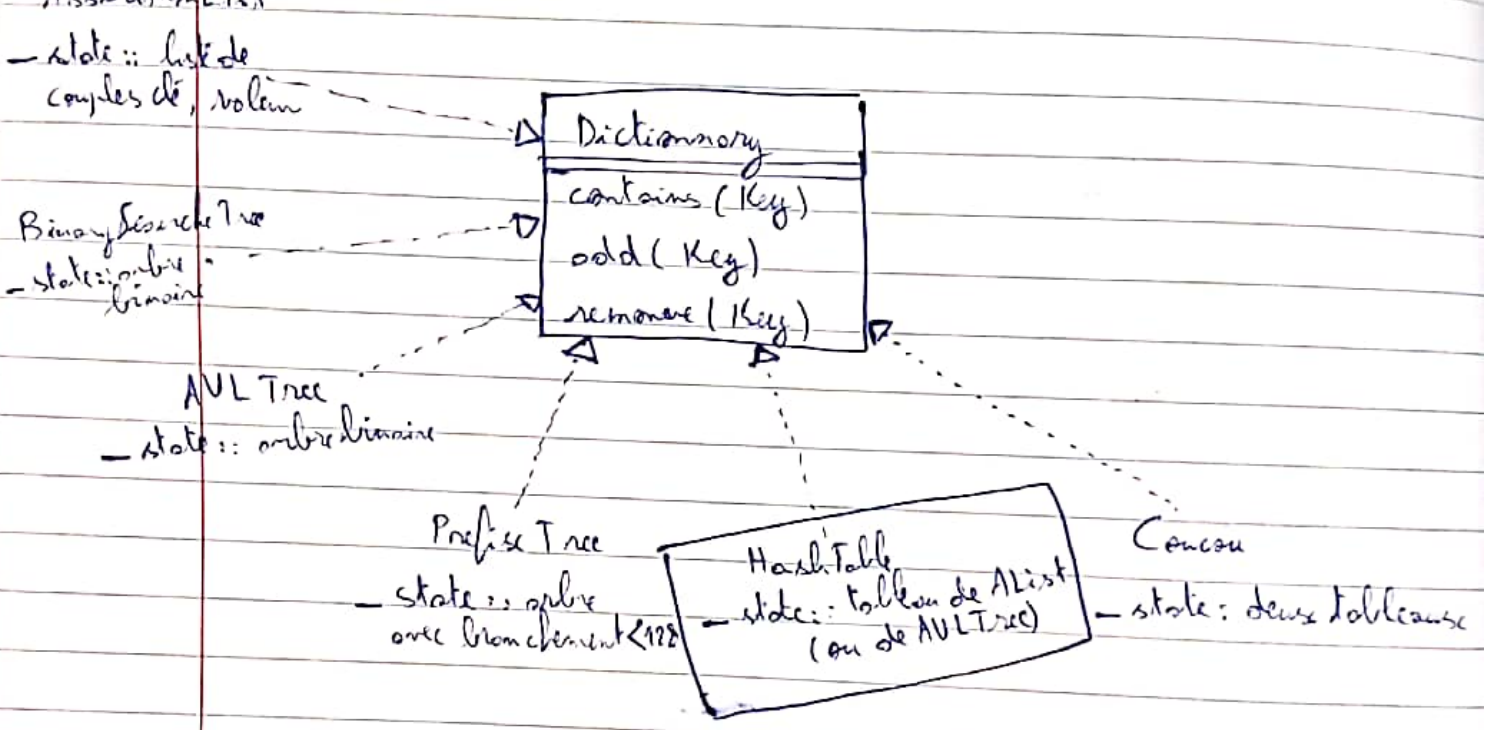
Revision :

25/11/2022

Cours 7 :

Implementation des dictionnaires :



Complexité :

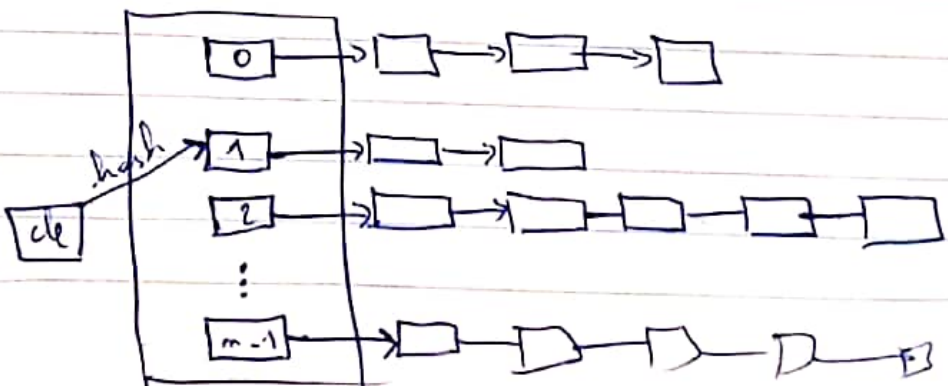
- Au pire des cas :

LocKey	Remove	add
O(Log n)	O(Log n)	O(Log n)

où n est le nombre de clés stockées

- Optimale si on se restreint à certaines opérations élémentaires (comparaison de clés)
- peut on faire mieux ?

Chainage :



Aperçu :

- on souhaite stocker un ensemble de Keys de taille K dans un tableau de taille m
- on utilise une fct :

$$hash : Keys \longrightarrow \{0, \dots, m-1\}.$$

- si $K > m$, alors on ne peut pas éviter de collissions

Algorithmique :

Implementation de contains

```
def contains(cle) :
    calculer i = hash(cle)
    Soit dico = T[i]
    retourner dico.contains(cle)
```

similaires pour add et remove.

