IUT Belfort Montbéliard

Introduction à la
RECHERCHE OPERATIONNELLE

Partie 2
Graphes

Karine DESCHINKEL

Problèmes types

- Problème de cheminement (rechercher l'itinéraire le plus court)
- Problème d'ordonnancement (trouver l'ordre des tâches à accomplir pour réaliser un projet)
- Problème de flot maximal dans un réseau de transport

  (comment acheminer le plus de produits d'un point à un autre dans un réseau avec des capacités)
- Problème d'affectation (exemple : 25 personnes à affecter sur 25 postes, chaque personne donne ses préférences; trouver une solution satisfaisant le plus de personnes possibles en minimisant la somme des classements du poste de chaque personne)
- Problème de transport (acheminer des produits entre des usines et des dépôts à coût minimal)
- Problème du voyageur de commerce
- Problème du sac à dos
- Problèmes linéaires

Modèles types

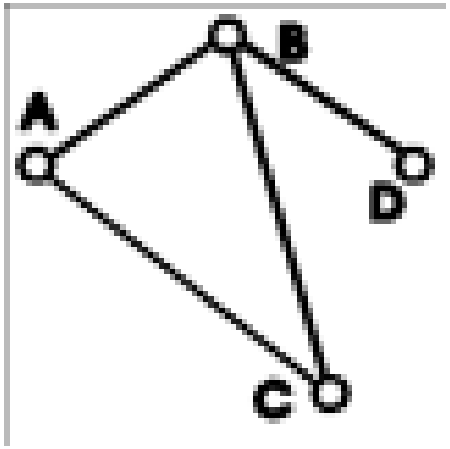
- GRAPHES
- PROGRAMME LINEAIRE
- PROGRAMME NON LINEAIRE
- AUTRES

Notions de graphe

- **Un graphe est un ensemble de points, dont certaines paires sont reliées par des lignes.**
- **Les points sont appelés sommets et les lignes sont nommées arêtes.**
- **Plus formellement, un graphe est composé de deux ensembles, l'ensemble des arêtes (E) et l'ensemble des sommets (V).**
- **Les sommets portent un nom (étiquette).**
- **L'ensemble des arêtes est constitué de paires non ordonnées de sommets.**

V={A, B, C, D}
-- l'ensemble des sommets

E={(A,B) , (A,C) , (B,C) , (B,D)}
-- l'ensemble des arêtes

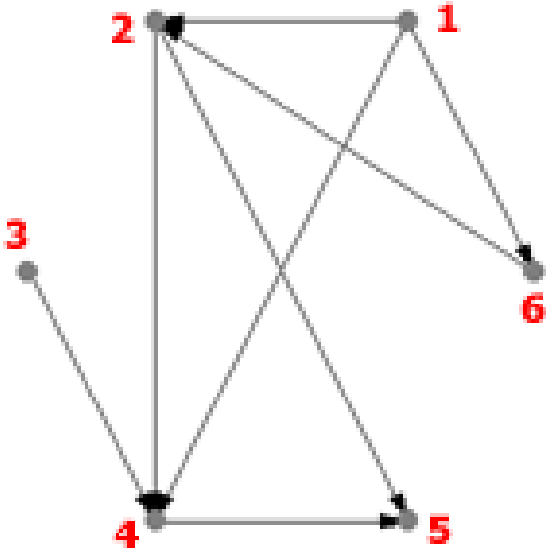


Notions de graphe

- **En donnant un sens aux arêtes (avec une flèche), on obtient un graphe orienté.**
- **Les arêtes sont appelés arcs.**
- **L'ensemble des arcs est constitué de paires ordonnées de sommets.**

V={1, 2, 3, 4, 5, 6}
-- l'ensemble des sommets

E={(1,2) , (1,4), (1,6) , (2,4) , (2,5), (3,4), (4,5), (6,2)}
-- l'ensemble des arcs



Notions de graphe

Modélisation sous forme de graphe :

- le graphe du web peut être modélisé par un graphe (*V*,*E*) de la manière suivante:

  les sommets sont des pages web, Etant données 2 pages web *a* et *b*, il existe un arc (*a*,*b*) dans *E* si et seulement s'il existe un lien hypertexte dans la page *a* qui pointe vers la page *b*.

- Un réseau routier peut se représenter comme un graphe (non orienté, sauf si l'on tient compte d'éventuelles voies à sens unique) dont les sommets sont les villes et les arêtes les routes qui mènent directement d'une ville à une autre sans passer par une ville intermédiaire.

- Le réseau internet est un graphe dont les sommets sont les serveurs et les utilisateurs et les arêtes les différentes interconnexions.

Notions de graphe

EXERCICE

Construire un graphe orienté dont les sommets sont les entiers compris entre 1 et 12 et dont les arcs représentent la relation «être diviseur de».

Plus court chemin dans un graphe

- **Problème :** Lorsqu'un chemin existe entre deux sommets dans un graphe, l'être humain se pose rapidement la question non seulement de trouver un tel chemin, mais bien souvent il est intéressé par le *plus court chemin* possible entre ces deux sommets. Notre oeil est d'ailleurs particulièrement efficace dans cette tâche, tant que le graphe est de taille raisonnable...

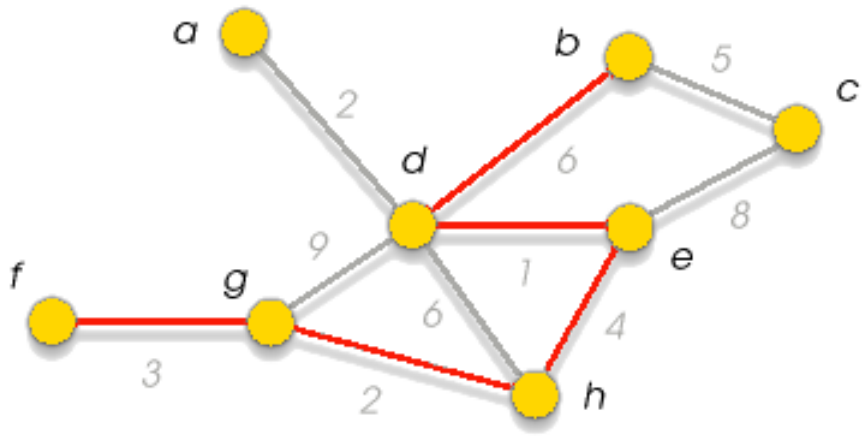
Mais dès que le graphe comporte plusieurs dizaines de sommets et d'arêtes, trouver le plus court chemin entre deux points devient vite un casse-tête : quel est l'itinéraire entre Ménilmontant et la Rue du Bac passant par le minimum de stations de métro?

Plus court chemin dans un graphe

- **Modèle : un graphe avec une valeur sur chaque arête (distance entre 2 sommets)**

Dans le graphe valué ci-contre, ***p=(f, g, h, e, d, b)*** a une longueur de 16.

Il est le plus court chemin reliant les sommets ***f*** et ***b***



Plus court chemin dans un graphe

► Algorithme de résolution :

- **Énumérer tous les chemins possibles entre le sommet origine et le sommet destination, et calculer leurs longueurs. Retenir le chemin le plus court.** **Si le nombre de chemins élémentaires est bien fini, il n'en demeure pas moins très grand, de l'ordre de *n!* sur un graphe d'ordre *n*. Si nous retenons cette estimation, sur un graphe de seulement 20 sommets, en supposant que trouver un chemin et calculer sa longueur puissent se faire en une seule opération, sur un ordinateur pouvant effectuer 1 milliard d'opérations par seconde, il nous faudra patienter 77 ans pour obtenir le meilleur trajet. Et si le graphe a 25 sommets, armons nous de patience pour les ... 490 millions d'années à venir!**

- **Algorithme de Dijkstra (valable pour valeurs d'arêtes positives)**

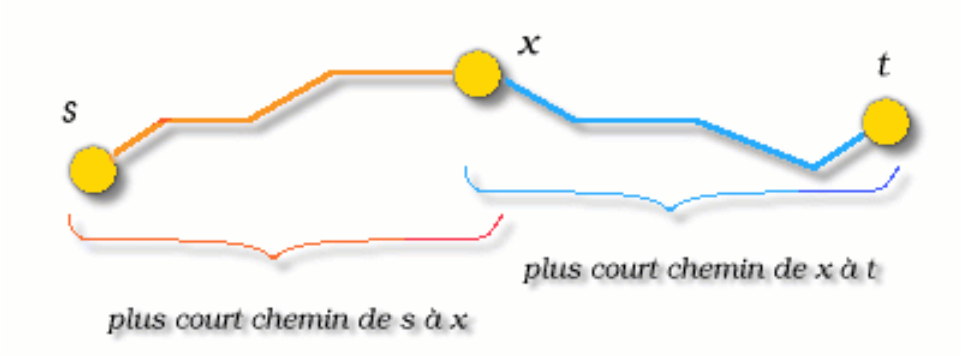
Plus court chemin dans un graphe

► Algorithme de résolution :

Principe de sous-optimalité

Si *p=(s,...,t)* est un plus court chemin entre *s* et *t*, alors
pour tout sommet *x* sur le chemin *p*,

- **le sous-chemin de *p* jusqu'à *x*, *(s,...,x)*, est un plus court chemin de *s* à *x***
- **le sous-chemin de *p* depuis *x*, *(x,...,t)*, est un plus court chemin de *x* à *t***



Preuve.

