Licence MI

Algorithmique Avancee

Plis fòs ba pengwen là !

Présentation de l'EC

- Module Complet : 14h C / 20h TD / 20h TP
- Cette partie : 7h C / 10h TD / 10h TP
- Objectif du cours : Structures de données avancées
- Langage informatique : Java
- Plan du cours :

1. HashTable
2. Graphes
3. Flots

Introduction :
Structures de données algorithmiques

Exemples de structures de données algorithmiques :

tableaux, piles, liste, files.

Objectif des structures de données algorithmiques :

- Rassembler les données.
- Sous une forme adaptée au problème.

Ces problèmes se résument souvent à :

- mise en évidence des liens entre données
- insertion, suppression
- recherche, parcours, tri

Une nouvelle structure algorithmique :

- des solutions plus efficaces
- des solutions à de nouveaux problèmes

Introduction :
Implémentation informatique

Lors de l'implémentation, d'autres problèmes se posent

- stockage.
- facilité d'implémentation.
- Ré-utilisabilité.

Exemple :

Trois exemples d'implémentations de piles :

- Tableau + 1 indice début, 1 indice fin.
- Une structure de liste simplement chainée.
- Une structure de liste doublement chainée.

Caractériser la qualité de chaque implémentation en C.

Introduction :
Intérêt du module

- Nouvelles structures
- Nouveaux algorithmes
- Nouveaux problèmes

Tableaux associatifs

Un tableau dans lequel un contenu est associé à une clef

- un dictionnaire
- un annuaire

Remarques :

- Un tableau est un tableau associatif a clefs entières.
- Comment faire un dictionnaire en C ?
- Quelle complexite pour la recherche dans cette solution ?

Les tableaux associatifs n'ont pas d'ordre (a priori).
Souvent, leur implémentation en a un !

Objectif :

-accès en temps constant à la donnée connaissant sa clef.

Hash Tables
(tables de hachage)

Une Table de Hachage est une façon d'implémenter un tableau associatif :

On se donne un tableau de taille N?

Pour un couple clef/contenu a insérer dans le tableau :

1. On calcule un entier significatif de sa clef (son hashCode : i)
2. On place le couple clef/valeur a la place tab[i]

Problèmes :

- il faut nécessairement 0< hash(clef) <N
- Comment choisir la fonction hash ?
  peu de collisions.
- Gestion des collisions.

Fonction de Hachage

Peu de collisions : le mieux : une distribution uniforme des clefs...

limitation de taille : résultat du hachage pris modulo N.

Exemple de fonction de hachage pour une chaine :

h(s) = s[0]*B^(n-1) %m + s[1]*B^(n-2) %m + ... + s[n-1] %N

(n longueur de s)

En java, pour une chaine de char, la méthode hashCode utilise B=31 (nombre premier pour éviter interférences entre %m et les B^k)

Fonction de Hachage

Probabilité de collisions :

Lié au Paradoxe des anniversaires...

A partir de combien de personnes la proba qu'au moins deux individus aient le même anniversaire devient elle supérieure a p ?

p = 0.5 : 23 personnes

p = 0.99 : 57 personnes

Démo : calculons la probabilité qu'aucun ne tombe le même jour : !p(n)

- nombre de possibilités d'anniversaires au total : N1 = 365^n
- nombre de possibilités d'anniversaires différents : N2=A(n,365) = 365!/(365-n)!

On a alors : !p(n) = 365!/(365-n)! / 365^n => p(n) = 1- 365!/(365-n)!/365^n

puis fixer p et retrouver n.

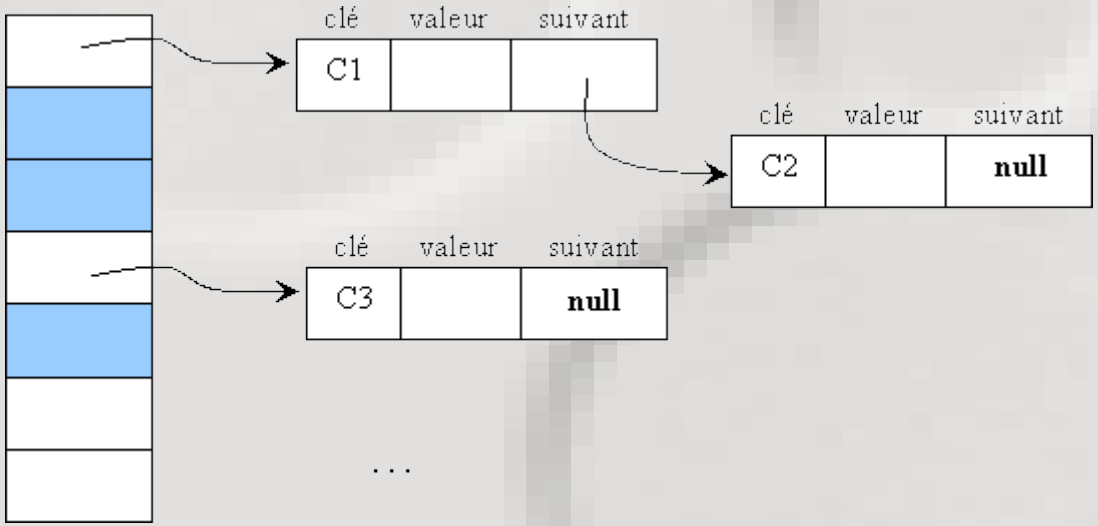
Ex : taille : 1 million, nb clefs : 2500 => p=0.95

=> GERER LES COLLISIONS !!!

Hash Tables
gestions des collisions

Adressage fermé (chainage) :

On arrive directement au bon endroit, reste a lever l'indécision entre les collisions.



Hash Tables
gestions des collisions

Adressage ouvert :

On place tout dans la table :