Performances :

Hypothèse :

- Calculer hash(cle) prend temps $O(1)$.
- Accéder à la case i du tableau prend temps $O(1)$
- la distribution de hash est uniforme.

paramètre :

- m taille de la table
- K nombre de clés presentes dans la table
- Facteur de charges

$$\alpha = \frac{K}{m}$$

on a $\alpha = E[\text{taille d'une liste}]$ (a demontrer)

Performances :

- En moyenne : $O(1+\alpha)$
- Pire cas : $O(K)$
  si toutes les clés sont stockés dans la même case

$E[\text{len}(T[\text{hash}(cle)])] = \frac{K}{m}$

Par conséquent le cout de contains, add, remove est $O(1+\frac{K}{m})$

Discussion :

- Si K est "petit" par rapport à n.
  alors on peut considérer $\alpha = \frac{K}{n}$ une constante
- En moyenne, on obtient que
  le temps de lookup / contains est $O(1+\alpha) = O(1)$
- Au cas le facteur de charge devient trop grand
  on double la taille de la table (et on réorganise les clés).
- Cette opération (doubler la taille de la table) ne se fera que très rarement :
  croissance exp de la table
  vs ajout en par en des clés.

Probabilités que la liste T[i] a taille p, p fixé

$$P[\text{len } T[i] = p] = \binom{K}{p} \cdot \left(\frac{1}{n}\right)^p \cdot \left(1 - \frac{1}{n}\right)^{K-p}$$

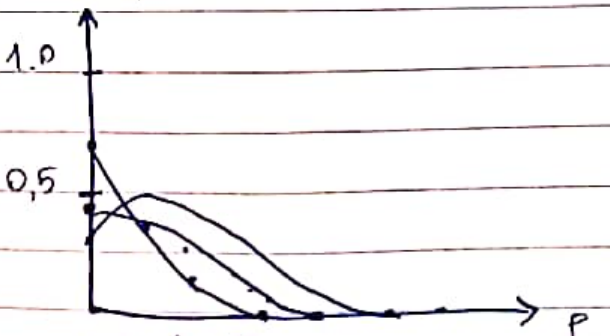
$$\leqslant \frac{K^p}{p!} \cdot \left(\frac{1}{n}\right)^p \cdot \left(\left(1 - \frac{1}{n}\right)^n\right)^{\frac{K-p}{n}}, \quad \binom{K}{p} = \frac{K(K-1)\dots(K-p+1)}{p!}$$

$$\leqslant \left(\frac{K}{n}\right)^p \cdot \frac{e^{p/n - K/n}}{p!} \qquad \left(1 - \frac{1}{n}\right)^n \leqslant e^{-1}$$

$$= \alpha^p \cdot \frac{e^{p/n - K/n}}{p!}$$

$$\propto \alpha^p \cdot \frac{e^{-\alpha}}{p!}$$ si p est petit par rapport à n.

$$f(p) = \frac{\alpha^p}{p!} e^{-\alpha}$$



$\alpha = 1,5$

$\alpha = 0,2$

$\alpha = 0,5$

Répartition de 800 000 clés sur un tableau de taille 1 000 000 ($\alpha = \frac{k}{m} = 0{,}8$) :

p	cases à p clés
0	449328
1	359463
2	143785
3	38342
4	7668
5	1226
6	163
7	18
8	1

* Fonctions de dispersion (ou hachage)

Souhaits

Construire

$hash : Keys \longrightarrow \{0, \dots, n-1\}$

telle que

- hash est uniformément distribuée
- hash est facile à calculer (temps $O(1)$).

* Dispersion par division :

- On choisit une bijection injective $h : Keys \longrightarrow \mathbb{N}$
- Par exemple, si $Keys = ASCII^*$:

$$h(a_0 a_1 \dots a_m) = \sum_{i=0}^{m} (int)\, a_i \cdot 128^i$$

- On utilise ensuite $i \longmapsto i \bmod n$. C'est à dire, on pose

$hash(cle) = h(cle) \bmod n$.

- Attention, si n est mal choisi, e.g. si $n = 128$, alors

$hash(a_0 a_1 \dots a_m) \bmod n = h(a_0)$

Donc, $T[hash(a_0 \dots a_m)]$ contient tous les préfixes $a_0 \dots a_i$.

Familles de fonctions

- Nous pourrons avoir besoins de plusieurs fcts de dispersion
- Exemple :
  si p est premier et $h(cle) \leq p$, pour tout $cle \in Keys$,
  on peut choisir $a \in \{1, \dots, p-1\}$ et $b \in \{0, \dots, p-1\}$
  et poser $hash_{a,b}(cle) = ((a \cdot h(cle) + b) \bmod p) \bmod m$
- $hash_{a,b}$ est une famille de fcts de dispersion

Dispersion par multiplication :

- On choisit une fct injective $h : Keys \longrightarrow \mathbb{N}$
- On choisit A t.q $0 < A < 1$
- On pose

$$hash = \lfloor m \cdot ((h(cle) \cdot A) \bmod 1) \rfloor$$

avec

$$x \bmod 1 = x - \lfloor x \rfloor \quad \text{(partie fractionnaire de } x\text{)}$$

- La valeur m (la taille du tableau) n'est pas critique
- D. Knuth suggère d'utiliser $A = \frac{\sqrt{5}-1}{2}$