La preuve utilise un argument d'échange. Par l'absurde supposons qu'il existe par exemple entre *s* et *x* un chemin *q* de longueur strictement inférieur à celle de *(s,...,x)*. Alors nous pouvons construire un nouveau chemin *p'* entre *s* et *t* en subsituant *q* à *(s,...,x)* dans le chemin *p*. La longueur de *p'* est alors strictment inférieure à celle de *p*, ce qui contredit que *p* est un plus court chemin de *s* à *t*.

Plus court chemin dans un graphe

► Algorithme de résolution :

Si *D(y)* est la longueur du plus court chemin d'un sommet *s* à *y*, le principe de sous-optimalité se traduit localement par les égalités :

D(y) = 0 si y=s

D(y) = min { D(x) + c(x,y) | x voisin de y } sinon

Plus court chemin dans un graphe

```
ALGORITHME Dijkstra
ENTREES G=(V,E)graphe avec une valuation positive c des arêtes, s un sommet de V
Initialiser tous les sommets à non marqué ;
Initialiser tous les labels L à +infini
L(s) := 0
Tant Que il existe un sommet non marqué
        // Sélection du plus 'proche' sommet non marqué
        Choisir le sommet y non marqué de plus petit label L
        Marquer y
        // Mise à jour des labels de ses voisins non marqués
        Pour chaque sommet z non marqué voisin de y
                L(z) := min{ L(z) , L(y) + c(y,z) }
        Fin Pour
Fin TantQue
```

Pour tout x de V, longueur du plus court chemin de s à x : D(x)=L(x)

Plus court chemin dans un graphe

► Exemple :

Une phase de sélection

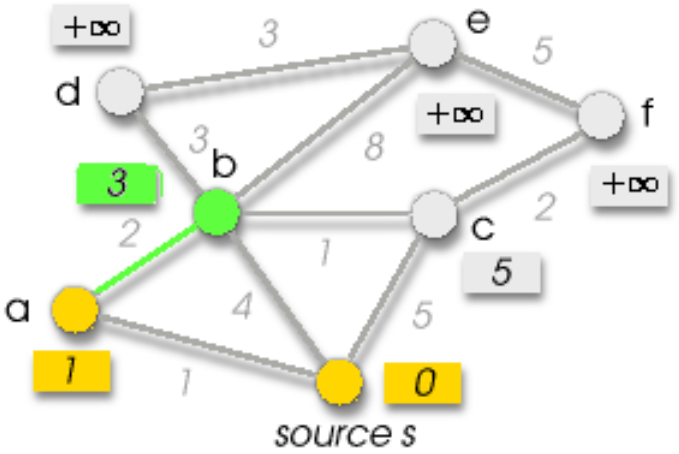
Le sommet ***y*** non marqué de plus petit label est sélectionné.

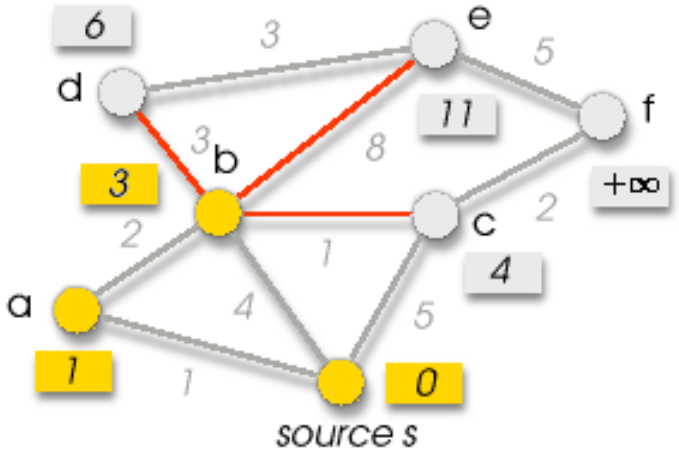
Sur l'exemple le sommet ***b*** de label **3** est le sommet sélectionné à la deuxième étape.

Une phase de mise à jour

Les labels des sommets ***z*** non marqués adjacents au sommet ***y*** qui vient d'être sélectionné sont mis à jour :

L(z) = min{ L(z), L(y) + c(y,z) } Sur l'exemple les labels des sommets c, d et e sont mis à jour.





Ordonnancement

- **Problème : Un problème d'ordonnancement consiste à organiser dans le temps la réalisation de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement) et de contraintes portant sur la disponibilité des ressources requises.**
- **Tâche : une activité à réaliser avec une durée**
- **Ressource : moyen technique ou humain utilisé pour réaliser une tâche**
- **Exemples :** - ordonnancement des voitures dans l'atelier (montage+peinture)

  - ordonnancement d'un projet pour la construction d'un immeuble

- **Un ordonnancement constitue une solution au problème d'ordonnancement. Il est défini par le planning d'exécution des tâches (« ordre » et « calendrier ») et d'allocation des ressources et vise à satisfaire un ou plusieurs objectifs.**

Ordonnancement

- **Exemple facile (pas de contraintes de ressources et de temps) :**

Les tâches suivantes sont à effectuer lors de la préparation d'un repas:

- ✓ **A : préparer le menu (30 min)**
- ✓ **B : acheter les ingrédients (90 min), après A**
- ✓ **C : préparer l'apéritif (30 mn) , après B**
- ✓ **D : nettoyer la table (10 min)**
- ✓ **E : mettre la table (10 min), après D**
- ✓ **F : préparer les ingrédients (30 min) , après B**
- ✓ **G : cuisiner les plats (60 min), après F**
- ✓ **H : servir le repas (10 min) , après E et G**

