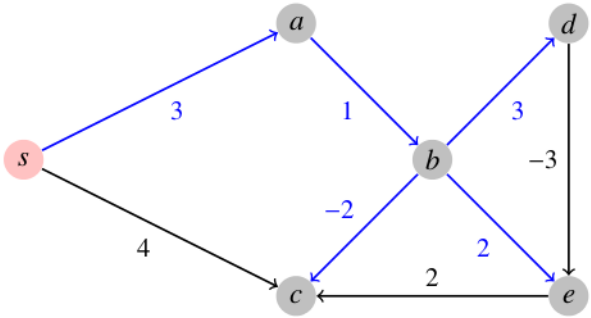
Exemple



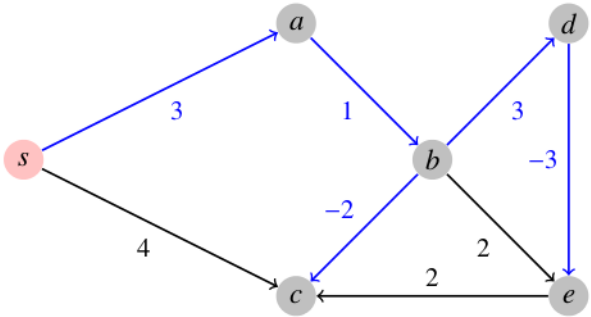
Exemple



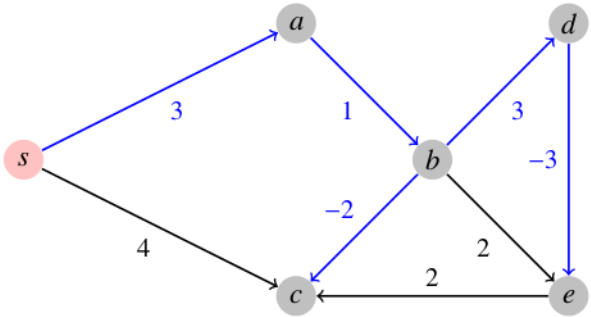
Exemple



Exemple



Exemple



L'arbre des plus courts chemins

Théorème

Soit $D = (V, A)$ un graphe orienté et supposons que chaque sommet est accessible depuis s. Le graphe orienté $T = (V, A')$ où $A' = \{(\pi(u), u) : u \in V \setminus \{s\}\}$ est un arbre avec racine s. Le chemin unique reliant s à tout sommet t dans T est un plus chemin de s à t dans D.

Temps d'exécution de Bellman-Ford

initialize

$\forall t \in V \setminus \{s\}, d_o(t) = \infty,\ \pi_0(t) = 0$

$d_0(s) = 0$

for $k = 1$ to n

for each $t \in V$:

$d_{k+1}(t) := d_k(t)$

$\pi_{k+1}(t) := \pi_k(t)$

for each $(u, t) \in A$

if: $d_k(u) + \ell(u, t) < d_{k+1}(t)$

$d_{k+1}(t) := d_k(u) + \ell(u, t)$

$\pi_{k+1}(t) := u$

if $\exists t \in V$ avec $d_n(t) < d_{n-1}(t)$

D a un circuit absorbant

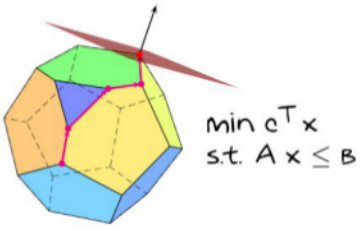
Temps d'exécution de Bellman-Ford (cont.)