- Si l'emplacement est libre, on place.
- Sinon, on cherche une place libre

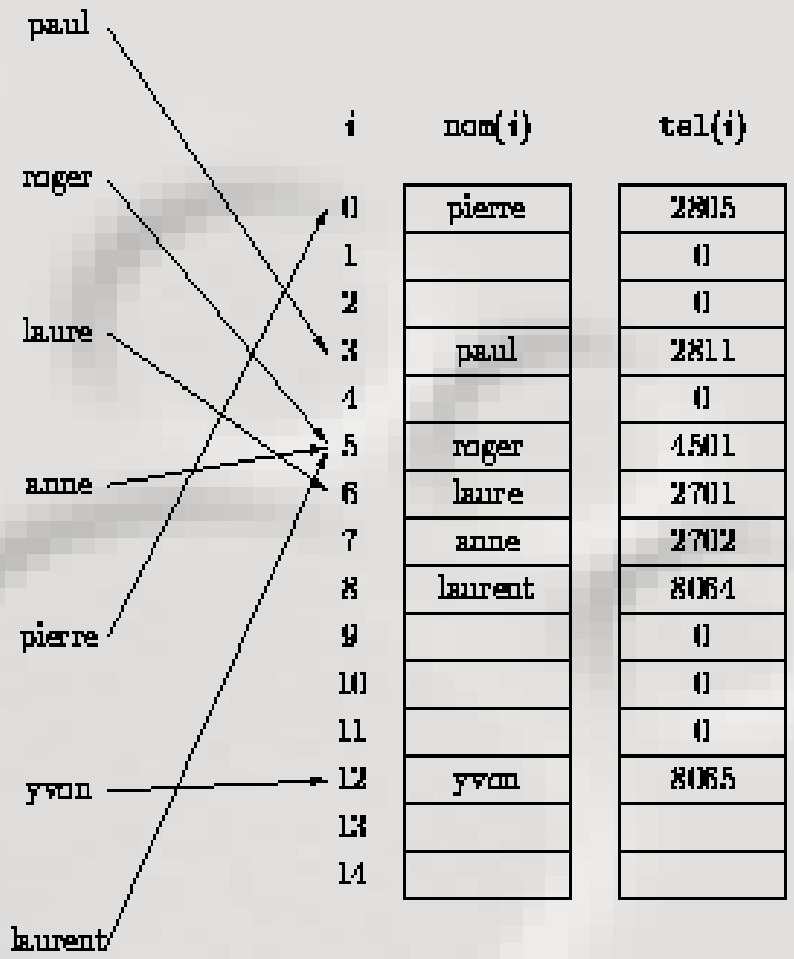
Cette recherche s'appelle le sondage.

Le sondage dépend de la clef k et du nombre de sondages i essayés

La fonction de sondage vérifie

- s(k,0) = h(k)
- la séquence s(k,i) (i in 0...N-1)

est une permutation de (0....N-1)



Hash Tables
gestions des collisions

Sondage Linéaire :

s(k,i)=h(k)+i %N

insertion par sondage linéaire :

i=hash(clef)

fini = false

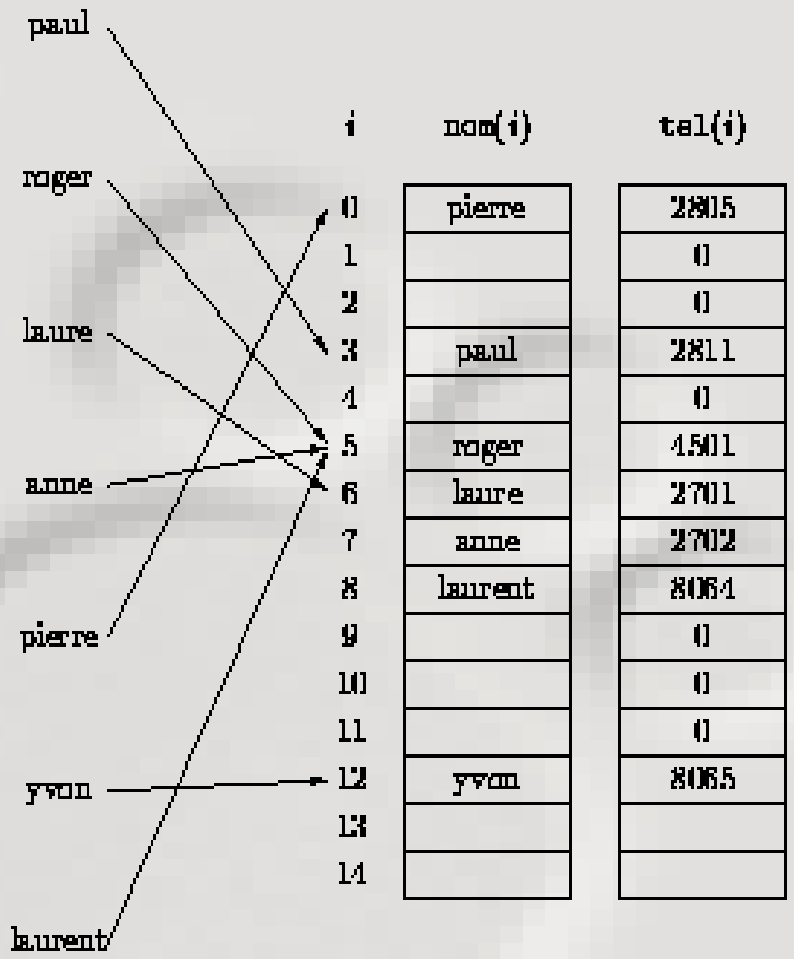
Tant que (! fini)

si (case i vide) insertion et fini.

sinon

i=(i+1)%N;

Ici : on ne traite pas la saturation



Hash Tables
gestions des collisions

Pour disperser un peu les valeurs...

Sondage Quadratique : deux nombres (choisis : premiers) c1 et c2

$s(k,i)=(h(k)+i\ c1 + c2\ i^2) \%N$

Double Hachage : une seconde fonction de hachage (choisie) h'

$s(k,i)=(h(k)+i\ h'(k)) \%N$

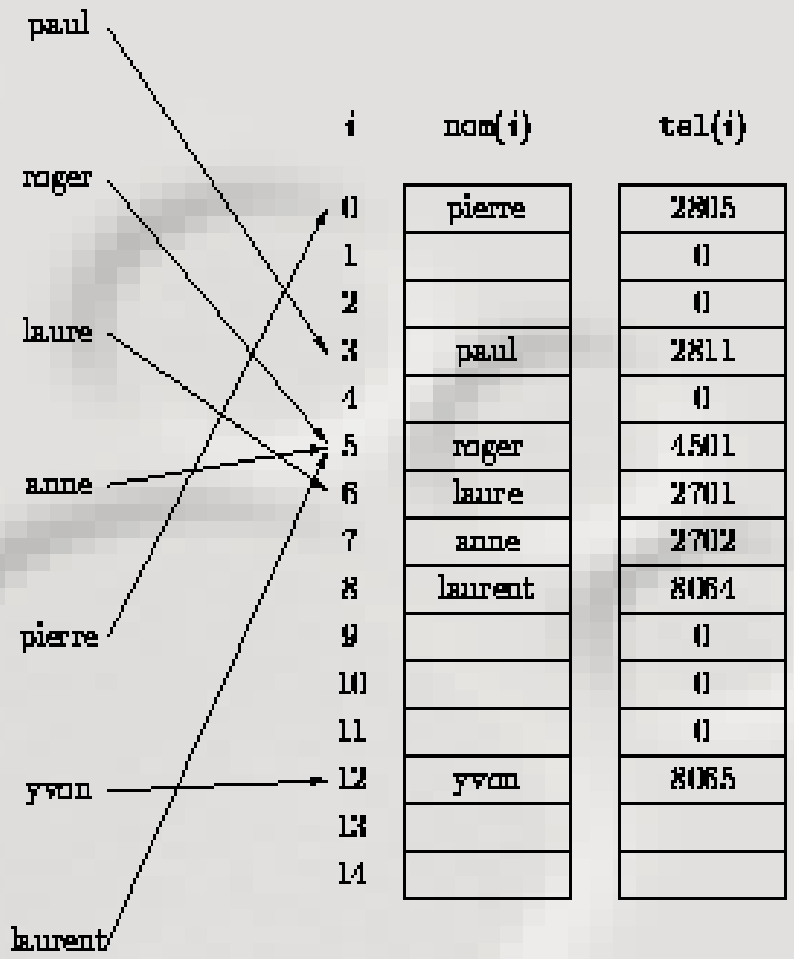
Hash Tables
gestions des collisions

Table à adressage ouvert par sondage lineaire :

Comment faire une recherche ?