Ordonnancement

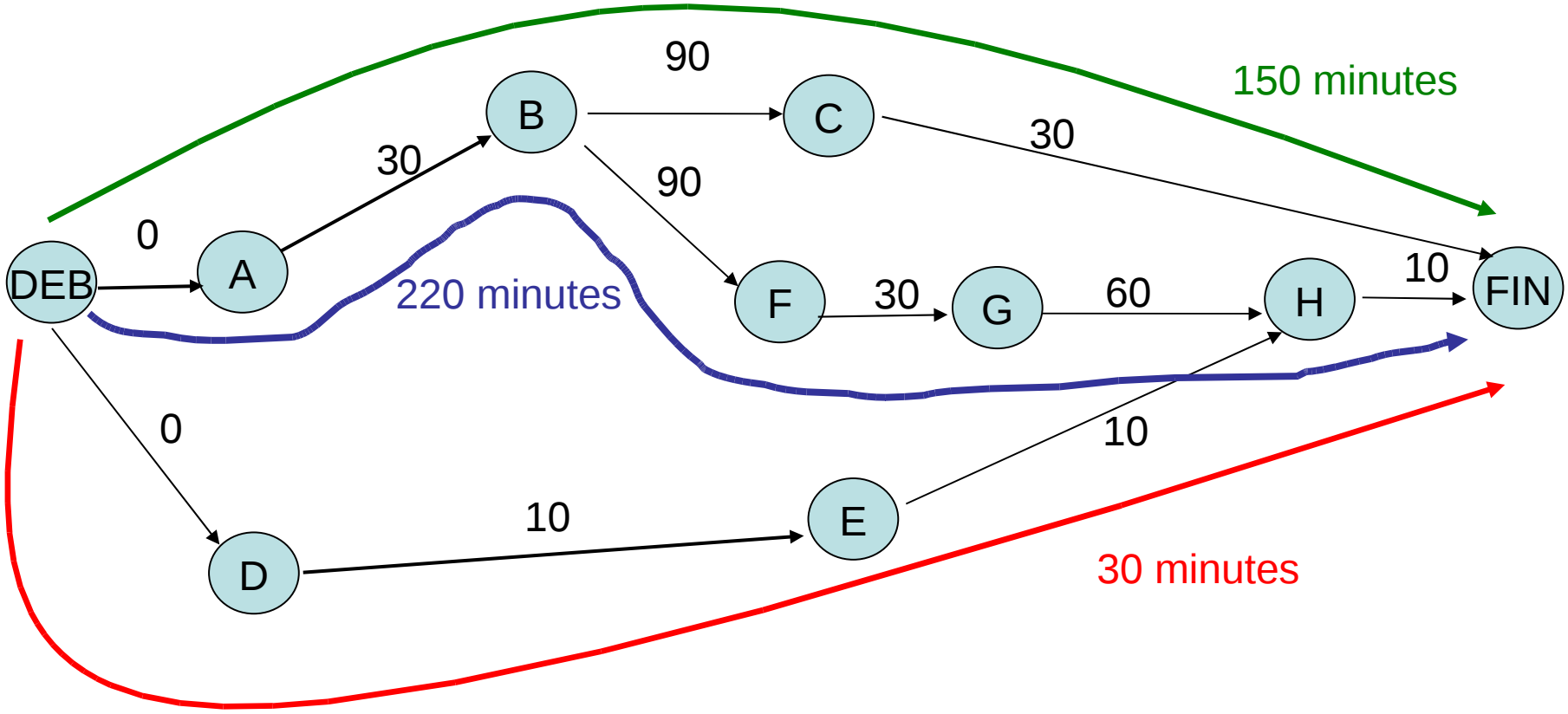
- **Modèle :**

2 modèles possibles

- **Graphe Potentiel-Tâche :**
  - 1 sommet = 1 tâche
  - 1 arc = relation d'antériorité entre 2 tâches
- **Graphe Pert :**
  - 1 sommet = 1 étape
  - 1 arc = 1 tâche

Ordonnancement

Graphe Potentiel-Tâche :



Durée du projet =
Longueur du plus long chemin du sommet DEBut jusqu'au sommet FIN

Ordonnancement

- **Algorithmes de résolution :**

  - t(x) : date de début au plus tôt d'une tâche x

  Il faut que les tâches qui précèdent soient terminées.

  t(x) = valeur du chemin le plus long entre les sommets Deb et sommet x

  → algorithme de plus court chemin

  - T(x) : date de début au plus tard d'une tâche x

  C'est la date de début qui ne remet pas en cause la durée du projet

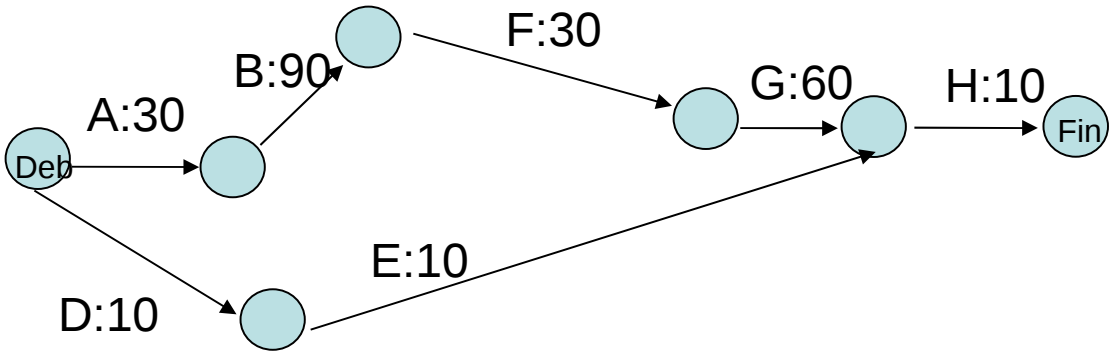
  t(x) = max (t(y) + duree (y), y précède x)

  T(x) = min (T(y) – duree(x), y succède x)

  durée du projet = t(FIN) = T(fin)

Ordonnancement

- **Graphe PERT :**



Ordonnancement

► Algorithmes de résolution :

t(n) : date de début au plus tôt de l'étape n

t(n) = max (t(m) + duree(m,n), m précède n)

t(x) = t(n) si la tâche x est issue de l'étape n

T(n) : date de début au plus tard de l'étape n

T(n) = min (T(m) - duree(n,m), m succède n)

T(x) = T(m) – duree(n,m) si la tâche x va de l'étape n à l'étape m

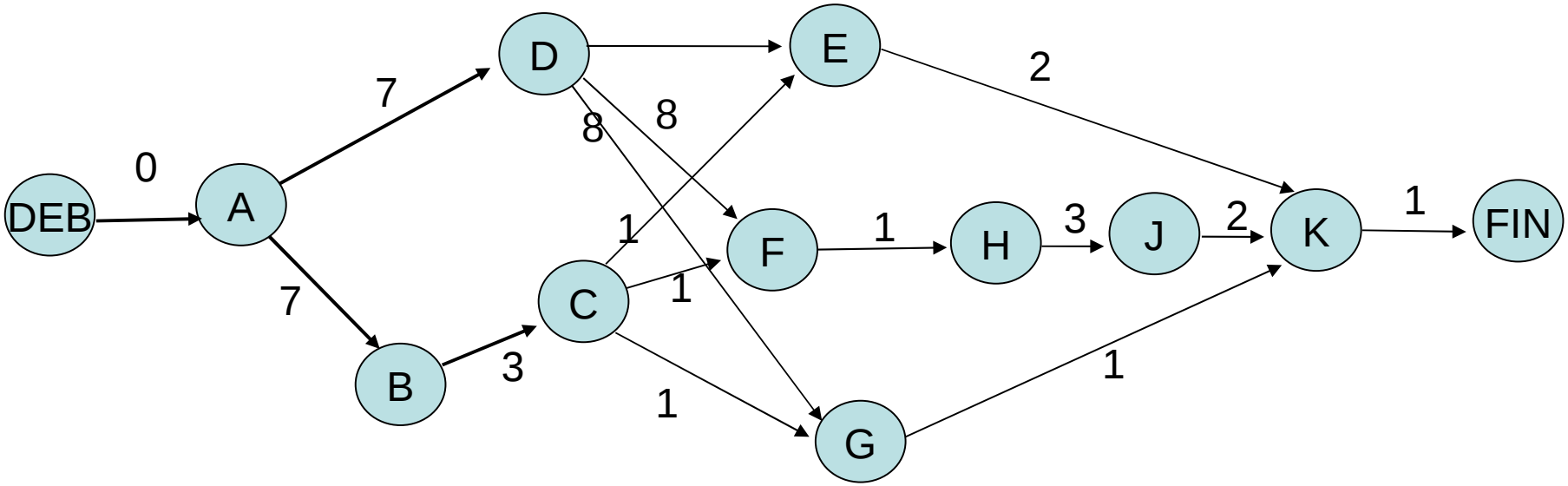
Ordonnancement

► Exemple :

Tâches	Durée semaines		Tâches antérieures
A	7	Travaux de maçonnerie	_
B	3	Charpente de la toiture	A
C	1	Toiture	B
D	8	Installation sanitaire et électrique	A
E	2	Façade	D C
F	1	Fenêtres	D C
G	1	Aménagement du jardin	D C
H	3	Travaux de plafonnage	F
J	2	Mise en peinture	H
K	1	Emménagement	E G J

