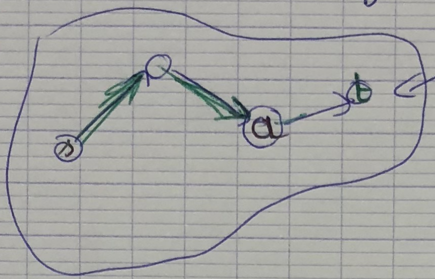
3 algos : calculs de distance de n sources à n dest.

- Bellman Ford :

Approche par prog. dynamique



Notons $D_k[a]$ la distance de s à a qui utilise au plus k arêtes

Si le graphe est $G = (V, E, \omega)$
sommet arêtes poids

convent° $\begin{cases} \omega(x,x) = 0 \\ \omega(x,y) = \infty \\ \text{si } (x,y) \notin E \end{cases}$

Soit s la source des distances :

$D_1[x] = \omega(s,x)$

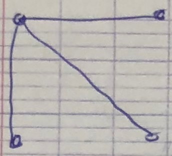
$k \in [\![2, |V|-1]\!]$, $D_k[t] = \min \{ D_{k-1}[x] + \omega(x,t) \mid x \in V \}$

BellmanFord1 $(G = (V, E, \omega), s \in V)$

$\Theta(\lvert V\rvert)$	$\forall x \in V, D_1[x] \leftarrow w(s,x)$
$\Theta(\lvert V\rvert)$	for $k \leftarrow 2$ to $\lvert V\rvert - 1$
$\Theta(\lvert V\rvert^2)$	for $t \in V$
$\Theta(\lvert V\rvert^3)$	$D_k[t] \leftarrow \min_{x \in V} D_{k-1}[x] + w(x,t)$
	return $D_{\lvert V\rvert-1}$
$\Theta(\lvert V\rvert^3)$	

Déroule l'algo sur le graphe suivant

15/03/18
P1.



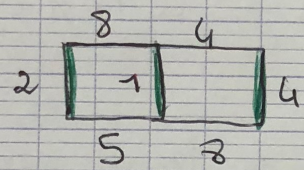
C'est une condit° necessaire mais pas suffisante.

Théorème de Tutte : un graphe $G=(V,E)$ possède un couplage parfait ssi pr chaque ss-E $U \subseteq V$ le graphe induit par $V-U$ (on supp les sommets de U) possède au plus $|U|$ composante

↳ Ne donne pas un algo efficace car il faut énumerer les $2^{|V|}$ ss-E U.

On peut décider l'∃ence d'un couplage parfait simplement en appliquant Edmonds $O(|V||E|)$ puis en vérifiant $|M| = \frac{|V|}{2}$.

COUPLAGE PONDÉRÉ

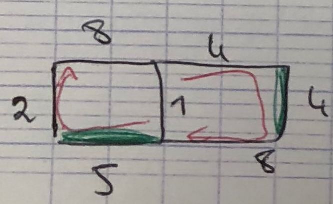


M couplage de poids 7

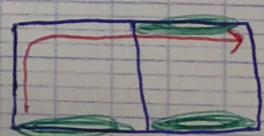
On veut le couplage de poids maximal.

Chemin améliorant pondéré.

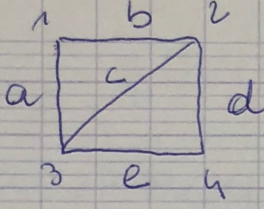
- Chemin qui alterne arête de M (jaune) et de E\M (blanches)
- Σ des poids des arêtes du chemin sur M doit être plus petite que Σ des arêtes du chemin hors M.
- Si le chemin termine ou commence par une arête de E\M alors l'une des extrémités de cette arête doit être libre.