Comment faire une suppression ?

Comment faire une recherche et une insertion si la suppression est possible ?



Hash Tables
Importance des collisions

Temps de recherche / insertion augmente avec le nombre de collisions.

Le nombre de collisions est lié à la charge (taux de remplissage) de la table. Ceci s'obtient par des probas.

En général, on fixe un taux de remplissage max, et la taille de la table augmente passé ce taux. (on replace alors les valeurs et on change la fonction de hash....)

Hash Tables
Augmentation de taille

Il faut preparer une table plus grande, et re inserer chaque valeur dedans...Comme la position depend de la taille, il faut repositionner (recalculer son hashcode) chaque élement dedans

=> long mais on ne peut pas trop faire autrement.

=> Pour accelerer un peu, on peut prendre la nouvelle taille multiple de la precedente (on ne deplace pas toujours les éléments)

Graphes
Rappels

Qu'est ce que :
- Un graphe ?
- Un graphe orienté ?
- Un graphe valué ?
- Un chemin ?
- Un parcours de graphe ?
- Composantes connexes ?

Dans quels cas cela sert il ?

Graphes
Implémentations

- Listes d'incidences

- Matrices d'adjacence

Graphes
Parcours largeur d'abord

Breadth First Search :

Exemple.

Algo.

Graphes

Parcours profondeur d'abord

Depth First Search :

Exemple.

Algo.

Graphes
Existence d'un chemin

Comment faire ?

=> Connexite

Graphes
Plus court chemin

Dans un graphe non valué ?

Dans un graphe valué (valeurs entieres ?)

Dans un graphe valué a valeurs réelles positives => Dijkstra

Graphes
Dijkstra

Principe, preuve

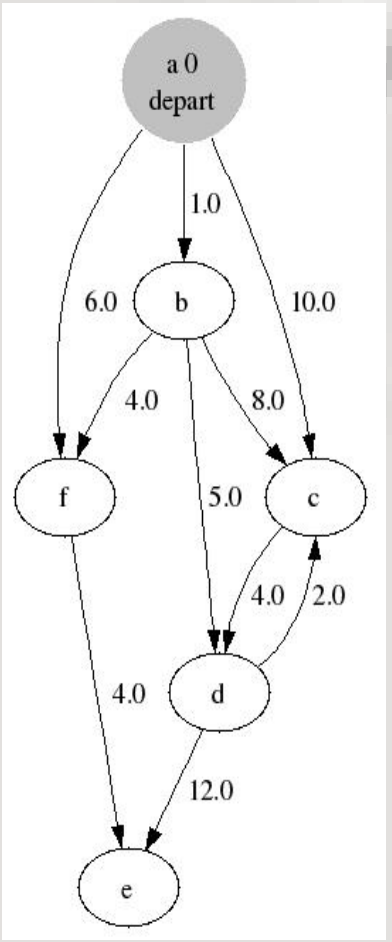
Graphes
Dijkstra

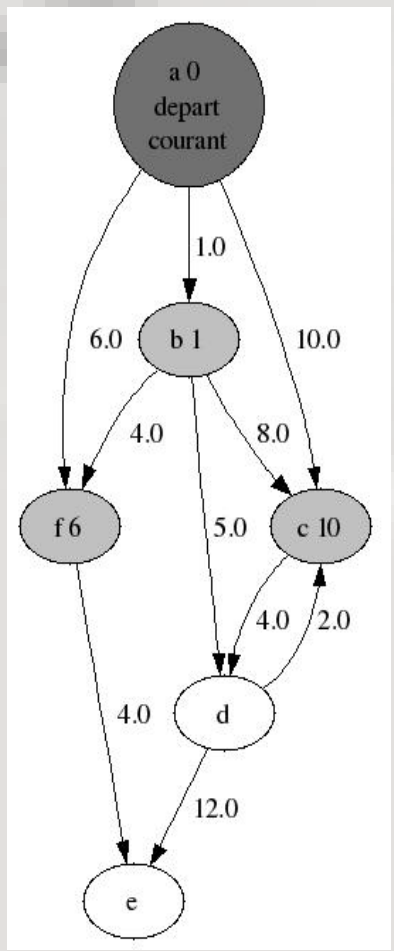
L'algorithme suppose de pouvoir :
- marquer les Sommets : aFaire ou dejaFait (et connaître leur état)
- définir et retrouver la distance du départ à chaque sommet.

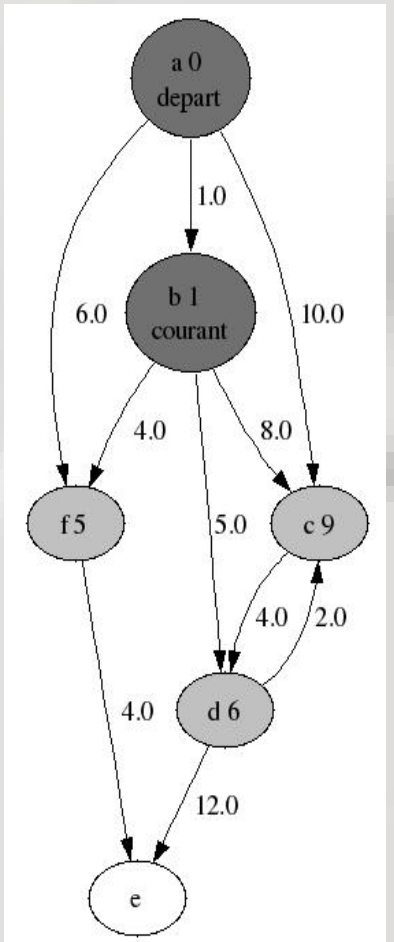
```
dist(depart) <- 0
marquer depart aFaire.
Tant que ( sommetsAFaire non vide)
        courant <- extractMin(sommetsAFaire); // choix du sommet de distance min
        Pour tous les successeurs s de courant
            Si ( (s !dejaFait) et (s !aFaire) )
                marquer s aFaire
            Si (s !dejaFait)
                Si  ( dist(s)> dist(courant) + poid (courant,s) )
                                dist(s) <- dist(courant) + poid (courant,s)
```