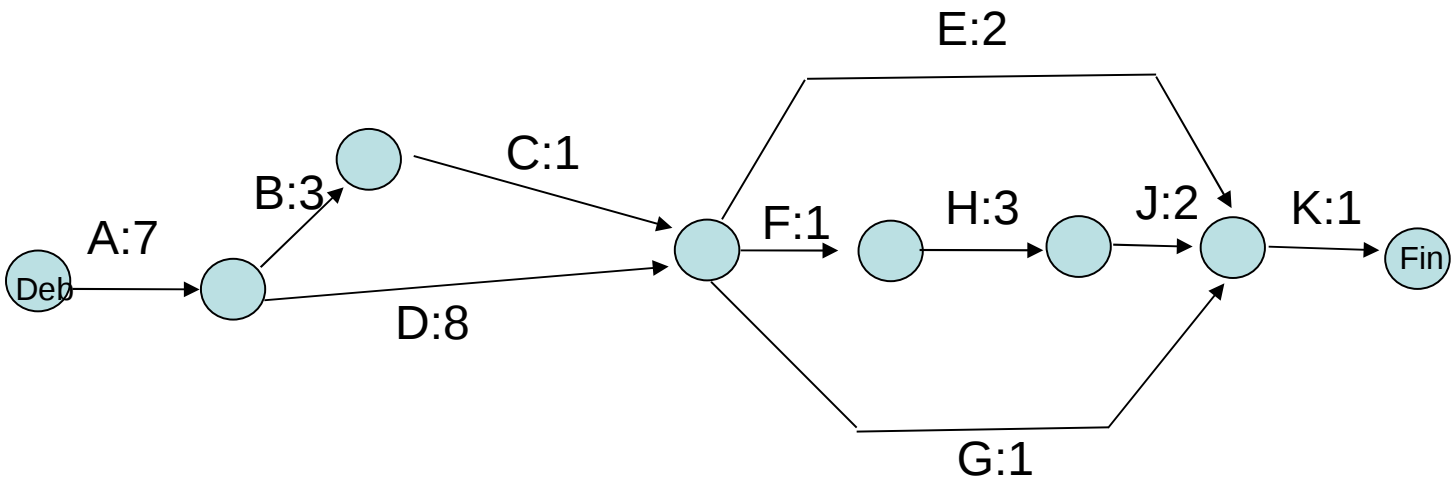
Ordonnancement

- **Exemple :**



Ordonnancement

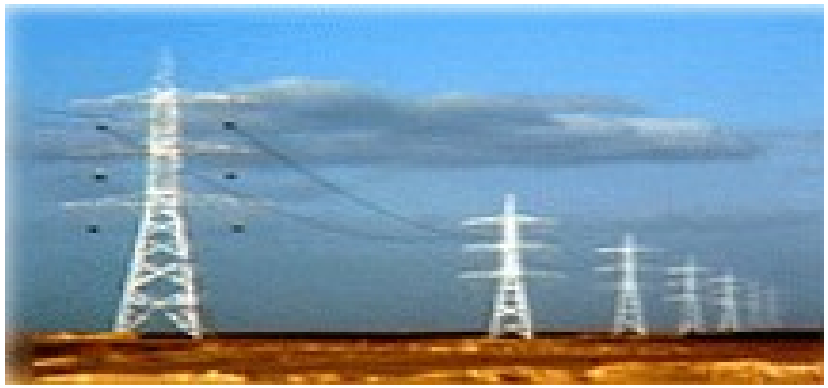
- **Exemple :**



Problème de Flot maximal

► Problème :

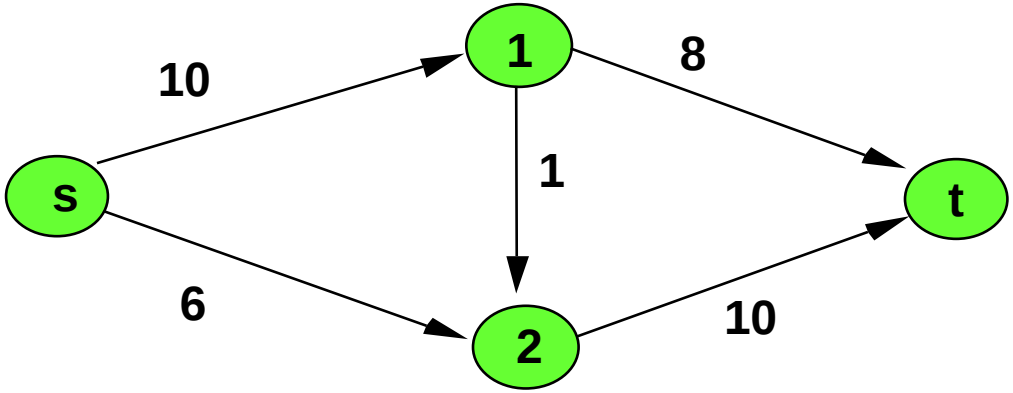
Les flots permettent de modéliser une très large classe de problèmes. Leur interprétation correspond à la circulation de flux physiques sur un réseau : distribution électrique, réseau d'adduction, acheminement de paquets sur Internet, ... Il s'agit d'acheminer la plus grande quantité possible de matière entre une source *s* et une destination *t*. Les liens permettant d'acheminer les flux ont une capacité limitée, et il n'y a ni perte ni création de matière lors de l'acheminement : pour chaque noeud intermédiaire du réseau, le flux entrant (ce qui arrive) doit être égal au flux sortant (ce qui repart).



Problème de Flot maximal

► Modèle :

- ✓ Un réseau est un graphe orienté *N=(V,A)* avec une valuation positive de ses arcs.
- ✓ La valuation *c(x,y)* d'un arc *(x,y)* est appelée la capacité de l'arc.
- ✓ On distingue sur *N* deux sommets particuliers : une source *s* et une destination *t*.
- ✓ Les autres sommets sont les noeuds intermédiaires du réseau.



Problème de Flot maximal

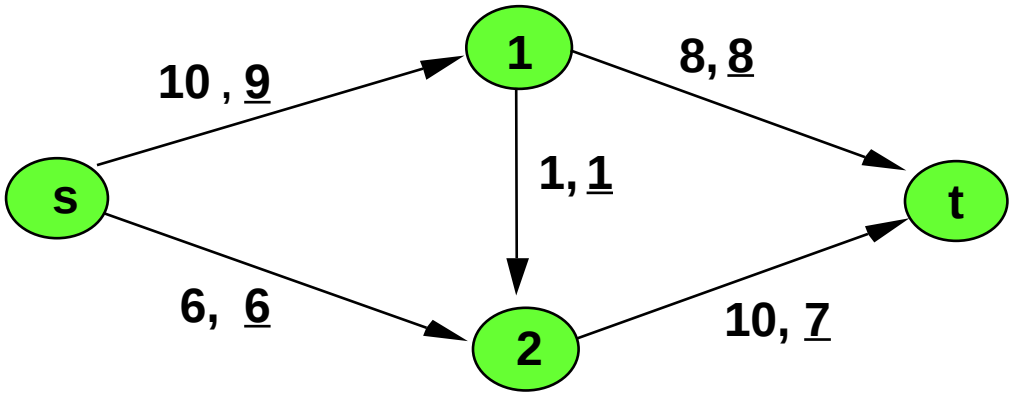
► Modèle :

✓ **Un flot est une valuation positive des arcs du réseau qui vérifie :**

$0 \leq F(x,y) \leq c(x,y)$

Flot entrant sur x = Flot sortant de x

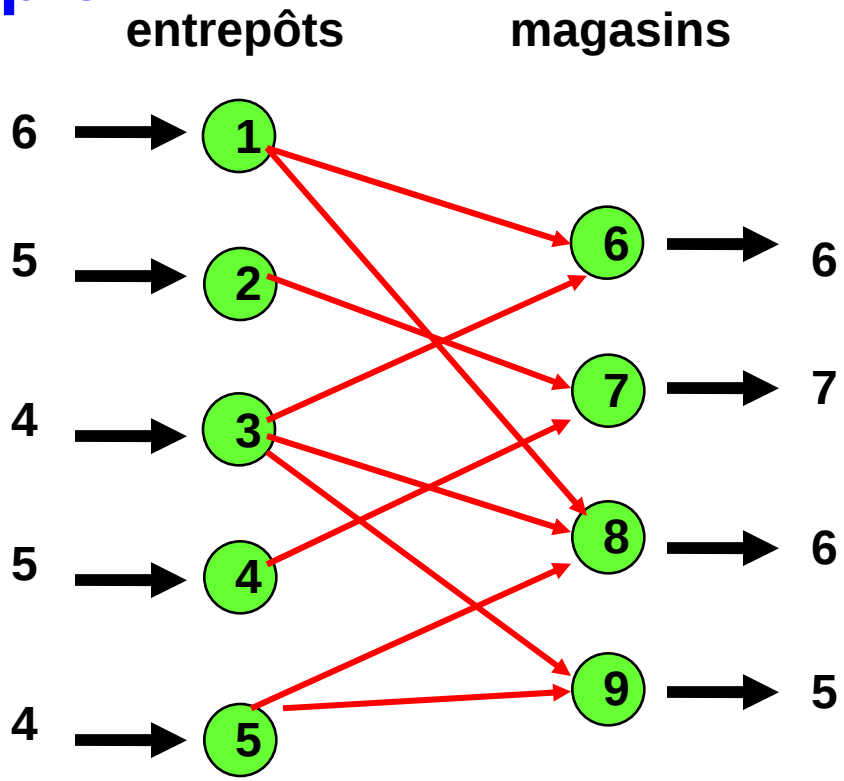
✓ **V= valeur du flot = flot sortant de t**