Théorème

Soit $D = (V, A)$ un graphe orienté avec longueurs ℓ et supposons que chaque sommet est accessible depuis $s \in V$, alors Bellman-Ford se déroule en temps $O(|V| \cdot |A|)$

Chemins, circuits et flots

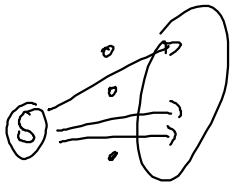
- Des plus courts chemins et la programmation linéaire

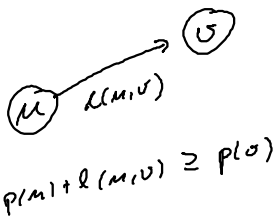


Potentiels

Soit $D = (V, A)$ un graphe orienté avec longueur d'arcs $\ell : A \to \mathbb{R}$. Une fonction $p : V \to \mathbb{R}$ est dite un *potentiel* si

$$\forall a = (u, v) \in A : \quad \ell(a) \geqslant p(v) - p(u).$$





L'existence de potentiels

Théorème

$D = (V, A)$ avec $\ell : A \to \mathbb{R}$ admet un potentiel p si et seulement si chaque circuit est de poids total non-négatif.

Calculer des distances avec la programmation linéaire

Théorème

Soit $D = (V, A)$ un graphe orienté avec longueur d'arcs $\ell : A \to \mathbb{R}$, $s \in V$ tel que chaque sommet de V est accessible depuis s et supposons que chaque circuit est de poids total non-négatif. Soit p un potentiel avec $p(s) = 0$ et $\sum_{v \in V} p(v)$ maximale. Alors

$$\forall t \in V : \; p(t) = \mathrm{dist}_\ell(s, t).$$

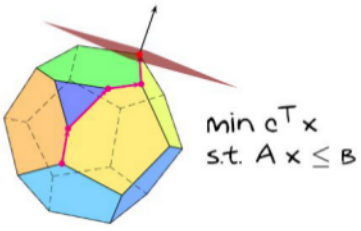
$$\max \sum_{v \in V} p(v)$$

$$\forall (u,v) : \; p(u) - p(v) \geq -\ell(u,v)$$

PL.

Chemins, circuits et flots

- Representation de graphes



Matrice d'adjacence

$G = (V, E)$ un graphe *non-orienté*,
$V = \{1, \ldots, n\}$
$M \in \{0,1\}^{|V|\times|V|}$ avec

$$a_{ij} = \begin{cases} 1 & \text{si } ij \in E \\ 0 & \text{sinon.} \end{cases}$$

$D = (V, A)$ un graphe *orienté*,
$V = \{1, \ldots, n\}$
$M \in \{0,1\}^{|V|\times|V|}$ avec

$$a_{ij} = \begin{cases} 1 & \text{si } (i,j) \in A \\ 0 & \text{sinon.} \end{cases}$$

Adjacency list

$G = (V, E)$ graphe *non-orienté*,
$V = \{1, \ldots, n\}$

- *Array* V indexé par sommets
- $V[i]$ *montre* une *liste* de voisins de i.

$D = (V, A)$ graphe *orienté* , $V = \{1, \ldots, n\}$

- *Array* V indexé par sommets
- $V[i]$ *montre* une *liste* d'extremités finales d'arcs sortant de i.

Représentation comme liste d'adjacence en Python

```
graph = {'A': ['B', 'C'],
         'B': ['C', 'D'],
         'C': ['D'],
         'D': ['C'],
         'E': ['F'],
         'F': ['C']}
```

Représentation comme liste d'adjacence en Python

```
graph = {'A': ['B', 'C'],
         'B': ['C', 'D'],
         'C': ['D'],
         'D': ['C'],
         'E': ['F'],
         'F': ['C']}
```

Représentation comme liste d'adjacence en Python

```
graph = {'A': ['B', 'C'],
         'B': ['C', 'D'],
         'C': ['D'],
         'D': ['C'],
         'E': ['F'],
         'F': ['C']}
```

Représentation comme liste d'adjacence en Python

```
graph = {'A': ['B', 'C'],
         'B': ['C', 'D'],
         'C': ['D'],
         'D': ['C'],
         'E': ['F'],
         'F': ['C']}
```

Représentation comme liste d'adjacence en Python

```
graph = {'A': ['B', 'C'],
         'B': ['C', 'D'],
         'C': ['D'],
         'D': ['C'],
         'E': ['F'],
         'F': ['C']}
```