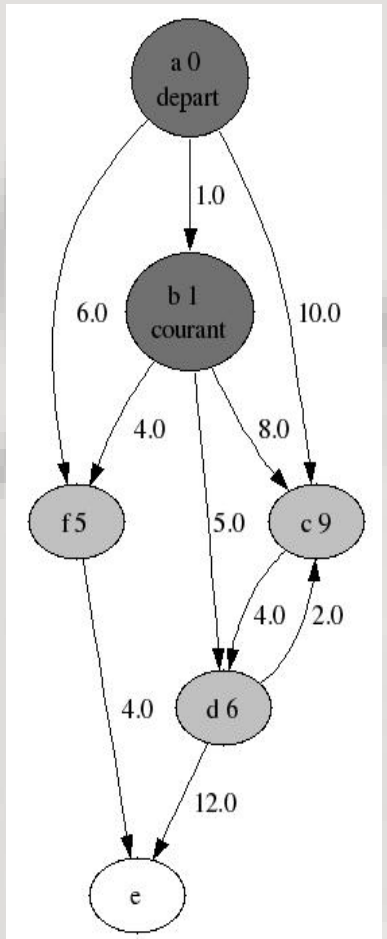
Graphes
Exemple Dijkstra

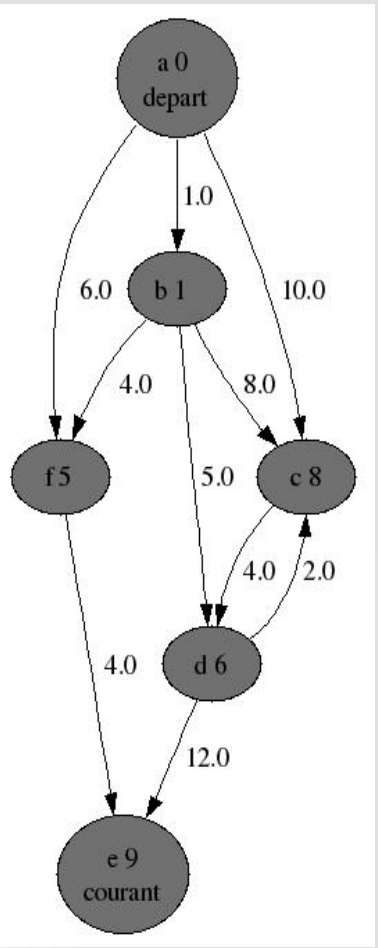
Initialisation step 0 step 1 step 2 ... ...fin (step6)











Graphes
Dijkstra

Remarques sur l'implémentation :

En fonction des cas 2 possibilités (et des mixes des deux) :

- Un sommet contient deux variables : distance et couleur
- On dispose de structures pour retrouver la distance des sommets et les sommets dejaFaits et aFaire

Chaque solution a des avantages et des inconvénients :

solution 1 :

++ Facile a mettre en place.

++ Peu gourmand en place mémoire.

-- Extraction du sommet courant couteuse.

solution 2 :

-- Plus pénibles a mettre en place

-- Gourmand en mémoire

++ Choix des structures optimisées (HashSet, Heap...)

++ Ne demande pas de modifier le code des sommets !

Flots
Introduction

Les problèmes de flots s'intéressent à l'acheminement de "matière" dans un réseau.

En général, il s'agit de problèmes de flot Maximum :
acheminer un maximum de "matière" entre deux points du réseau.

Utilité :

- Logistique (transport d'un produit entre fabricants et vendeurs)
- Ingénierie réseaux électriques, hydrologiques....
- Formaliser d'autres problèmes qui trouvent ainsi une solution

Dans ce que l'on verra ici, tout élément de matière doit être interchangeable avec tout autre (non immédiatement adapté au transport de personnes qui ont une destination particulière et une origine particulière)

Flots
Formalisme (réseau / flux)

Ils se formalisent en :

- Un **réseau de transport** : un graphe orienté valué + source + puit
  - . chaque noeud est un point du réseau.
  - . la valeur de chaque arête est la capacité de transport de l'arete.
  - . il existe au moins un chemin entre la source et le puit.
  - . En logistique, la source est le fabricant, le puit le consomateur
- Un **flux direct** : une application de V^2 dans R+

  Pour chaque couple de sommet *(x,y)* du réseau, c'est la quantité de matière circulant sur l'arc *(x -> y)*.

  Il faut de plus :

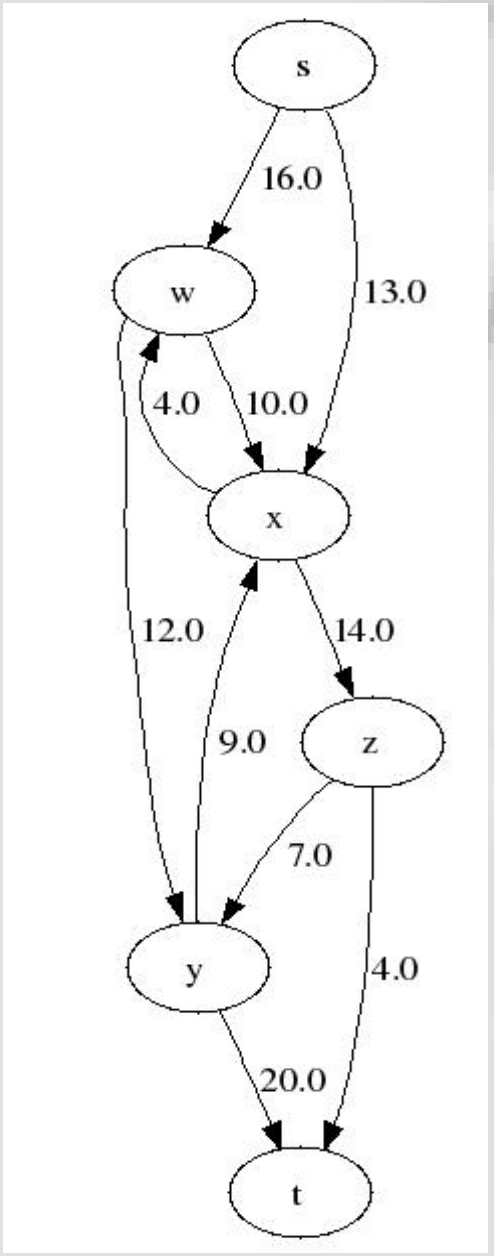
  . fluxDirect (x,y) <= C(x,y) : le flux direct est inférieur à la capacité de l'arête.

  . fluxDirect (x,y) >= 0 : le flux direct est positif.

  . Si l'arete n'existe pas, le flux direct est nul.

  . Conservation de la matière (Kirshoff) : pour chaque sommet sauf source et puit, la somme des flux directs des arêtes qui le contiennent est nulle.

Flots



Exemple de Réseau

chaque poids est la capacité
du réseau pour l'arête associée.
La source est s, le puit t.