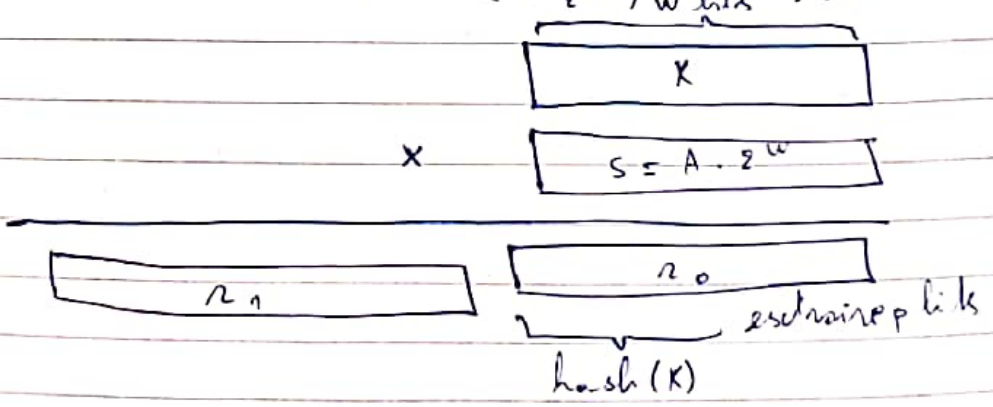
Exemple :

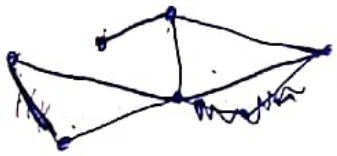
Formule (avec $K = h(cle) \in \mathbb{N}$)

$$hash(K) = \lfloor m \cdot ((K \cdot A) \bmod 1) \rfloor$$

Si $A = \frac{S}{2^w}$ avec S entier sur w bits et $m = 2^p$, alors :

$$hash(K) = \lfloor 2^p \cdot \left(\left(\frac{K \cdot S}{2^w} \right) \bmod 1 \right) \rfloor$$



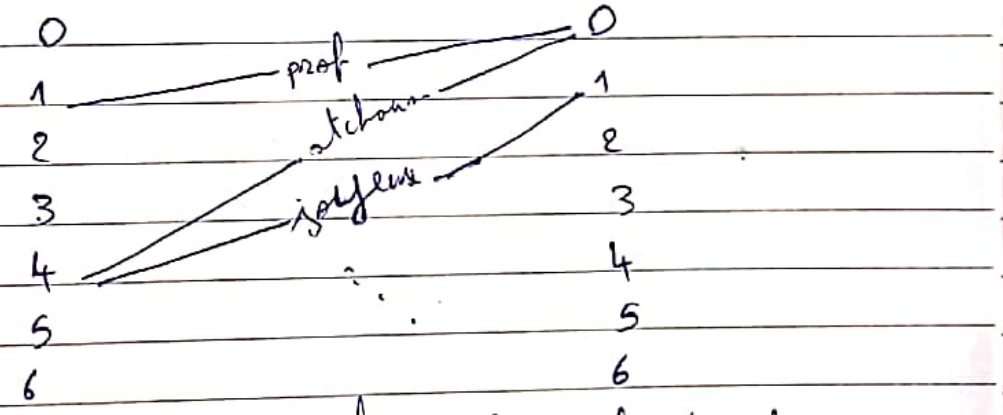


* technique de Coucou :

clé	h_1	h_2	0
Joyeux	4	1	1
Atchoum	4	0	2
Prof	1	0	3
Simplet	5	2	4
Grincheux	6	5	5
Timide	5	1	6
Dormeur	5	3	

T_1	T_2
	Atchoum
Prof	Timide
	Simplet
Joyeux	
Dormeur	
Grincheux	

* Graphe du CouCou :



Lemme : Si une insertion échoue, alors il préexiste un cycle dans le graphe du coucou

* Probabilité d'échec d'insertion de K clés

Th :

En moyenne, avec un tel $\beta (\sim 0,79)$, on arrive à caser toutes les K clés avant

$$\frac{1}{1-\pi} \leq \frac{1}{1-\frac{\beta^2}{2}\frac{1}{1-\beta^2}} \sim 5,887\ldots$$

Iteration du pile ou face :

- $P[\text{pile}] = p$, $P[\text{face}] = 1 - p$
- Combien d'iterations pour obtenir pile ?
- Espace de proba : $\Omega = \{\text{pile}, \text{Face}\}^{\mathbb{N}}$.
- Variables aleatoire : $X : \Omega \to \mathbb{R}$.

  $X(w)$ = premiere occurence de pile dans le mot infini w
- $P[X = j] = p(1-p)^{j-1}$

Theorème : $E[X] = \frac{1}{p}$

Car

$$E[X] = p + (1-p)(1 + E(X))$$

Insertion, temps moyen :

- Si echec dans l'insertion,

  on repart de zero (reconstruction de la table) !!!!
- Reconstruction : on tire deux nouvelles fcts, et on essaie à nouveau. Temps reconstruction : $O(K)$
- E est le nombre moyen d'echecs de reconstruction de la table On considere E une constante
- On a

$$P[\text{echec avec } K \text{ cles}] \leq \frac{K}{m^2} \cdot \frac{1}{1 - (\frac{K}{m})^2} = O(\frac{1}{K})$$

- Donc

$$E[\text{insertion}] = O(1) + O(\frac{1}{K} \cdot E \cdot K) = O(1)$$

- Monolite :

Lookup	Remove	Add	
$O(1)$	$O(1)$	$O(1)$ (*)	(*) en moyenne amortisé

Revision:

Cours 8: Algorithmes d'Approximation

Ordonnancement:

Ordonnancement des taches:

- m machines $M_1, \dots, M_m$.
- n taches, chaque tache prenant temps p_j $(j = 1, \dots, n)$.