poids 9



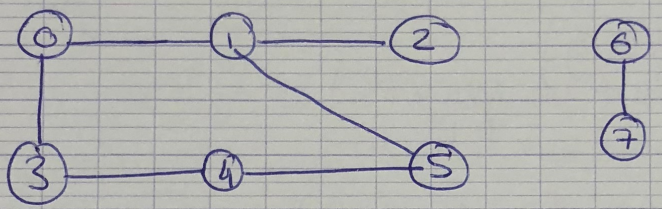
poids 14



$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ \left[\begin{array}{cccc} a+b & -b & -a & 0 \\ -b & b+c+d & -c & -d \\ -a & -c & a+c+e & -e \\ 0 & -a & -e & d+e \end{array}\right] \end{array}$$

$$\begin{vmatrix} a+b & -a & 0 \\ -a & a+c+e & -e \\ 0 & -e & d+e \end{vmatrix} = (a+b)(a+c+e)(d+c) - (a+b)e^2 - (d+e)a^2$$

$$= acd + ace + aed + bad + bae + bcd + bce + bed.$$



$G = (V, E)$ — V : sommet, E : arête

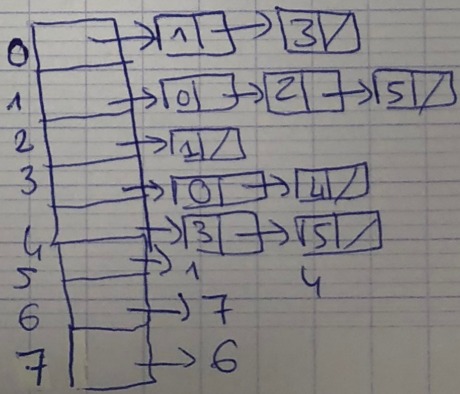
$V = \{0,1,2,3,4,5,6,7\}$

$E = \{(0,1), (0,3), (1,2), (1,5), (3,4), (4,5), (6,7)\}$

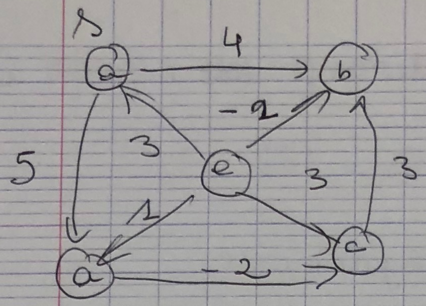
On convient que si (x,y) est une arête alrs (y,x) aussi

$(5,4) \in E$

liste d'adjacence



$\infty \rightarrow$ lorsqu'il $\nexists$ pas d'arc.



nombre d'arc

	a	b	c	d	e
D_1	0	4	∞	3	∞
D_2	0	4	2	5	3
D_3	0	4	2	3	3
D_4	0	4	2	3	1.

$t = a$ min $\left\{ \begin{array}{l} D_1[a] \\ b \\ c \\ d \\ e \end{array} \right.$ $\begin{array}{l} a \rightarrow a \quad 0+0 \\ b \rightarrow a \quad 4+ \\ c \rightarrow a \quad \infty + \\ d \rightarrow a \quad 5+ \\ e \rightarrow a \quad \infty + \end{array}$

0

1 _ Optimisat° 1 : ne pas essayer de passer par un x qui n'a pas un arc vers t

$\Theta(|V|)$ | BellmanFord2 $(G = (V, E, \omega), s \in V)$
$\Theta(|V|)$ | $\forall x \in V, D_1[x] \leftarrow \omega(s, x)$
$\Theta(|V|^2)$ | for $k \leftarrow 2$ to $|V| - 1$
$\Theta(|V|.|E|)$ | $D_k \leftarrow D_{k-1}$
$\Theta(|V|.|E|)$ | for $(x, t) \in E$
| $D_k[t] \leftarrow \min(D_k[t], D_{k-1}[x] + \omega(x, t))$
| return $D_{|V|-1}$

$\Theta(|V|^2 + |V|.|E|)$
$= \Theta(|V|.|E|)$
si le graphe est connexe.
(raisonnable ds le calcul de distance)

2 _ Optimisat° 2 : on remplace D_1, D_{k-1} et D_k par D.