V=15

Problème de Flot maximal

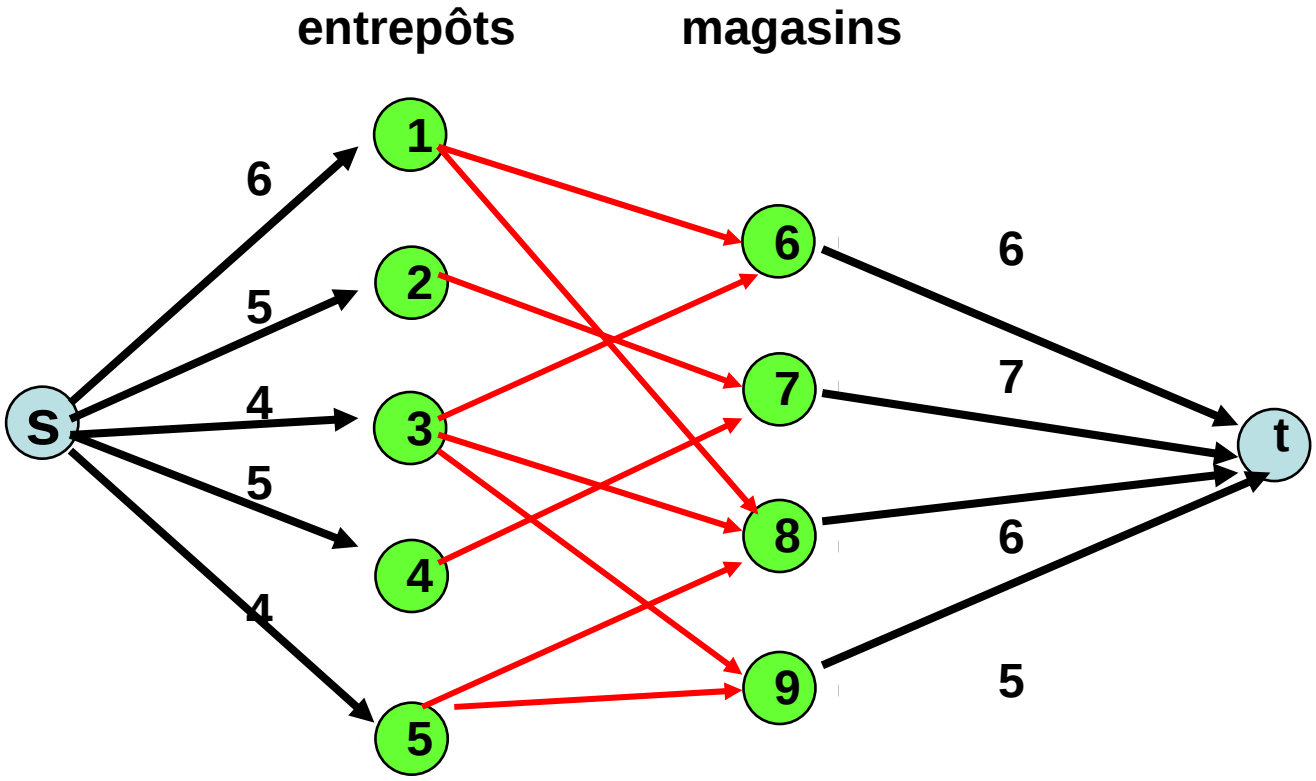
- **Exemple 1:**



Existe t-il un programme de transport des marchandises des entrepôts vers les magasins?

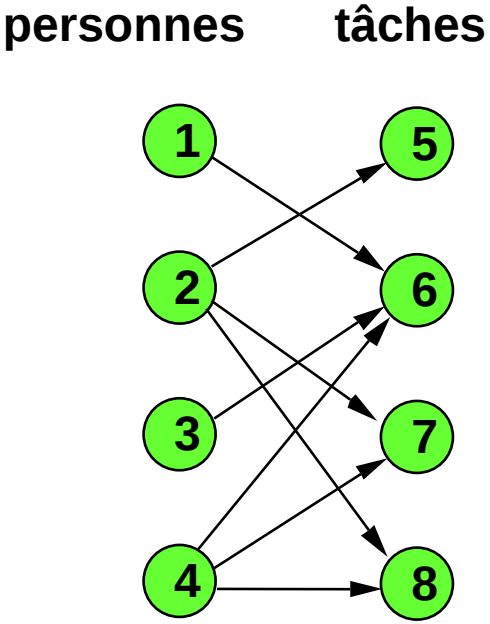
Problème de Flot maximal

- **Exemple 1:**



Problème de Flot maximal

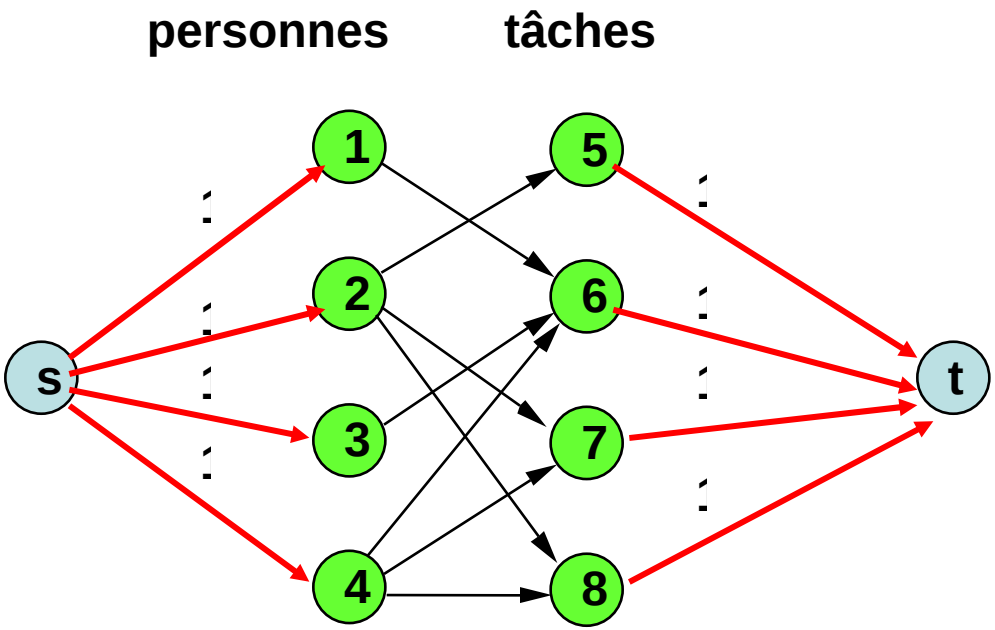
- **Exemple 2 :**



Peut-on affecter les personnes aux tâches de façon à ce que chaque personne soit affectée à une tâche
et chaque tâche réalisée par une personne?

Problème de Flot maximal

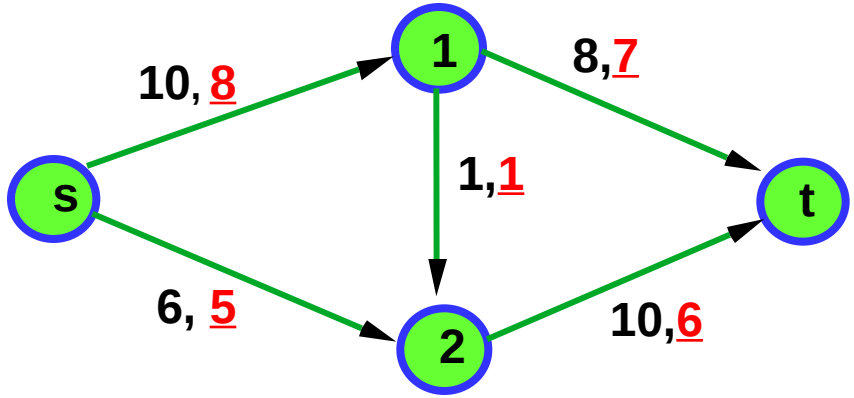
- **Exemple 2 :**

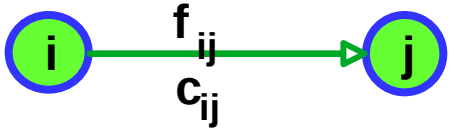


Est ce que le flot max de s à t est de valeur 4?