Solution optimale (souhaitée):

- une partition des taches en m ensembles (ou listes) $J_1, \dots, J_m$
- telle que, en posant

$$t_i := \sum_{j \in J_i} p_j \quad , \quad T := \max\{t_1, t_2, \dots, t_m\}$$

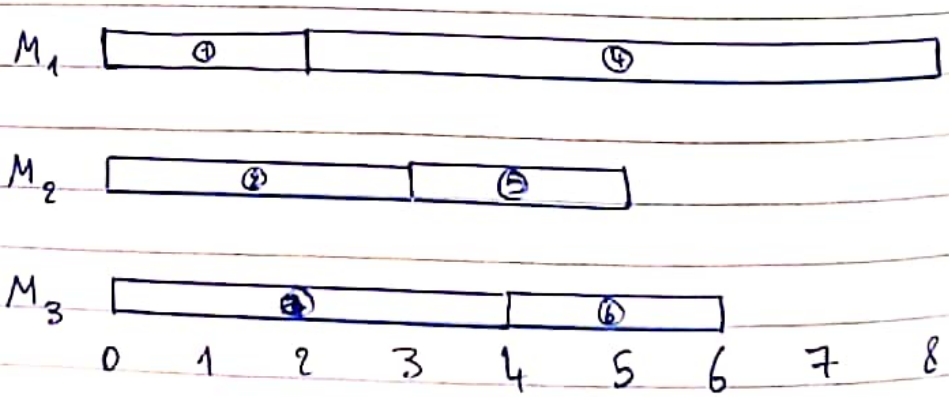
T soit minimum.

C'à d: on souhaite trouver une affectation des taches aux machines de façon à minimiser le temps global de calcul.

* Algorithme LSA (List Scheduling Algorithm)
  - on affecte la prochaine tache à la première machine qui est libre

Exemple: liste de taches à repartir

2 3 4 6 2 2



Remarque:

- Une Solution optimale prends temps global 7.
- Pire cas ?

* LSA comme algorithme d'approximation :

Prop : LSA est un algorithme d'approximation avec facteur 2.

C'est à dire : pour toute instance I du problème, soient
- T_I le temps d'une solution calculée par LSA sur entrée I
- OPT_I le temps d'une solution optimale sur cette instance.

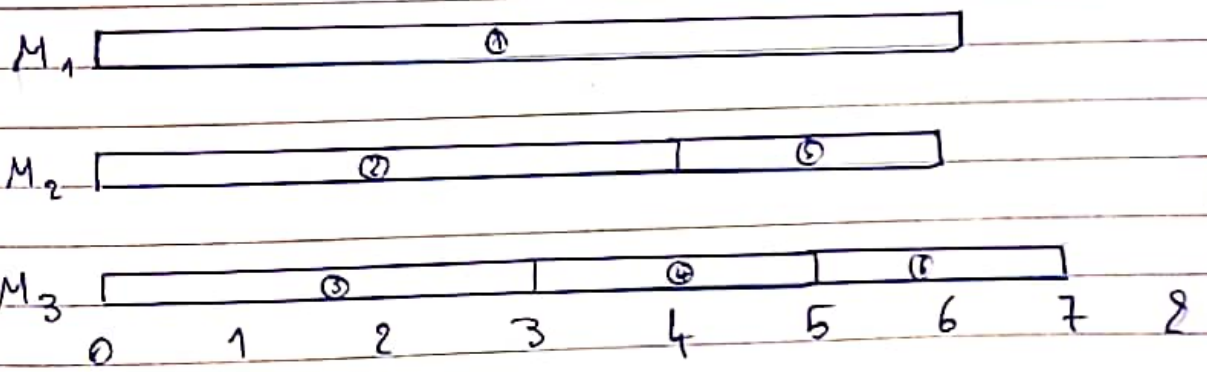
On a

$$OPT_I \leq T_I \leq 2 \cdot OPT_I .$$

* Algorithm LPT (Largest Processing Time)
- On trie les tâches par durée décroissante
- On affecte la prochaine tâche à la première machine qui est libre.

Exemple, liste triée de tâches à répartir :

6 4 3 2 2 2



* Facteur d'approximation de LPT :

Prop : LPT est un algorithme d'approximation avec facteur $\frac{4}{3}$

c'est à dire : $T_I \leq \frac{4}{3} \cdot OPT_I$.

* Algorithmes d'approximation :
- Problème d'optimisation :
  - ▷ On se donne un ensemble d'instances du problème
  - ▷ Une instance peut avoir plusieurs solutions.
  - ▷ (Eventuellement) Les solutions cherchées doivent satisfaire des contraintes.
  - ▷ Pour chaque instance du problème, on cherche une solution S qui minimise / maximise une fct f (fonction objectif)

- Trouver une solution optimale d'une instance I (dont la valeur sera notée OPT_I) est difficile.
- Le problème devient facile si on cherche une solution dont la valeur approche OPT_I.

* Définition algorithme d'approximation
Soit $\mathcal{I}$ l'ensemble des instances d'un problème dont la fct objectif, à minimiser (resp (maximiser)), est f.

Déf :
Un algorithme A est un algorithme d'approximation avec facteur $\delta \geq 1$ si, pour tout $I \in \mathcal{I}$
$$f(A(I)) \underset{\text{resp} \geq}{\leq} \delta \cdot OPT(I)$$
où
- $A(I)$ est la solution calculé par A sur entrée I,
- $OPT(I)$ est la valeur de f d'une solution minimisant f (resp maximisant)

Remarques :
- La difficulté (de calculer une solution optimale) arrive souvent du fait qu'on cherche des valeurs optimales entières
- La difficulté peut disparaître si on s'autorise à produire des solutions à valeurs réelles / rationnelles (ce qu'on ne veut pas, en général).
- La valeur optimale réelle (des solutions à valeurs réelles) est une borne (inf ou sup) pour la valeur optimale entière
- Souvent, le facteur δ dépend de la taille $|I| = n$ de l'instance I, e.g. $\delta = \delta_n$. Dans le cas que nous couvrons, ce n'est pas le cas et la perte de facteur constant

* Sac à dos.