$\Theta(|V|.|E|)$ $\left\{ \begin{array}{l} \forall x \in V, D[x] \leftarrow \omega(s, x) \\ \text{for } k \leftarrow 2 \text{ to } |V| - 1 \\ \quad \text{for } (x, t) \in E \\ \quad\quad D[t] \leftarrow \min D[t], D[x] + \omega(x, t) \\ \text{return } D \end{array} \right.$

8/03/18 Notes → maximum ⇒ maximal

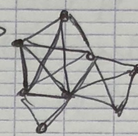
$\mathcal{M}$ un couplage M, en $|M| \leq \left\lfloor \frac{M}{2} \right\rfloor$

Def : un couplage est parfait si les extrémités des arêtes de M couvrent V

$\Rightarrow |M| = \frac{|V|}{2}$

⇒ pas de couplage parfait si $|V|$ impaire

- Construire un couplage maximal :



- trouver un chemin ameliorant P
- calculer $M' \leftarrow M \oplus P$ (arêtes de M ou P mais pas ds les 2) (xor)
- on obtient $|M'| = |M| + 1$

repeter jusq à abs d'un chemin ameliorant.

$M \leftarrow \emptyset$
tant que le graphe possède des arêtes
↳ prendre une arête au hasard
↳ l'ajouter à M.

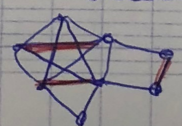
Def : Un sommet $v \in V$ est libre s'il n'est l'extrémité d'aucune arête de M.
Un chemin ameliorant est un chemin reliant 2 sommets libres en alternant arêtes de $E \setminus M$ et arêtes de M.

Th : Soit $G = (V, E)$ et M un couplage, il $\exists$ un chemin ameliorant ssi M n'est pas maximum.

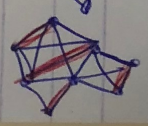
($\Rightarrow$) Soit P un chemin ameliorant alors $M' = M \oplus P$ possède 1 arête de plus que M → M pas maximum

($\Leftarrow$) on suppose M non maximum soit M' un couplage maximum ($|M'| > |M|$)

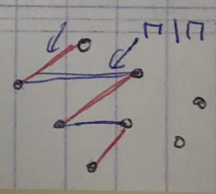
Considerons le graphe $G' = (V, M \oplus M')$



M non maximal

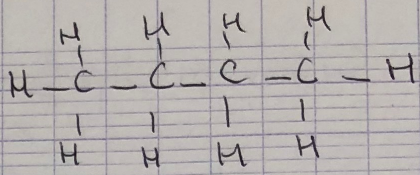


M' maximal

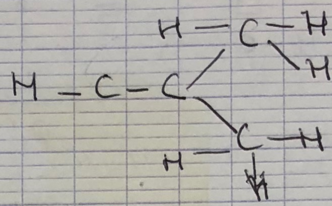


$M | M'$

G'

$n=4$ butane 

2



n	iso
5	3
6	5
7	9
8	18
	35
	75
	159
	355

C_nH_{2n+2}

un alcane = 1 arbre avec n carbone et x hydrogènes

on veut mq $x = 2n+2$.

$n+x$ sommets
$n+x-1$ arêtes → prop : un arbre de n sommets possède n-1 arêtes

Propriété : Σ des degrés de $\forall$ les sommets = 2x le nbre d'arêtes.

$$\sum_{v \in V} deg(v) = 2 \times |E|$$

$$\Sigma\, deg(v) = 2 \times (n+x-1)$$

$$4n + x = 2n + 2x - 2$$

$$2n+2 = x$$

Propriété : le nbre de sommet de degré impair est pair

preuve : $\Sigma\, deg(v) = 2 \times |E|$ est pair

La formule de Cayley :
le nbre d'arbre (= graphe acycliq et connexe)

8/02/18 sans racine de sommets numerotés est n^{n-2}
P2

Démo (moderne)

On montre une bijectᵒ entre 2 ensembles : {les fctᵒ de $[\![1,n]\!]$ vers $[\![1,n]\!]$} et {les arbres numerotés entre 1 et n, avec 2 marqueurs : "début", "fin"}.