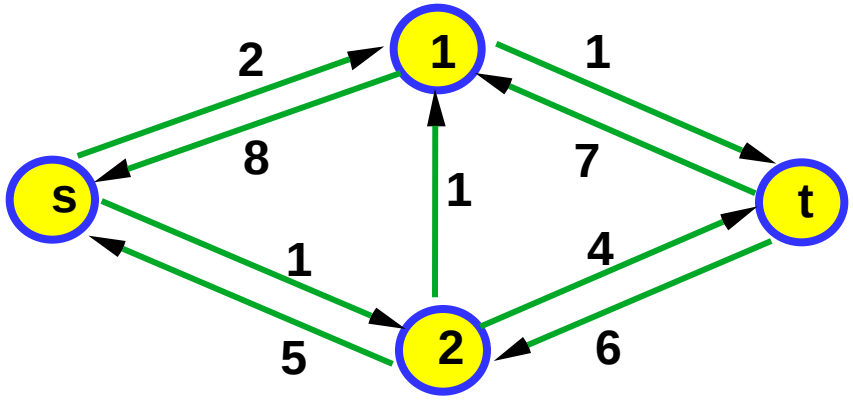
Problème de Flot maximal

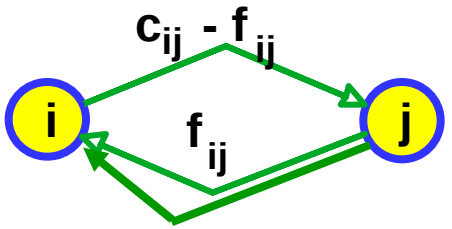
R





Re(x)





Le *réseau résiduel* associé à un flot f

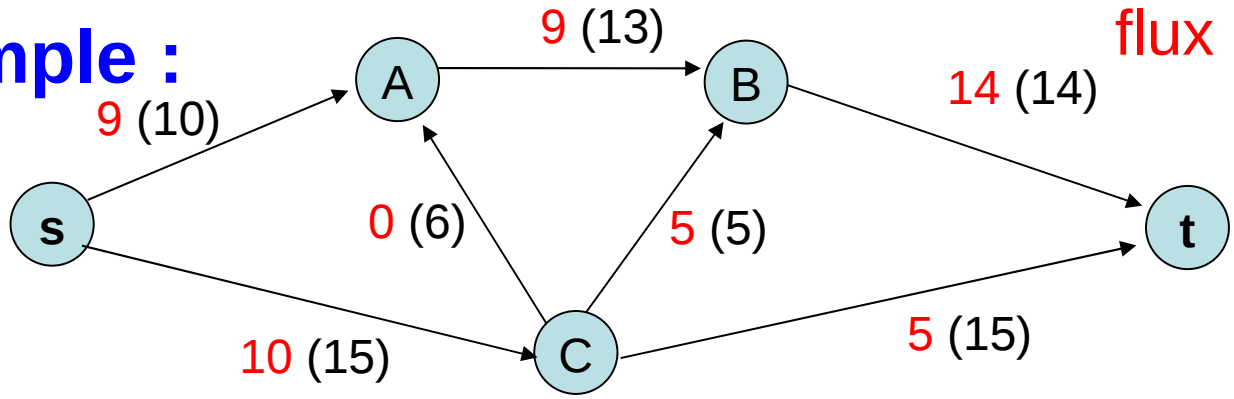
r_{ij} capacité résiduelle de l' arc (i,j)

Problème de Flot maximal

- **Algorithmes de résolution (Ford & Fulkerson) :**
- **Début**
  - **f := 0**;(ou flot complet: flot saturant au moins un arc sur tout chemin de s à p dans R)
  - **creér le réseau résiduel ou graphe d'écart Re(x);**
  - **Tant qu'il existe un chemin de s à t dans le réseau résiduel Re(x) faire**
    - début
    - Soit P un chemin de s à t dans Re(x);
    - $\Delta := \delta(P)$;
    - envoyer Δ unités de flot le long de P;
    - remettre à jour les capacités résiduelles r;
    - fin
- **fin {le flot est maintenant maximal}.**