* Le problème du sac à dos (Knapsack)

Pb: Instance $(p_1, v_1), (p_2, v_2), \dots, (p_n, v_n)$ et P.
- n objets,
- chaque objet possède un poid $p_i \in \mathbb{N}$ est une valeur $v_i \in \mathbb{N}$
- $P \in \mathbb{N}$ est la capacité total (un poids du sac à dos

Solution:
- un sous-ensemble $Sac \subseteq \{1, \dots, n\}$ tq $\sum_{i \in Sac} p_i \leq P$
- fonction objectif à maximiser :

$$f(Sac) = \sum_{i \in Sac} v_i$$

c.a.d : remplir le Sac à dos en maximisant la valeur des objets.

Algo Sac à dos glouton :
1) Trier les objets par ordre décroissant des rapports v_i / p_i.
2) Calculer $V_{max} \{v_i \mid i = 1, \dots, n\}$
3) $Sac := \emptyset$, poids_residuel := P
4) pour i : 1, ..., n faire :
   si $p_i \leq$ poids_residuel :
   $Sac := Sac \cup \{i\}$.
   poids_residuel := poids_residuel $- p_i$
5) Valeur_Glouton = max $(\sum_{i \in Sac} v_i, V_{max})$
6) Retourner Valeur_Glouton

Prop :
L'algorithme Sac à dos_Glouton est un algorithme d'approximation de facteur $\frac{1}{2}$.

Remplissage des boites :

bin packing :

Prob, Instance $s_1, \dots, s_n$

- n objets
- chaque objet possède une taille $s_i \in R$ avec $0 < s_i < 1$.

Solution :

- Une partition $P_1, \dots, P_K$ de $\{1, \dots, n\}$ t.q., pour tout $i \in \{1, \dots K\}$

$$\sum_{j \in P_i} s_j \leq 1$$

- Cet objectif a minimiser

$$f(P_1, \dots, P_K) = K.$$

cdd : on souhaite ranger les objets dans un minimum de boites de taille unitaire

Algorithme Nextfit :

L'un après l'autre, on range un objet

- dans la dernière boite, si l'on peut.
- sinon, on l'ajoute dans une nouvelle boite

Exemple, tailles des objets à ranger :

0,6 0,7 0,2 0,4 0,1

Rmq. Une solution optimale utilise 2 boites

Prop : Nextfit est un algo de approximation avec facteur 2

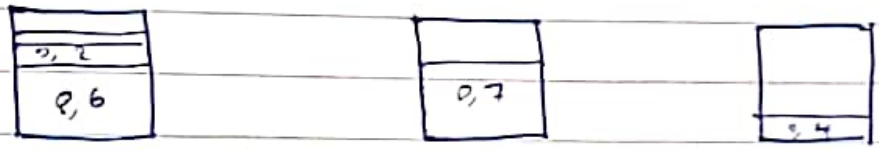
* Algorithme FirstFit.

On range les objets, l'un après l'autre,

- dans la première boite (la plus à gauche) pouvant les accueillir.

Exemple : tailles des objets à ranger :

0,6 0,7 0,2 0,4 0,1



c'est algo de facteur 1,7

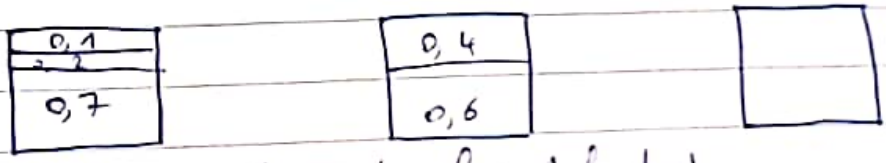
* Algo FirstFit Decreasing

On range les objets

- par taille décroissante
- dans la 1ère boite pouvant les accueillir

Exemple : taille des objets à ranger

0,7 0,6 0,4 0,2 0,1



de facteur 1,5 (plus 1 boite)

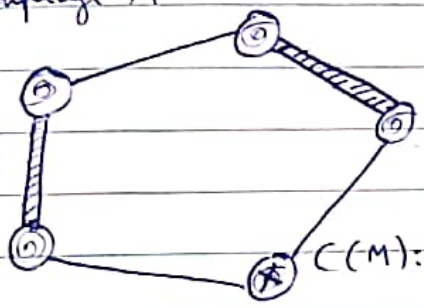
15/11/2022

Cours 10:

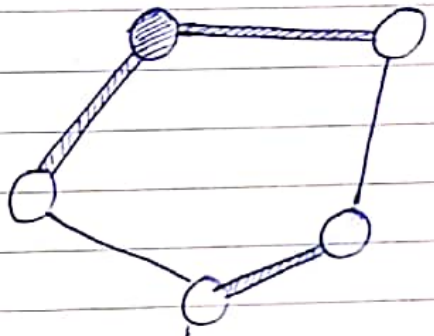
* Couplages, defs, exemples

exemple:

Couplage M



C(M): ~~Les~~ sommets couverts



def: Soit $G = (V, E)$ un graphe non-orienté
Un couplage de G est un sous-ensemble $M \subseteq E$ d'arêtes tel que pour tout $v \in V$, il existe au plus un seul $e \in M$ tq $v \in e$
c à d, arêtes distinctes de M ne partagent pas d'extremités:
$e_1, e_2 \in M$ et $e_1 \cap e_2 \neq \emptyset$ implique $e_1 = e_2$
On denote par C(M) l'ensemble de sommets couverts par M, $C(M) := \{v \in V \mid$ il existe $e \in M$ tel que $v \in e\}$

Def: Un couplage M de (V, E) est:

- maximal, si
  $M \subseteq M'$ et M' couplage implique $M = M'$
- maximum, si
  plus grand nombre possible d'arêtes $card(M) = max\{card(N) \mid N \subseteq E$ est un couplage de $(V,E)\}$
- parfait, si,
  $\forall v \in V$, il existe $e \in M$ t.q $v \in e$,
  c à d, si $V = C(M)$

Lemme: Si M est un couplage parfait de (V, E), alors la cardinalité de V est pair.