card (du 1er ensemble) $= n^n$

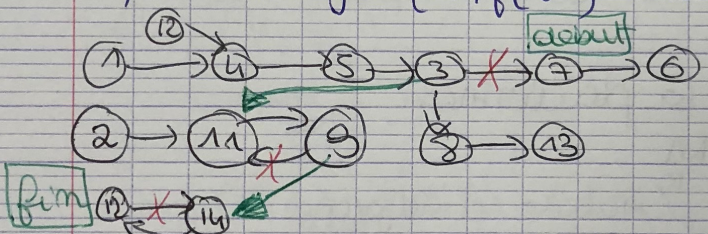
——— 2e ——— $) = T(n) \times n^2$ où $T(n) = n^{n-2}$

nbre infini

x sommets impairs et pairs

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$f(x)$	4	11	7	5	3	4	6	3	11	4	9	14	8	12

1) On intègre $(x, f(x))$ cᵐᵉ un arc :



2) Casser les cycles après leur plus petite valeur

3) Connecter les cycles ds l'ordre des plus petites valeurs ⟶

4) Marquer le debut du plus petit cycle et la fin du plus grd.

5) Renver l'orientatᵒ du graphe (→ devient ←)

edges = [[1,3], [0,2,5], ...]

Depth-First Search (DFS) parcours en prof.

0
1
2
5
4
3
6
7

```
         list
def dfs(adj):
    n = len(adj)
    seen = [False] * n      (liste de n False)
    def rec(start):
        print(start)
        for y in adj[start]:
            if not seen[y]
                seen[y] = True
                rec(y)
    for start in range(n):
        if not seen[start]:
            seen[start] = True
            rec(start)
```

dfs(edges)
0
1
2
5
4
3
6
7

Vers° itérative avec une pile de paire
(i,j) i smt courant, j = je succes. à traiter

```
def dfs_iter(adj):
    n = len(adj)
    seen = [False] * n
    for start in range(n)
        if seen[start]:
            continue
        stack = [(start, 0)]
        ~~seen[start] = True~~
        while stack:
            src, pos = stack.pop()
            if pos == 0:
                print(src)
                seen[src] = True
```

a) G' possède $\oplus$ d'arêtes de M' que de M.
b) Chq sommet de G' touche au plus une arête de M et ______ de M'.
Dans les composants de G' qui ne sont pas des sommets isolés $\exists$ nécessairement (à cause de a) un composant avec plus d'arêtes de M'.
Ce composant est un chemin améliorant.

$M \leftarrow \emptyset$
tant qu' $\exists$ un chemin améliorant P max $\lfloor \frac{V}{2} \rfloor$ itérat°
  $\underbrace{}_{\text{algo à écrire}}$
L $M \leftarrow M \oplus P$
return M

$O(|V| \times |E|)$

Algo d'Edmonds pr trouver un chemin améliorant
Entrée $G(V,E)$ $M \subseteq E$ couplage
Sortie $P \in E^{*}$ chemin améliorant, ε si pas chemin améliorant
- retirer les étiquettes "[r, c, p]" des sommets
- marquer $\forall$ les arêtes cme non visitées
- répéter aux choix.
A_ trouver un sommet libre $v \in V$, lui donner l'étiquette $[v, B, v]$
B_ Trouver une arête non visitée $(v, w) \in E$ tq v est étiqueté par $[r, B, p]$
1) Marquer (v, w) cme visité
2) Si w est étiqueté et libre alors on a un chemin améliorant
$P \leftarrow$ chemin de r à w
break
3) Si w est non étiqueté et $\exists x$ tq $(w, x) \in M$

étiqueter w par $[r, j, v]$ et x par $[r, B, w]$

4) Si w a pour étiquette $[s, B, q]$ avec $s \neq r$
   $P \leftarrow$ chemin de r à v + (v, w) + le chemin de w à D
   break

5) Si w a pour etiquette $[S, B, q]$ avec $s = r$,
   On a detecté un cycle de taille impartante.
   On retire $\forall$ sommet de ce cycle et on remplace par un new sommet x_i (le bourgeon) dans le graphe. On étiquette x par $[r, B, P']$ avec P' le parent de la racine du cycle.