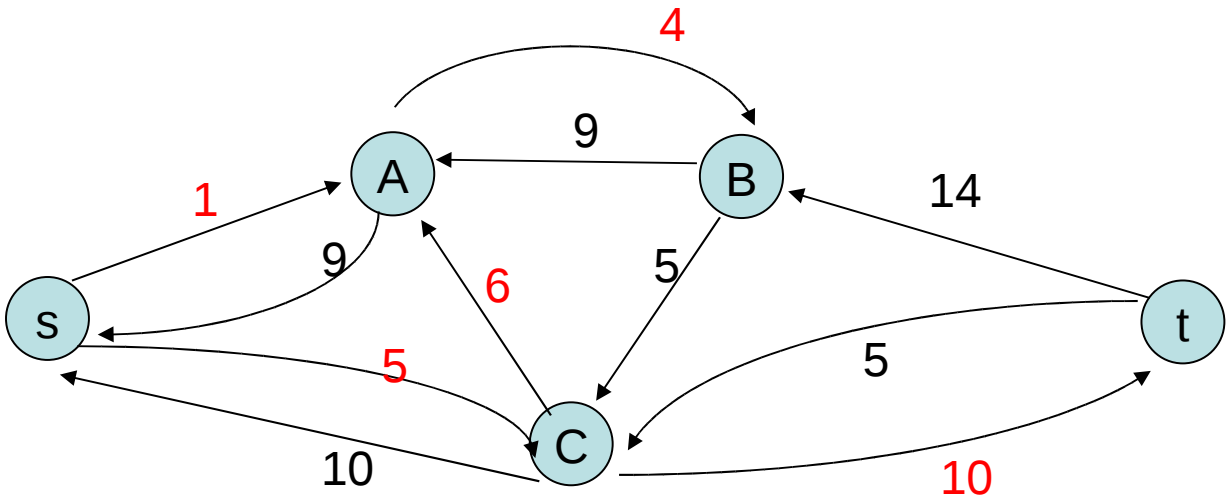
Problème de Flot max

► **Exemple :**



$V(x^0)=19$

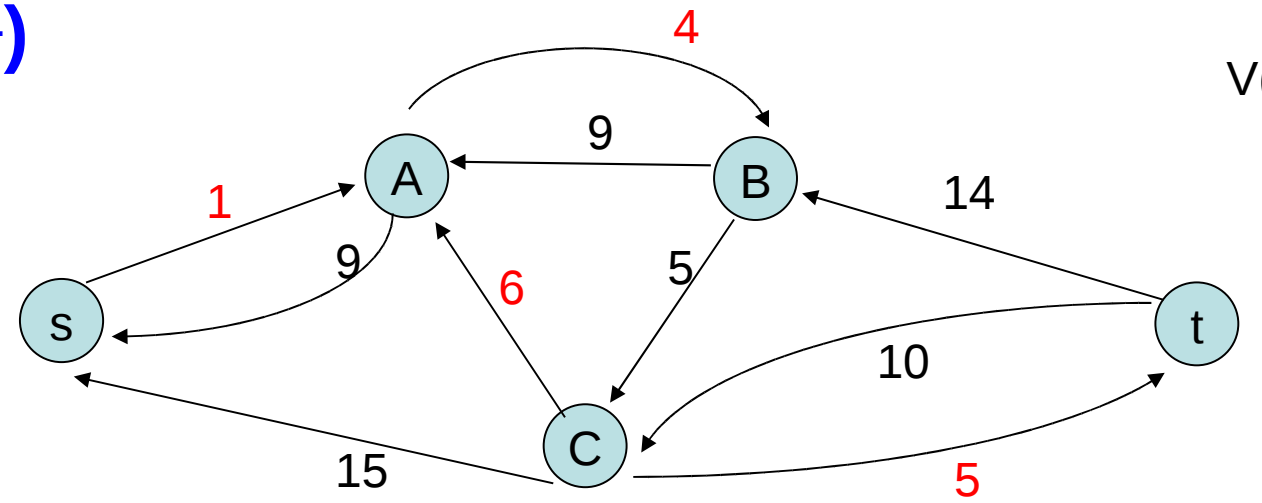
► **Réseau résiduel $Re(x^0)$**



S C t➔5 unités

Problème de Flot max

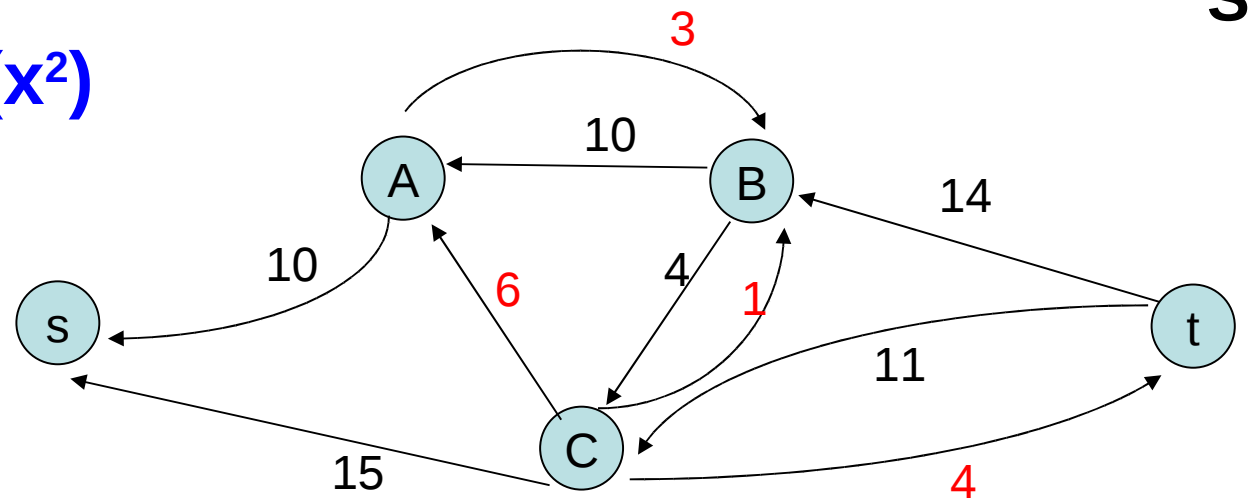
- **Re(x^1)**



$V(x^1)=v(x^0)+5=24$

S A B C t ➔ 1

- **Re(x^2)**

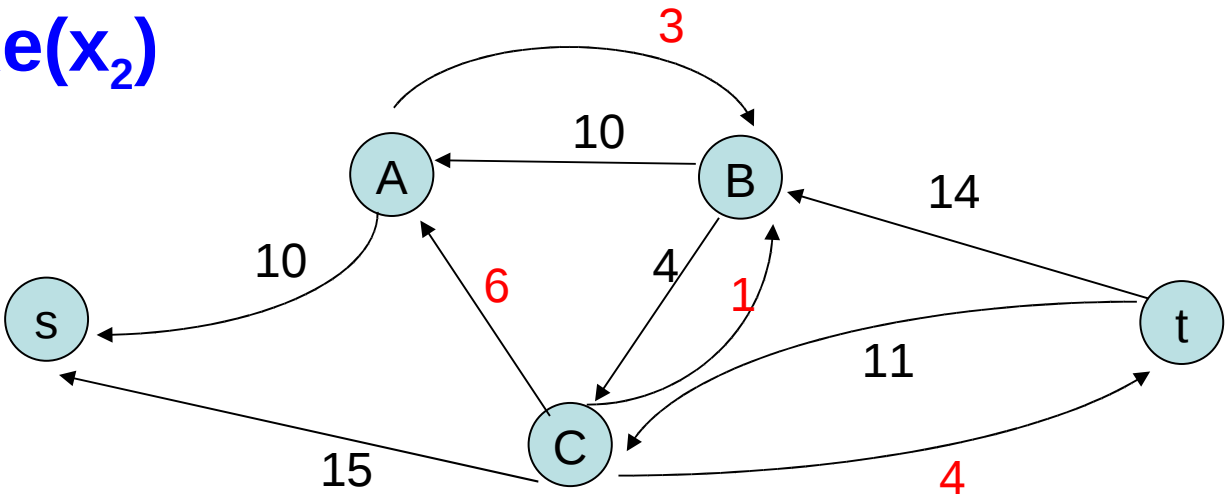


$V(x^2)=v(x^1)+1=25$

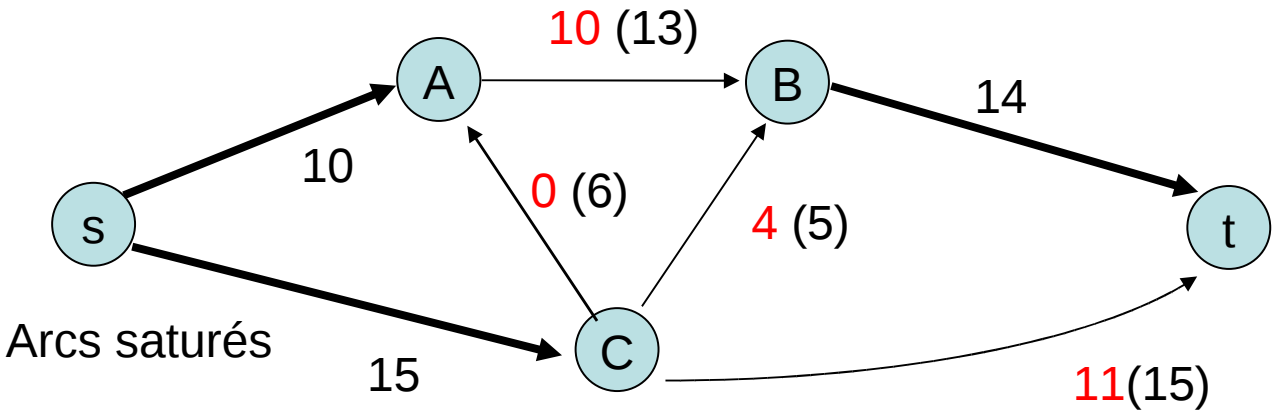
Pas de chemin de s à t, X2 flot optimal

Problème de Flot maximal

- **$Re(x_2)$**



- **On repasse à la fin au graphe initial $R(x_2)$**

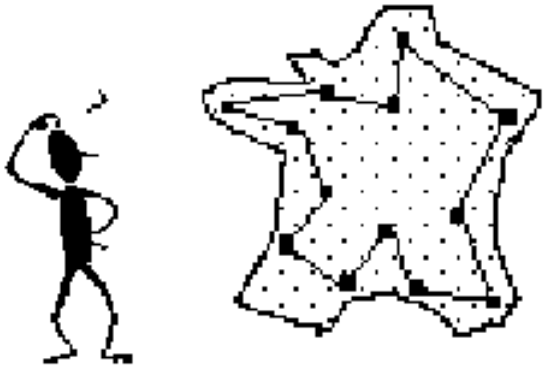


$V(x^2)=25$

Problème du voyageur de commerce

► Problème :

Un voyageur de commerce doit visiter n villes données en passant par chaque ville exactement une fois. Il commence par une ville quelconque et termine en retournant à la ville de départ. Les distances entre les villes sont connues. Quel chemin faut-il choisir afin de minimiser la distance parcourue ? La notion de distance peut-être remplacée par d'autres notions comme le temps qu'il met ou l'argent qu'il dépense : dans tous les cas, on parle de coût.



Problème du voyageur de commerce

Nombre de possibilités de chemins et temps de calcul en fonction du nombre de villes

(on suppose qu'il faut 1 μ s pour évaluer une possibilité)

Nb villes	Nb possibilités	Temps de calcul
5	12	12μ s
10	181440	0,18ms
15	43 milliards	12 heures
20	60E+15	1928 ans
25	310E+21	9,8 milliars d'années

Problème du voyageur de commerce

► Historique :

19ième siècle : Les premières approches mathématiques exposées pour le problème du voyageur de commerce ont été traitées au siècle par les mathématiciens Sir William Rowan Hamilton et Thomas Penyngton Kirkman. Hamilton en a fait un jeu : Hamilton's Icosian game. Les joueurs devaient réaliser une tournée passant par 20 points en utilisant uniquement les connections prédéfinies.

années 1930 : Le PVC est traité plus en profondeur par Karl Menger à Harvard. Il est ensuite développé à Princeton par les mathématiciens Hassler Whitney et Merril Flood. Une attention particulière est portée sur les connections par Menger et Whitney ainsi que sur la croissance du PVC.

1954 : Solution du PVC pour 49 villes par Dantzig, Fulkerson et Johnson par la méthode du ***cutting-plane***[3].

1975 : Solution pour 100 villes par Camerini, Fratta and Maffioli

1987 : Solution pour 532, puis 2392 villes par Padberg et Rinaldi

1998 : Solution pour les 13 509 villes des Etats-Unis.

2001 : Solution pour les 15 112 villes d'Allemagne par Applegate, Bixby, Chvàtal et Cook des universités de Rice et Princeton.

Problème du voyageur de commerce

- **Modèle : un graphe avec des arêtes valuées**

Le problème du voyageur de commerce est de parcourir le graphe en passant au moins une fois en chaque sommet et en minimisant la distance totale, étant donné que chaque arête a une certaine longueur.