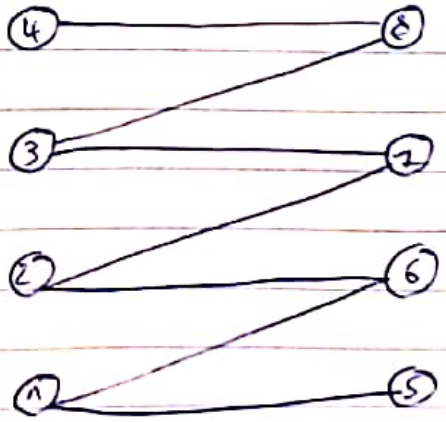
* Deux probs:
1) Etant donné (V, E), trouver un couplage max.
2) Etant donné (V, E), y a-t-il un couplage parfait ?

* méthode de Berge

* Chemins augmentants
Soit M un couplage de (V, E)
Def: un chemin $\Pi = v_0 v_1 \dots v_m$ est
- alternant (ou M-alternant) ssi
  - $v_{i-1}v_i \in M$ et $v_i v_{i+1} \notin M$, $i = 1, \dots, n-1$ ou
  - $v_{i-1}v_i \notin M$ et $v_i v_{i+1} \in M$, $i = 1, \dots, n-1$
- augmentant (ou M-augmentant), s'il est alternant et $v_0, v_n \notin C(M)$

Th: (Berge 1957) Un couplage M est maximum si et seulement si il n'existe pas un chemin M-augmentant

* Couplages dans les graphes bipartis
Def: Un graphe (V, E) est biparti ssi $V = V_0 \cup V_1$, et $uv \in E$ avec $u \in V_i$ implique $v \in V_{1-i}$.
Ex:



* Couplage biparti max par la meth de Ford-Fulk

- Transformer le graphe biparti en reseau de transport :
  - ▷ orienter toutes les arcles de V_0 vers V_1
  - ▷ ajoute une source, et la relier a tous les sommets dans V_0
  - ▷ ajouter un puits, et le relier à tous les sommets de V_1,
  - ▷ on donne à chaque arête capacité 1.
- On construit un flot max sur ce reseau
- Les arêtes dont le flot 1/1 sont celles du couplage

Exemples : dans le cours

* Thms de Holl (1935)

Th : Un graphe biparti a un couplage tel que $V_0 \subseteq C(M)$ ssi

$$\text{card}(S) \leq \text{card}(N(S))$$

pour tout sous-ensemble $S \subseteq V_0$

Ici

$$N(S) := \{v \in V \mid \exists u \in S \text{ t.q } uv \in E\}$$

Remarque Un tel couplage est necessairement maximum.

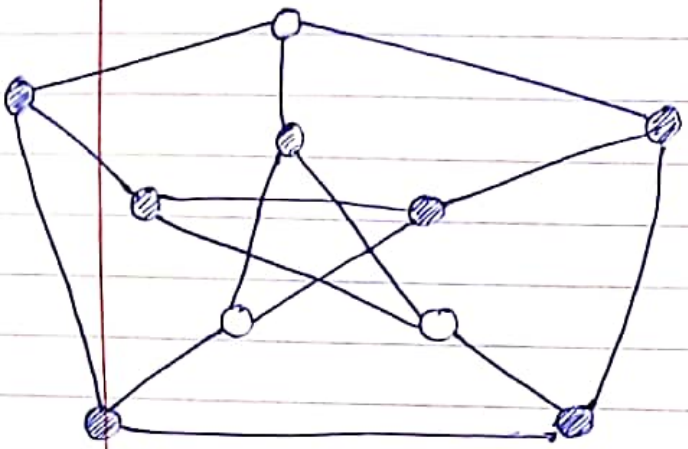
Corollaire. Un graphe biparti possede un couplage parfait ssi $\text{card}(V_0) = \text{card}(V_1)$ et $\text{card}(S) \leq \text{card}(N(S))$ pour tout $S \subseteq V$

* Couplages et couvertures :

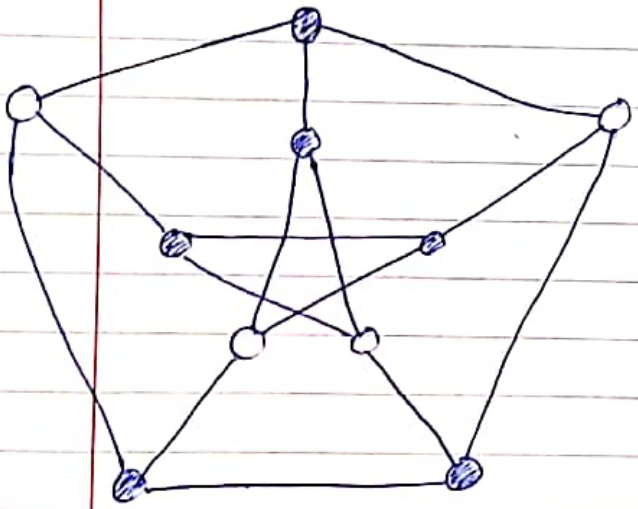
Def :