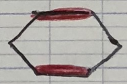
6) Si w a pour étiquette $[S, J, q]$ ne rien faire.
   Si ni Ⓐ ni Ⓑ n'est possible return ε, (chemin vide)
   sortie de boucle par break

P est un chemin amelirant qui peut contenir du bourgeon. Deplier les bourgeons en ajoutant les chemins de taille paire approrié ds le chemin améliorant return P.

Rappel.

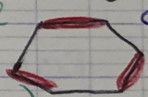
couplage

couplage maximal



couplage parfait (touche $\forall$ les sommets).

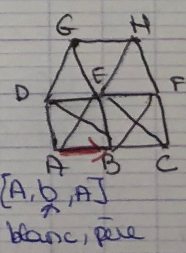
couplage maximum



algo d'Edmond $O(|V| \cdot |E|)$.

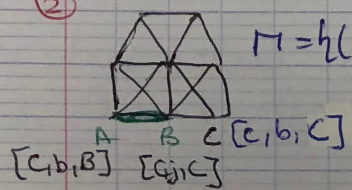
Exp. d'Applicat° de l'algo d'Edmond :
Sommet initiale : A.

①



$M = \emptyset$

1ere itérat° ②



$M = \{(A, B)\}$.

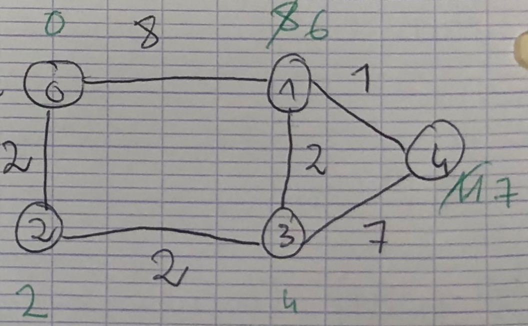
* Sur un graphe connexe $|E| \geq |V| - 1$.

$|E| = \Omega(|V|)$ sur un graphe connexe.

* $\sum_{v \in V} \deg(v) = 2|E| = \Theta(|E|)$

	`def distmap(adj, start)`
$\Theta(1)$	`    n = len(adj)`
$\Theta(\lvert V\rvert)$	`    dist = [None] * n`
$\Theta(1)$	`    q = deque[start]`
$\Theta(1)$	`    dist[start] = 0`
$\Theta(\lvert V\rvert)$	`    while q:`
$\Theta(\lvert V\rvert)$	`        src = q.popleft()`
$\Theta(\lvert V\rvert)$	`        d = dist[src]`
$O(\lvert E\rvert)$	`        for dst in adj[src]`
$O(\lvert E\rvert)$	`            if dist[dst] is None`
$O(\lvert E\rvert)$	`                dist[dst] = d + 1`
" "	`                q.append(dst`
" "	`    return dist`
$\Theta(\lvert V\rvert) + O(\lvert E\rvert)$	

$w(s, d)$ poids de l'arête (s, d)



cf algo dijkstra photo

Algo Bellman Ford calcul de distance (1 src, n dst) ds graphe avec poids $\in \mathbb{R}$ sans cycle de somme < 0 $\Rightarrow$ les PCC (plus court chemin) sont élémentaires sans cycles

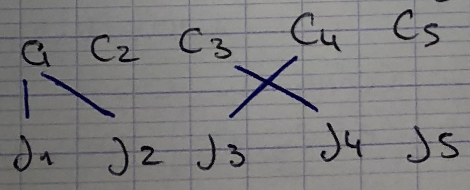
```
FloydWarshall( G =(V, E, w))
  D ← w
  for k ∈ V
    for s ∈ V
      for t ∈ V
        D'[st] ← min D[s,t], D[s,k] + D[k,t]
  D → D'
  return D
```