Exemple de Flot

Pour chaque arête:

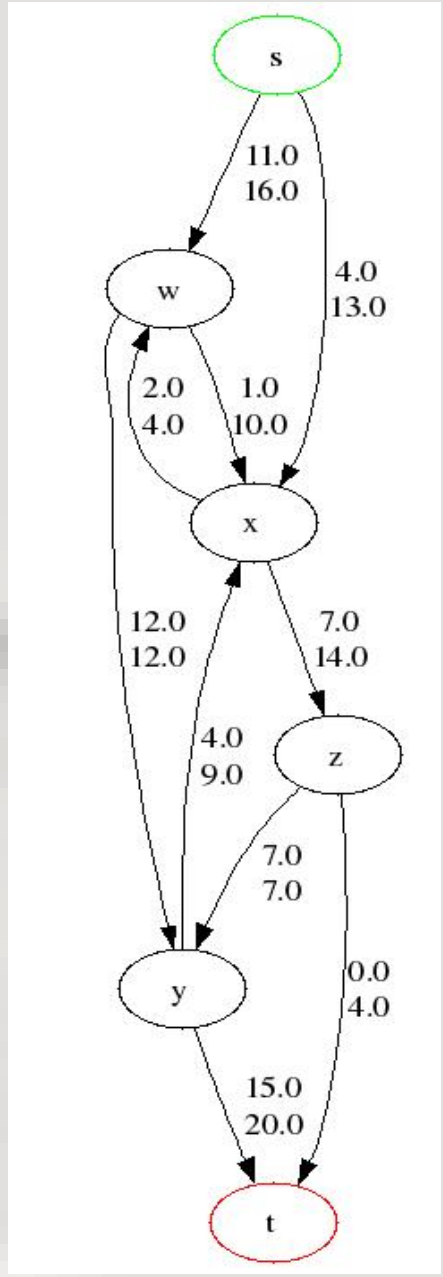
. le chiffre du haut est le flux direct associé a l'arète.

. le chiffre du bas est la capacité de l'arête dans le réseau.

. Il y a bien conservation de matière.

Exemple :La source est un fabricant de produit. Le puit est le centre commercial local. Le réseau est l'ensemble des routes et le nombre de camions de produits qu'elles supportent quotidiennement.

Avec le flot de droite, quelle quantité est livrée ? 15



Flots
Simplification multi-sources/puits

Imaginons un problème de logistique plus complexe :

- 2 producteurs de produits Pi (chacun produit pi produits par jours)
- 2 centres de distribution Ci (chacun peut vendre ci produits par jours)
- Un réseau routier permettant d'acheminer ces produits.

Ceci correspondrait a 2 sources / 2 puits que l'on ne verra pas...

On peut se ramener au problème a un puit/ une source comme suit :
- On utilise le réseau routier comme réseau de transport sans puit ni source
- On crée une source (super source / méta source) reliée à chaque producteur par une arête de poid pi, dans le sens source producteur
- On crée un puit (super/méta puit) relié a chaque consommateur par une arete de poid ci dans le sens consommateur / puit.

On est alors ramené au problème précédent... voilà !

Flots
Exemple de simplification

Dessin a rajouter !

Flots
Formalisme suite

Reprenons :

- Un **flux direct** : une application de V^2 dans R+

Pour chaque couple de sommet *(x,y)* du réseau, c'est la quantité de matière circulant sur l'arc *(x -> y)*.

Il faut de plus :

. fluxDirect (x,y) <= C(x,y) : le flux direct est inférieur à la capacité de l'arête.

. fluxDirect (x,y) >= 0 : le flux direct est positif.

. Si l'arete n'existe pas, le flux direct est nul.

. Conservation de la matière (Kirshoff) : pour chaque sommet sauf source et puit, la somme des flux directs des arêtes qui le contiennent est nulle.

Problemes de Flots Max

Il s'agit de trouver le flot maximum pour un Réseau.

On cherche le moyen d'acheminer un maximum de matière de la source au puit.

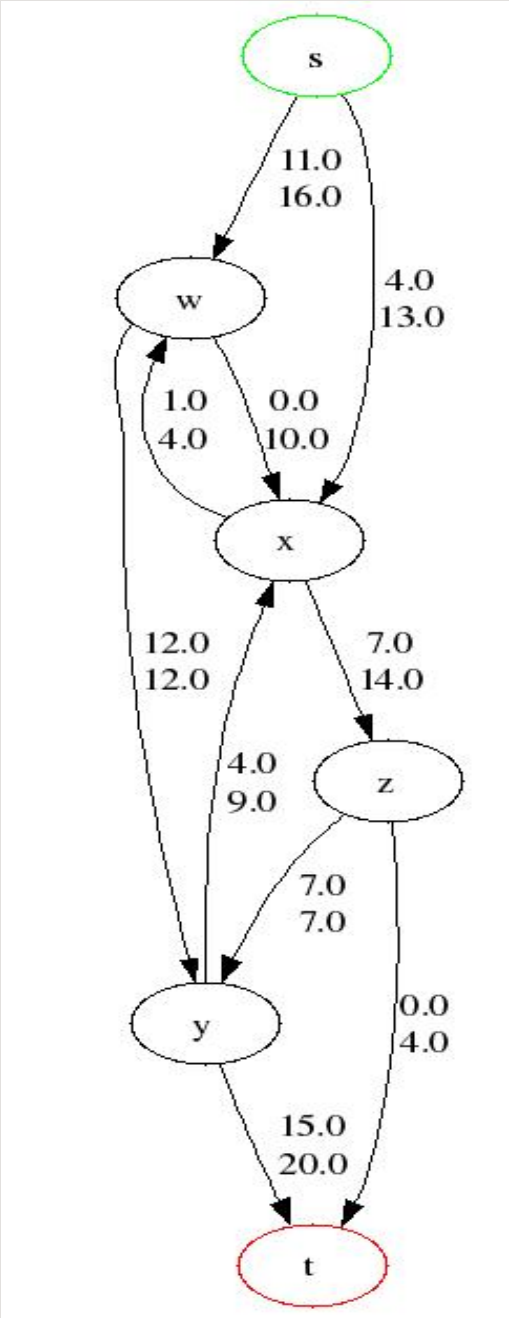
Idée (ne marche pas) :

Partir d'un flot et l'améliorer en cherchant des chemins non saturés allant de la source au puit.

(un chemin est non saturé si le flux direct de chacun de ses arcs est < a la capacité de l'arc).

Pb : on risque de se bloquer pour la suite en choisissant un chemin.