- Une couverture d'un graphe (V, E) est un sous-ensemble $K \subseteq V$ tel que, pour tout $e \in E$, il existe $v \in K$ tel que $v \in e$.
- Une couverture K de (V, E) est minimum s'il n'existe pas une couverture K' de (V, E) telle que $\text{card}(K') < \text{card}(K)$
- Une couverture K de (V, E) est minimale s'il n'existe pas un sous ensemble propre $K' \subsetneq K$ qui est aussi une couverture de (V, E)

Minimale, mais pas minimum



minimum, donc aussi minimale :



Le théorème de König-Egerváry

Lemme. Si M est un couplage et K est une couverture, alors

$$card(M) \leq card(K)$$

Par conséquent, si $card(M) = card(K)$, alors M est un couplage maximum et K est une couverture minimum.

Théorème : (König-Egerváry) Si G est un graphe biparti, alors M est un couplage maximum ssi il existe une couverture minimum K telle que $card(K) = card(M)$

Autrement : dans un graphe biparti le nombre d'arêtes d'un couplage maximum est égal au nombre de sommets d'une couverture minimum

La formule de Berge - Tutte

Soit : $\nu(G)$:= taille d'un couplage maximum de G

Thm : $\nu(G) := \frac{1}{2} \min \{ n - (o(G \setminus S) - card(S)) \mid S \subseteq V \}$

Remarque :

Lemme : Soient $p : A \to \mathbb{R}$ et $c : B \to \mathbb{R}$ tels que, pour tout $a \in A$ et $b \in B$ on a :

$$p(a) \leq c(b)$$

Si $a_0 \in A$ et $b_0 \in B$ tels que

$$p(a_0) = c(b_0)$$

alors a_0 est un optimum (maximum) parmi les A et b_0 est un optimum (minimum) parmi les B.

Par exemple :

- A les flots, B les coupes
- A les couplages, B les couvertures.

* Couplages, en general
  Le theoreme de Petersen
  la formule de Tutte - Berge

* Probleme
  Est ce que tout graphe 3-regulier possede un couplage parfait ?
  Un graphe (V, E) est K-regulier si $deg(v) = K$, pour tout $v \in V$

* Th de Petersen

Th : Soit G un graphe (connexe) 3-regulier sans arete separante (pont, bridge). Alors G possede un couplage parfait

Def :

$e \in E$ est une arete separante de (V, E) si le nombre de composantes connexes de $(V, E \setminus \{e\})$ est plus grand que celui de (V, E).

Soit

$o(G)$ = nombre de composantes connexes de cardinalité impaire

$G \setminus S := (V \setminus S, \{ e \in E \mid e \subseteq V \setminus S \})$

on utilisera :

Théorème : (Tutte) Un graphe G possède un couplage parfait si, et seulement si,

$$o(G \setminus S) \leq \text{card}(S)$$

pour tout sous-ensemble $S \subseteq V$

* Ensemble libre d'un couplage

Pour M un couplage de (V, E), posons

$$U_M := V \setminus C(M)$$

Rmq : On a les relations suivantes :

$$\text{card}(U_M) = \text{card}(V) - 2\,\text{card}(M),$$

$$\text{card}(M) = \frac{\text{card}(V) - \text{card}(U_M)}{2}$$

Par conséquent, $\text{card}(M)$ est maximum ssi $\text{card}(U_M)$ est minimum

Proposition : Pour tout couplage M et $S \subseteq V$, on a

$$\text{card}(U_M) \geq o(G \setminus S) - \text{card}(S)$$

* Barrière

Déf : Un sous-ensemble $B \subseteq V$ est une M-barrière si

$$\text{card}(U_M) = o(G \setminus B) - \text{card}(B)$$

Rmq : Rappel : on a toujours $\text{card}(U_M) \geq o(G \setminus B) - \text{card}(B)$

Lemme : Si B est une M-barrière, alors $\text{card}(U_M)$ est minimum et M est un couplage maximum.

Déf : Un sous-ensemble $B \subseteq V$ est une barrière si B est une M-barrière pour un quelque couplage (maximum) M.

Th : (Berge-Tutte) Tout graphe possède une barrière.

Remarque : la notion de barrière généralise celle de couverture aux graphes qui ne sont pas bipartis

* Le Thm de Berge-Tutte.

Th : (Tutte) Un graphe G possède un couplage parfait ssi

$$o(G \setminus S) \leq \text{card}(S) \quad \text{pour tout } S \subseteq V$$

$V(G) =$ taille d'un couplage maximum de G

Th : $V(G) = \frac{1}{2} \min \{ n - (o(G \setminus S) - \text{card}(S)) \mid S \subseteq V \}$

23/12/2022 Cours 11 : Graphes Eulériens et Hamiltoniens (et +)

Rappels :

Soit (V, E) un graphe (simple, non orienté)

- Une chaîne est une suite

$$v_0\, e_1\, v_1\, e_2\, v_2 \dots v_{k-1}\, e_k\, v_k$$

telle que $e_i = \{v_{i-1}, v_i\} \in E$

- Une chaîne est fermée si $v_0 = v_k$

Def : Un graphe (V, E) est :

- Eulérien s'il existe une chaîne fermée qui visite une et une seule fois toutes les arêtes de E (cycles Eulérien)
- Hamiltonien s'il existe une chaîne fermée qui visite une et une seule fois toutes les sommets de V (cycle Hamiltonien)