$\Theta |V|^3$

```
FloydW 2 ( G = (V, E, w))
  D ← w
  ∀ (s, t) ∈ V², Father [s, t] ← s
  for k ∈ V
    for s ∈ V
      for t ∈ V
        m ← D[s, k] + D[k, t]
        if m ≥ D[s, t]
          D'[s, t] ← D[s
        else
          D'[s, t] ← m
          Father [s, t] ← Father [k, t]
  return D, Father.
```

	FW	BF	Diskjhra
precalcul	$\Theta \vert V\vert^3$	—	
requete	$O(\vert V\vert)$	$O(\vert V\vert . \vert E\vert)$	$O(\vert E\vert . \log \vert V\vert)$

Couplage (matching)



candidat

job

Dans l'autre sens
-5 : Remettre les flèches de façon à aller vers vers $\boxed{\text{fin}}$.

-3 : Chercher la + petite valeur entre $\boxed{\text{debut}}$ et -2 et $\boxed{\text{fin}}$ c'est la fin du 1er cycle.
Repeter pr les cycles suivants

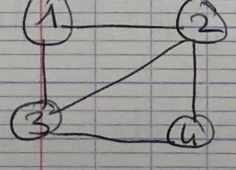
-4 : supprimer $\boxed{\text{debut}}$ et $\boxed{\text{fin}}$

-1 trivial

Theorème : le nbre d'arbres couvrants d'un graphe G est égal à la valeur absolue de n'importe quel cofacteur de la matrice Laplacienne de G

La matrice laplacienne :

$$\underbrace{L = D}_{\text{matrice de degré}} - \underbrace{M}_{\text{matrice d'adjacence}}$$



$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \qquad M = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

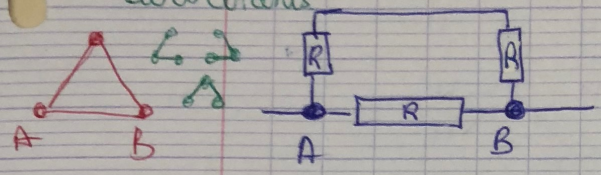
$$L = \begin{bmatrix} 2 & -1 & -1 & 0 \\ \cancel{-1} & \cancel{3} & \cancel{-1} & \cancel{-1} \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$$

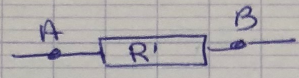
[⊔] ⊓
Z N ↙ ↗

$$\begin{vmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{vmatrix} = 12 - 2 - 2 = 8.$$

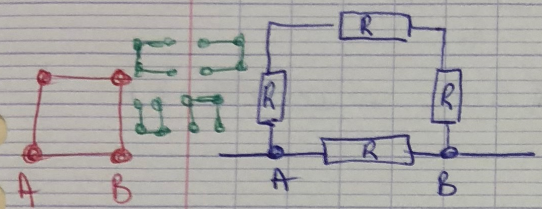
1847 : Gustav Kirchhoff.

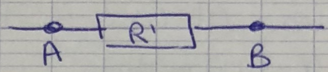
arbres couvrants



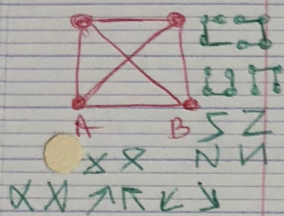
= 

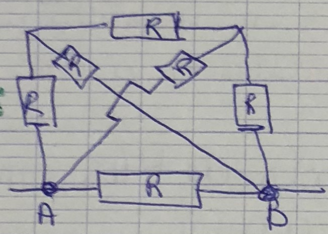
$$\Rightarrow \frac{1}{R'} = \frac{1}{R} + \frac{1}{2R} \quad \text{avec } R' = \frac{2}{3}R.$$



= 

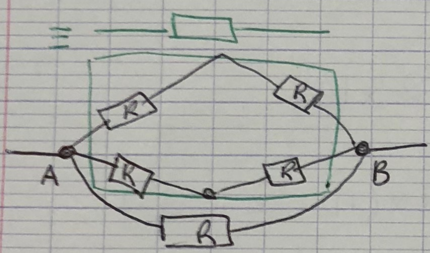
$$\Rightarrow \frac{1}{R'} = \frac{1}{R} + \frac{1}{3R} \quad \text{avec } R' = \frac{3}{4}R$$