Problème du voyageur de commerce

► Algorithmes de résolution :

Les algorithmes pour résoudre le PVC peuvent être répartis en deux classes :

- ✓ **les algorithmes déterministes qui trouvent la solution optimale**
- ✓ **les algorithmes d'approximation qui fournissent une solution presque optimale**

Algorithmes déterministes

Les algorithmes déterministes permettent de trouver la solution optimale, mais leur compléxité est de l'ordre du factoriel. Les algorithmes les plus efficaces sont «cutting-plane»[3] et «facet-finding»[4]. Ces algorithmes complexes ont un code de l'ordre de 10 000 lignes. De plus, ils sont très gourmands en puissance de calcul. Par exemple, il a fallut 27 heures à un puissant super-calculateur pour trouver la solution optimale pour 2392 villes.

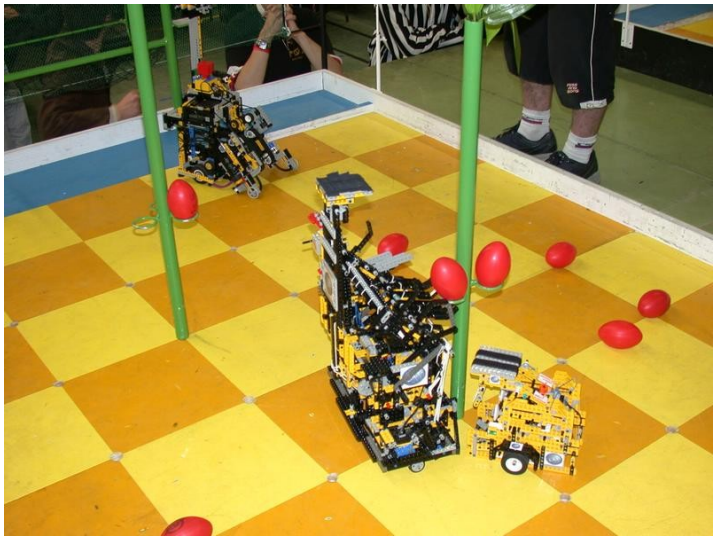
Algorithmes d'approximation (heuristiques)

Les algorithmes d'approximation permettent de trouver une solution dont le coût est proche du coût de la solution optimale. Ils ont l'avantage de permettre en un temps raisonnable de trouver une solution. De ce fait, ils ne sont à utiliser que dans les cas où une solution approchée est acceptable.

Problème du voyageur de commerce

- **PVC et coupe de France de robotique 2004**

  Le jeu est le Coconut Rugby : il s'agit de récupérer des petites balles souples (de la forme d'un petit ballon de rugby) disposées sur le terrain, et de les amener ou lancer dans l'en-but adverse.
  Il y a 8 balles posées à terre, disposées aléatoirement (mais symétriquement).
  Il y a aussi 2 palmiers qui contiennent chacun 3 balles, mais sont aussi des obstacles.
  Les robots sont placés de part et d'autre du terrain et doivent récupérer le maximum de balle avant de les placer dans le but adverse en les lançant ou en les emportant.
  Bien entendu, connaître le chemin le plus court entre toutes les balles peut être un avantage décisif pour vaincre le robot adverse... D'où le lien évident avec le PVC.



Problème de coloration de graphes

► Problème :

- ✓ **exemple 1 : Peut-on colorier une carte avec quatre couleurs de sorte que deux pays ayant une frontière commune (autre que réduite à un point) n'aient jamais la même couleur?**

- ✓ **exemple 2 : Organiser un examen suivant les matières que doit passer chaque étudiant. Comment mettre en parallèle plusieurs épreuves sans léser un candidat ?**

- ✓ **exemple 3 : Optimiser l'utilisation des machines de travail. Comment mettre en parallèle des fabrications utilisant plusieurs machines ?**

- ✓ **exemple 4 : Problème d'incompatibilité. Comment faire cohabiter des personnes ou animaux en tenant compte de leur incompatibilité?**

Problème de coloration

- **Modèle :**

  Le problème consiste à attribuer à chaque sommet d'un graphe une couleur sans que deux sommets reliés par une arête soient de la même couleur. C'est cette notion qui est utilisée dans le *théorème des quatre couleurs* et ses dérivés. Le nombre minimal de couleurs est appelé nombre chromatique qui se note $\gamma(G)$.

Problème de coloration

- **Algorithmes de résolution : Algorithme de Welsh et Powell**

2. Repérer le degré de chaque sommet.
3. Ranger les sommets par ordre de degrés décroissants. (dans certains cas plusieurs possibilités)
4. Attribuer au premier sommet (A) de la liste une couleur.
5. Suivre la liste en attribuant la même couleur au premier sommet (B) qui ne soit pas adjacent à (A).
6. Suivre (si possible) la liste jusqu'au prochain sommet (C) qui ne soit adjacent ni à A ni à B.
7. Continuer jusqu'à ce que la liste soit finie.
8. Prendre une deuxième couleur pour le premier sommet (D) non encore colorié de la liste.
9. Répéter les opérations 4 à 6.
10. Continuer jusqu'à avoir colorié tous les sommets.

Cette méthode n'aboutit pas forcément à une coloration minimale.

Il faut donc observer si on peut faire mieux (c'est-à-dire avec moins de couleurs).

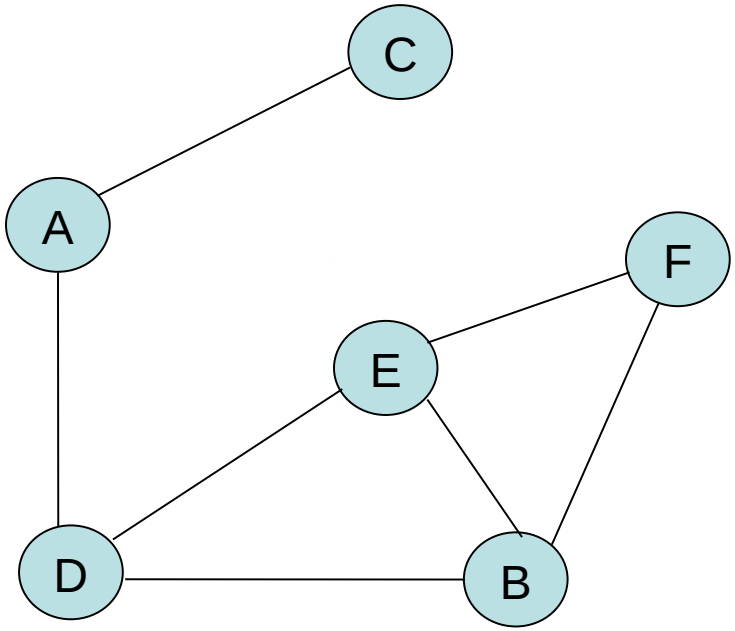
Problème de coloration

- **Exemple : 5 étudiants, 6 examens**

	A	B	C	D	E	F
1	*		*			
2	*			*		
3		*			*	*
4				*	*	
5		*		*		

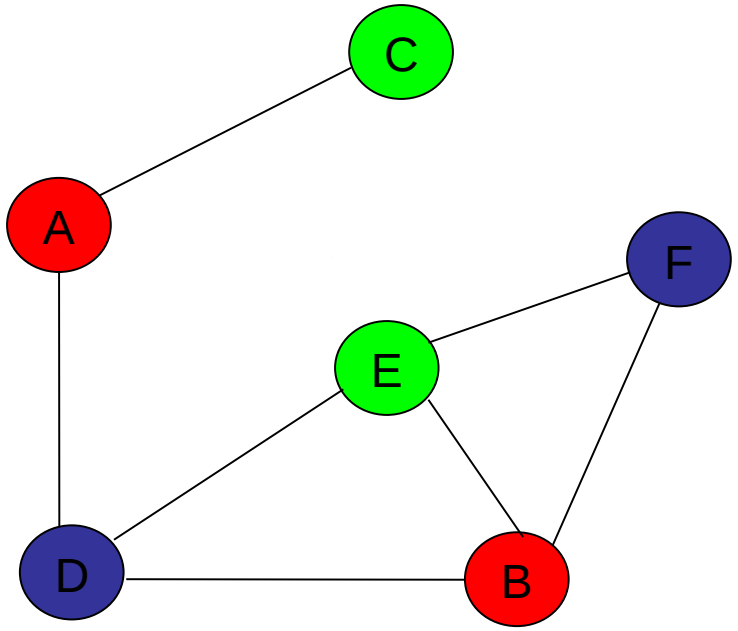
Problème de coloration

- **Exemple : 5 étudiants, 6 examens**



Problème de coloration

- **Exemple : 5 étudiants, 6 examens**



3 couleurs = 3 sessions d'examens

Problème de coloration