Exemple



Flot de départ :

sa valeur est 15.

un chemin non saturé est

(s,x) (x,z) (z,t)

On peut l'augmenter de 4.0

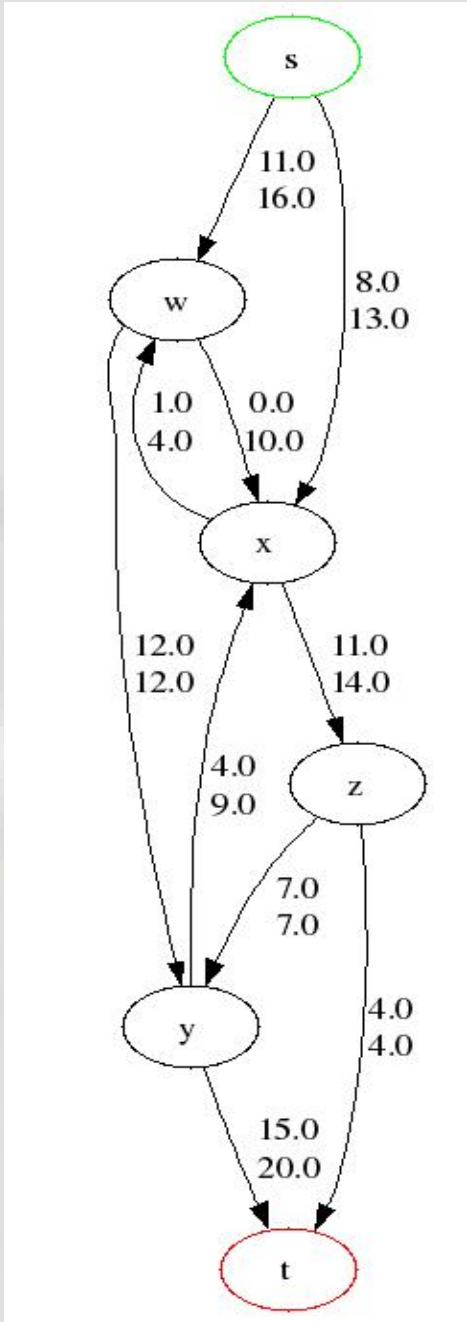
ce qui donne le flot de droite.

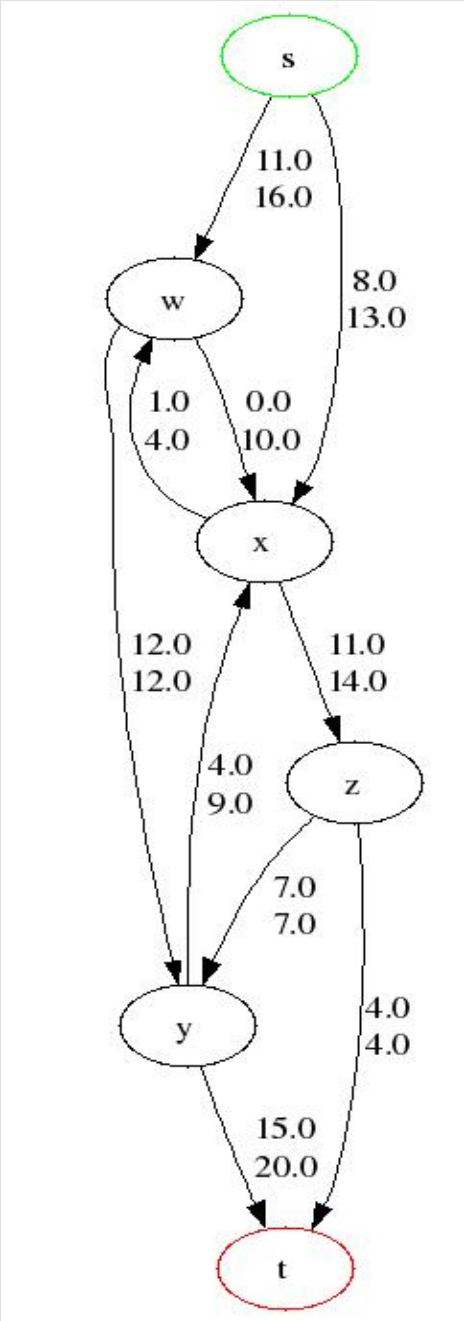
Flot de fin :

il n'existe plus aucun chemin non saturé allant de s a t....

Ce Flot a une valeur de 19

On fera mieux plus tard !
le problème est (x,y)





Exemple

Flot de départ :

sa valeur est 19.

Il ne faut plus s'intéresser au flux direct sur un arc mais a la quantité de matière acheminée entre deux sommets.

Par exemple, aucune matière réelle ne circule de x a y.

En revanche, -4 unités de matière sont acheminées de x à y, ce qui ne nous arrange pas vraiment...

Flots
Définitions

Flux entre un sommet *x* et un sommet *y* :

Quantité de matière acheminée dans le sens *x* -> *y*.

flux(x,y) = fluxDirect (x,y) - fluxDirect (y,x)

On travaillera ensuite le plus souvent avec les flux et non plus les flux directs.

On a toujours *flux(x,y) = - flux(y,x)* Ce qui n'est pas vrai avec les flux directs.

Flux sortant d'un sommet : Quantité de matière acheminée hors du sommet.

C'est la somme des flux positifs entre le sommet et ses **voisins** dans le réseau.

Flux entrant d'un sommet *x* : Quantité de matière acheminée vers le sommet.

c'est la somme des flux négatifs entre le sommet et ses **voisins** dans le réseau (l'opposé de la somme).

D'après la loi de kirshoff, le flux entrant est égal au flux sortant.

Valeur du flot : Quantité de matière acheminée de la source au puit.

C'est le flux sortant de la source. C'est aussi le flux entrant du puit (kirshoff).

Augmentation de flux

Augmenter le Flux entre un sommet *x* et un sommet *y* :

Deux moyens :

- sens direct : On augmente le flux direct entre x et y (jusqu'à la capacité).
- sens indirect : On diminue le flux direct entre y et x (jusqu'à zéro)

l'augmentation max dans le sens direct est : $augMaxDirect = C(x,y) - fluxDirect(x,y)$ $(>=0)$

l'augmentation max dans le sens indirect est : $augMaxIndirect = fluxDirect(y,x)$ $(>=0)$

l'augmentation max total du flux est : $augMax = augMaxDirect + augMaxIndirect$ $(>=0)$

Augmenter le flux entre x et y de delta : Si delta > augmentationMax : Impossible.

Sinon, une méthode qui augmente au plus le flux direct x,y) :

```
// On determine l'augmentation a faire dans le sens direct :
if (delta < augMaxDirect)     augDirect = delta.
else                          augDirect = augMaxDirect.
fluxDirect (x,y) +=delta
// On met a jour delta : on a deja incremente de augDirect
delta -= augDirect;
if (delta > 0)
        fluxDirect(x,y) -= delta
```

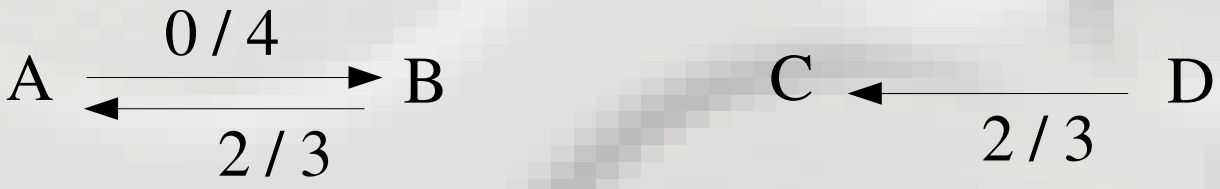
Augmentation de flux

Augmentation du flux entre de sommets :

- Remarques

- On ne peut pas augmenter le flux entre deux sommets indépendamment des autres, a cause de Kirshoff.
- La façon d'augmenter le flux est sans importance du moment qu'elle respecte les règles (flux directs positifs et inférieurs aux capacités)

- Exemples



Augmenter flux (A,B) de 5 ? A->B : 4 et B-> A : 1 ou A->B : 3 et B-> A : 0 ou.....