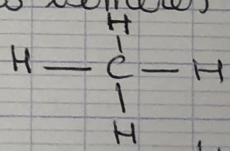
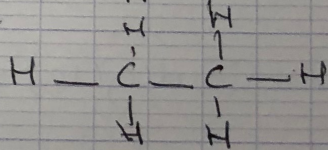
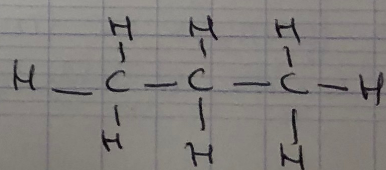
$$R' = \frac{1}{2}R$$

Condit° : résistance de m̂ valeur.



1860 : Cayley.

énumérat° des isomères des alcanes :

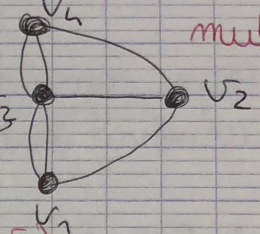
			isomères
$n = 1$	méthane		1
$n = 2$	éthane		1
$n = 3$	propane		1

8/02/18
P_1

- Théorie des graphes :

1735 : Euler
7 ponts de Konigsberg

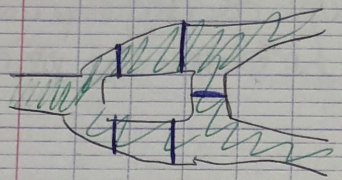
Graphe : E de sommets avec un E d'arrêtes
$G = (V, E)$
$V = \{v_1, v_2, v_3, v_4\}$
$E = ((v_1, v_2), (v_2, v_3) \dots)$



multigraphe.

le parcours : $\sigma = (v_1, v_2)(v_2, v_4)(v_4, v_3) \dots$

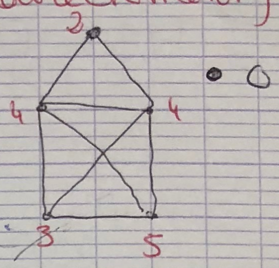
Problème à resoudre



est-il possible de faire une promenade qui traverse chq pont exactement une fois ?

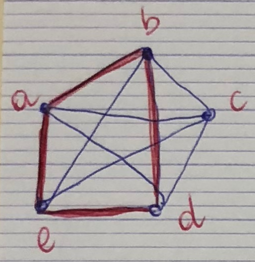
à chaq fois qu'on passe par un sommet on utilise un autre pr rentrer et on arrête pr sortir sauf pour les sommets initiaux et finaux.
⇒ si un chemin eulérien, existe il faut recursivement que le nbre de sommets de degré impair soit 0 (cas où l'arrivée = départ)
ou 2 (————— ≠ départ)
⇒ il $\nexists$ de solut° aux 7 ponts de K.

Theoreme : un (multi)graphe non orienté possède un cycle eulerien ssi $\forall$ ces sommets sont de degré pair et les sommets de degré > 0 sont ds la m̂ composante connexe (ensemble de sommets qui sont ∀ connectés indirectement)

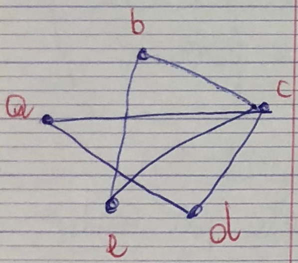


Theoreme 2 : si il y a 2 sommets de degré impair, alors ∃ un chemin eulérien.

Algo pour produire un cycle eulérien ds un graphe connexe dont ∀ les sommets sont pairs.



1. Choisir un sommet (départ)
Construire un chemin à partir de ce sommet en effaçant les arêtes au fur et à mesure.
Comme les sommets sont de degré pair on peut tjrs avancer jusqu'à retomber au pt de départ
a → b → d → e

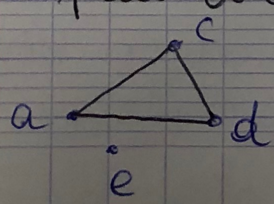


2. ds les arêtes qui reste un des sommets est forcement sur le chemin crée, car le graphe est convexe.

3. Si on répète l'étape 1, à partir de ce sommet
b → c → e → b

4. Inserer ce cycle ds le chemin
a → b → c → e → b → d → e → a.

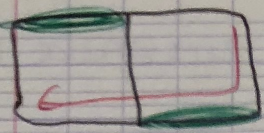
5. Repeter 2-3-4 jusqu'à ce qu'il n'y ait plus d'aretes.



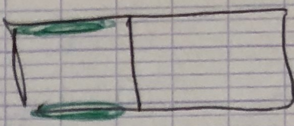
a → c → d → a

a → c → d → a → b → c

e ← d ← b ← e



poids 16.



poids 17

Algo Hongrois / L'algo d'Edmonds & Jonhsons trouve un couplage pondéré maximum (ou min) en $O(|V|^3)$

- une couverture des sommets par les arêtes est un ss-E de E qui touche ∀ les sommets.

par ex
- 1 arbre couvrant
- 1 couplage parfait
- un graphe connexe est sa propre couverture

* une couverture minimale → on ne peut pas retirer d'arêtes
* _______ minimum → il n'∄ d'arête plus petite.

Applicat° : vendeur de jouets de forme geometrique de ≠ couleurs. ∀ les combi formes couleurs n'∄. On cherche le nbre d'objets min representant chaque forme et chaque couleur.

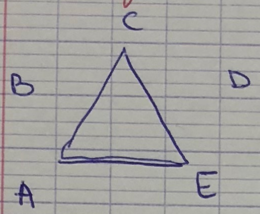
Algo pr calculer une couverture minimum
- On calcule un couplage maximum.
- s'il reste des sommets libres on ajoute des arêtes sortantes pr chaque sommet au couplage

⇒ ça devient une couverture minimum.

PROBLEME DU POSTIER CHINOIS

Entrée : graphe connexe
Sortie : le plus court chemin qui
- visite chaque arêtes au ⊖ 1 fois
- il revient à son point de départ

— Cycle eulérien ∃ ds un graphe connexe si ∀ les degrés sont pairs.



$\overbrace{A\,B\,D\,A}$
BCDEB

$\overbrace{A\,B\,C\,D\,E\,B\,D\,A}$
CEAC

$\boxed{ABCEACDEBDA}$

~~2k~~ sommets de degrés impair

Algo donnant une solut° au problème du postier chinois.

car graphe connexe ↳ $O(|E|)$ — Soit S l'E des sommets de degrés impairs.

$O(|V|^3)$ - calcul le graphe G_S (étiqueté par les distance de sommets de S ds G).

$O(M^3)$ - Calculer un couplage de poids minimum sur G_S.

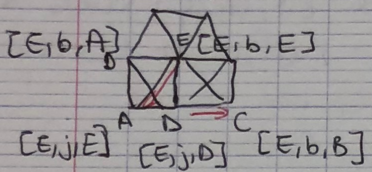
$O(|E|)$ - doubler les arêtes des plus courts chemins correspondant au couplage ds G.

$O(|E|)$ - calculer un cycle eulerien ds G_S

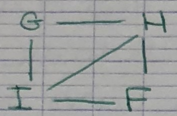
$O(|V|^3)$

M → couplage ≡ matching

M = {(A, D), (B, C)}

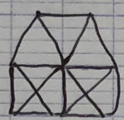


on a détecté un cycle impaire
I = EADBC bourgeon.



I → F chemin améliorant
mais pas chemin ds le
graphe de départ.

M = {(A, D), (B, C), (E, F)}



Notes: ds un graphe biparti, il est impossible
de créer un cycle de taille impaire ⇒
pas de bourgeon dans Edmonds.

- Le couplage maximum peut aussi être calculé
cme un flot maximum

- Pour qu'un couplage parfait existe (touche ∀ les
sommets) il faut que ∀ les sommets soient
paires

$M = \frac{|V|}{2}$