Augmenter flux (C,D) de 3 ? Non : augMax : 2

Augmenter flux (C,D) de 2 ? D->C : 0

Augmentation de flux

Augmenter le Flux sur un chemin de la source au puits :

C'est augmenter le flux d'une même quantité sur chaque arete du chemin.
Parcourir toutes les arêtes du chemin et augmenter.

De combien peut on l'augmenter au max ? le minimum des augMax de chaque arête du chemin. Si cette valeur est positive, le chemin est un **chemin améliorant**.

- Si le chemin va de la source au puit, on respecte Kirshoff.
- Si on augmente comme indiqué avant, on respecte les contraintes de capacité du réseau.

=> Le Flot reste correct et a une valeur plus importante.

Chemin augmentant

Précisions sur les chemins augmentant :

il n'est pas nécessaire qu'une arête *(x,y)* existe dans le réseau pour que l'on puisse augmenter le flux entre x et y.

il faut par contre que :

-x et y soient voisins

-le flux(x,y) peut etre augmenté d'une valeur > 0 (augMax)

Pour trouver un chemin augmentant :

1.construire un graphe orienté valué annexe : le **réseau résiduel** :

- Ses sommets sont les sommets du réseau.
- Une arete existe si f(x,y) peut être augmenté de augMax (>0)
- Le poid d'une arête est ce augMax.

2. prendre un chemin entre s et t. Exemple : plus court chemin entre s et t.

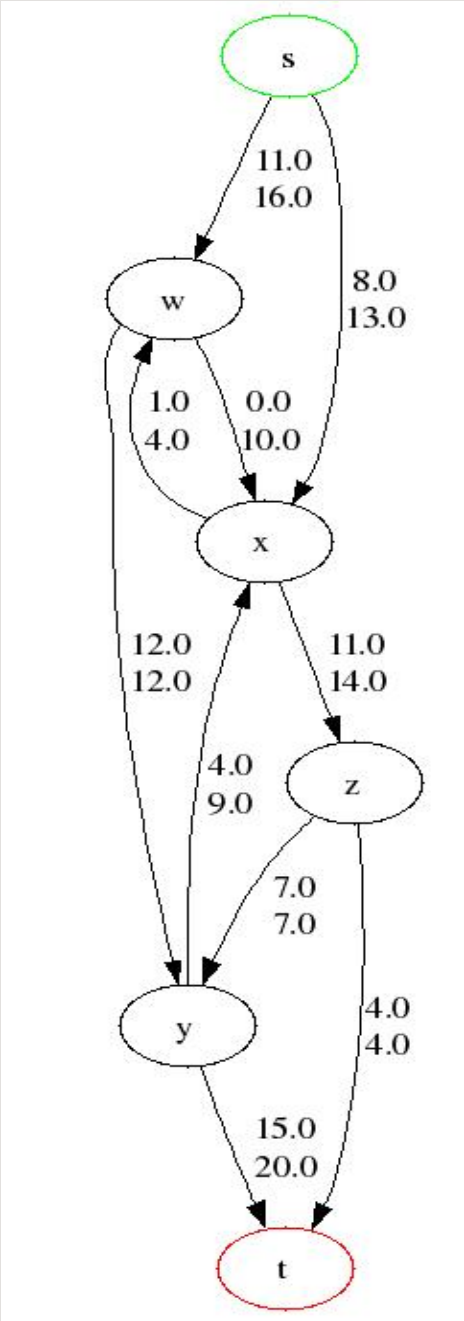
Flot
Ford Fulkerson

Tous les algos fonctionnent sur le principe suivant de Ford Fulkerson.
Les implémentations varient dans le choix des chemins améliorants.

1. Départ du flot nul
2. Recherche de chemin améliorant entre la source et le puit.
3. Augmentation du flux au maximum le long de ce chemin.
4 Boucler en 2 tant qu'un chemin améliorant existe.

Cela fonctionne car :
- le flux augmente a chaque itération (ne diminue pas)
- Le flux obtenu est maximum car il exhibe une coupe dont la capacité est la valeur du flot. (voir plus loin max cut / min flow)

Exemple

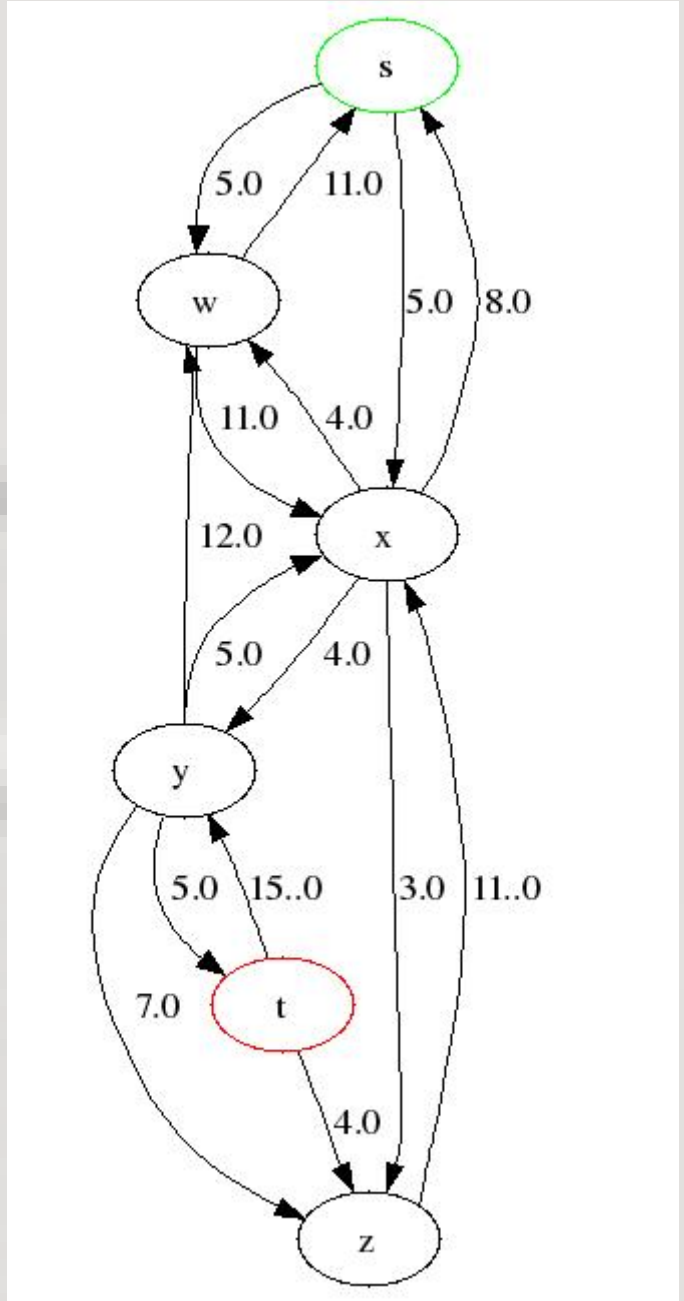


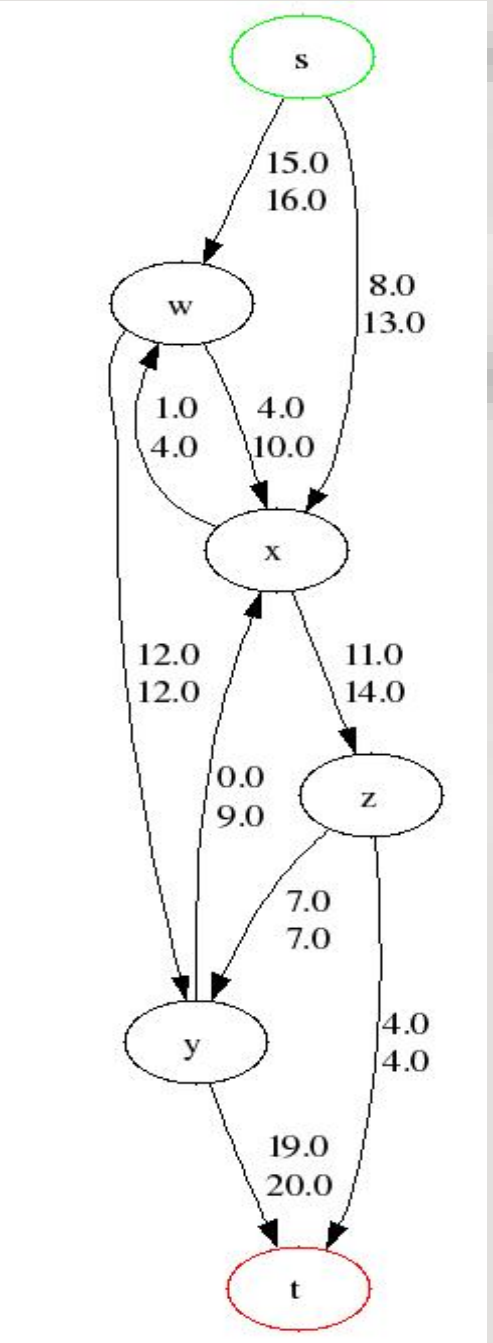
Flot précédent

réseau résiduel

Ce graphe fait apparaître un chemin entre s et t (s,x)(x,y)(y,t) de valeur 4

On devrait obtenir ainsi un flot de 23.





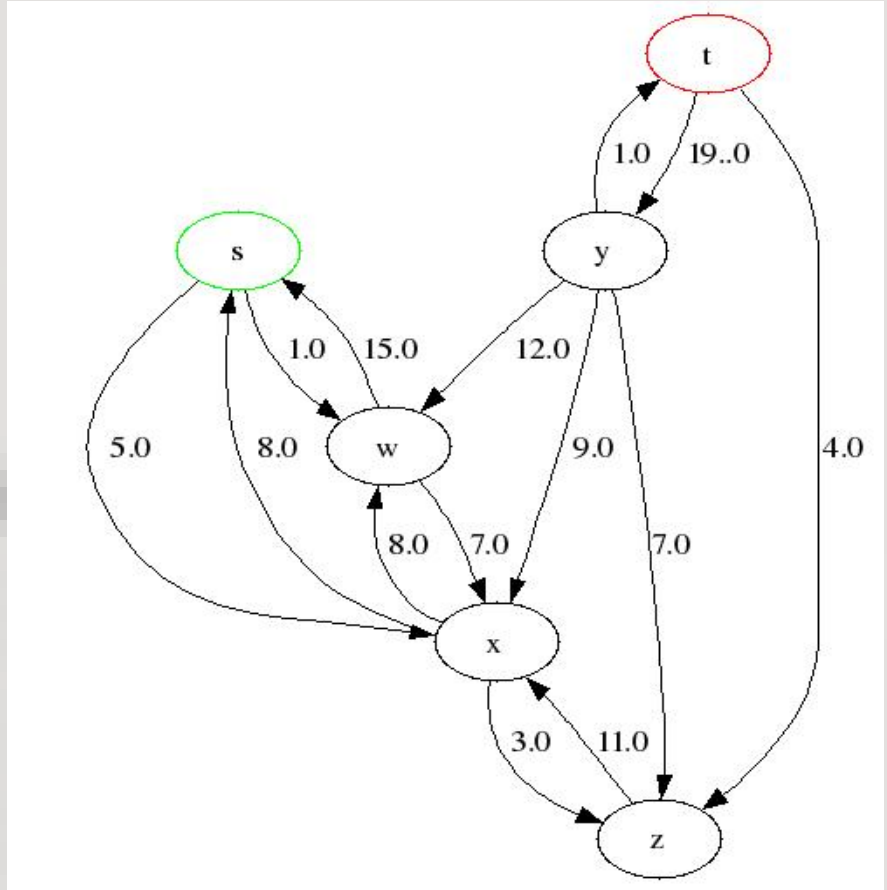
Exemple

Flot augmenté

valeur 23

réseau résiduel

Plus de chemin de s à t



Flots
Remarques d'implémentation

Pour implémenter les flots, une solution (la mienne) est d'utiliser :

- Un graphe orienté valué (le réseau).
- Un graphe orienté valué (le réseau résiduel)
- le flot : association entre les arcs du réseau et leur flux direct. : une Hashtable entre un arc et le flux direct associé.

- Pour le réseau, il est important de pouvoir connaître les voisins d'un sommet, pas seulement ses successeurs. Le graphe doit pouvoir faire cela rapidement :
    - matrice d'adjacence
    - liste d'adjacence + listes d'incidences. (implémentation de la classe GrapheOriente)

Coupes et Flots

a finir

en gros : un flot est toujours de valeur inférieure a la capacité d'une coupe.
(Facile par récurrence.)
Donc le flot max est de valeur inférieure à la capacité de la coupe Min (évident)

Donc si on obtient une coupe de valeur égale a celle du flot, c'est le flot max..

Ford Fulkerson y arrive : lorsque le réseau residuel ne présente aucun chemin allant de la source au puit, cela constitue une coupe. On montre (a faire) que celle ci est de la valeur du flot.

CQFD (ce qu'il faudrait démontrer.....)