Eulériens

Th : Soit $G = (V, E)$ un graphe connexe (avec au moins deux sommets). Les conditions suivantes sont équivalentes :

(i) G est eulérien

(ii) pour tout $v \in V$, $\deg(v)$ est pair

(iii) il existe une partition des arêtes

$$E = C_1 \cup C_2 \cup \dots \cup C_k$$

telle que C_i est un cycle

a. La conjecture de Hajos sur les cycles (1968)

Conjecture :

Pour tout graphe Eulérien G avec n sommets, il existe une partition $\mathcal{C}$ de E en cycles telle que pour tout $C \in \mathcal{C}$, on a $\text{card}(C) \leq \frac{1}{2}(n-1)$

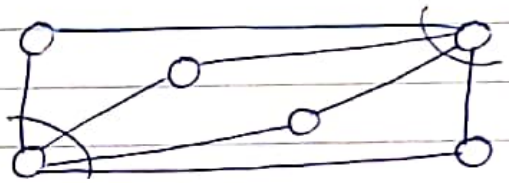
Graphes Hamiltoniens

Une condition nécessaire

Rappel : Soit $G = (V, E)$ un graphe et $S \subseteq V$.

- le graphe induit par S, noté $G[S]$, est (S, E') avec $E' = \{uv \in E \mid u, v \in S\}$.
- le graphe $G \setminus S$ est $G[V \setminus S]$

Prop : Si $G = (V, E)$ est un graph Hamiltonien, alors pour tout $S \subseteq V$ t.q $S \neq \emptyset$, le graph $G \setminus S$ possede au plus $\text{card}(S)$ composantes connexes



Une condition suffisante : le theoreme de Dirac (1952)

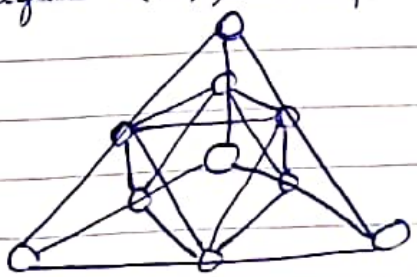
Th : Si un graphe (V, E)

- possede $n \geq 3$ sommets, et
- son degre minimum $\delta(G) \geq \frac{n}{2}$

alors G Hamiltonien

Le graphe de ligne : Eulerienité et Hamiltonicité

Graphe de ligne $L(G)$, exemple :



Graphe de ligne, def formelle

Def : Soit $G = (V, E)$ un graphe. Le graph $L(G)$ est (E, E') avec $\{e, e'\} \in E'$ ssi $e \cap e' \neq \emptyset$

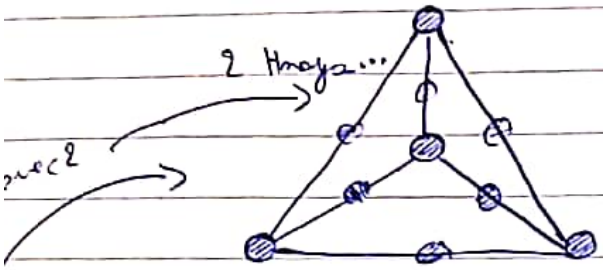
Prop : Si G est Eulerien, alors $L(G)$ est Eulerien et Hamiltonien

Rmq : on peut avoir $L(G)$ Hamiltonien et G pas Eulerien

par Exemple $G = K_4$

Subdivision d'un graphe:

Par l'exemple : $G (= S_0(G))$



↳ par l'exemple $S_1(G)$

Prop: Si G est Eulerien, alors $S_K(G)$ est Eulerien aussi, pour tout $K \geq 0$

Prop: Si G est Eulerien, alors $L(S_K(G))$ est Hamiltonien, pour tout $K \geq 0$.

Th: G est Eulerien si et seulement si $L(S_2(G))$ est Hamiltonien

Interlude : les graphes planaires

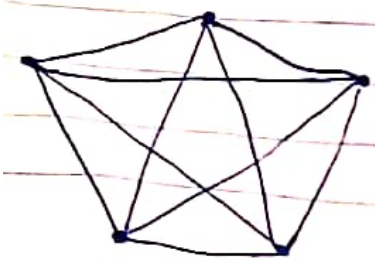
Graphe planaire, definition

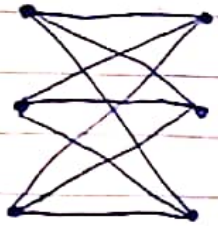
Def: Un graphe $V=(V, E)$ est planaire s'il est possible le dessiner dans le plan sans croiser les arêtes.

Theorème de Kuratowski.

Lemme: Si G est planaire, alors $S_K(G)$ est planaire, pour tout $K \geq 0$.

Theoreme: Si G est planaire ssi il ne contient pas, comme sous-graphe, une subdivision de K_5 ou de $K_{3,3}$





"(Dessin)"

Le graphe duale.

Def: Etant donné un dessin d'un graphe planaire $G=(V,E)$, une face est une portion connexe de $\mathbb{R}^2 \setminus E$.

Posons

$$F = \{X \subseteq \mathbb{R}^2 \setminus E \mid X \text{ est une face de } (V,E)\}$$

Une arête $uv \in E$ sépare X, Y ssi $i_{u,v}([0,1]) \subseteq \bar{X} \cap \bar{Y}$

Le graphe $\tilde{G}$ duale de G est (F,E) où $e \in E$ relie X, Y ssi elle sépare X et Y

Rmq: Le graphe duale peut avoir plusieurs arêtes reliant les mêmes sommets et même des boucles.
Donc il n'est pas en général un graphe simple.

Formule d'Euler:

Th: Si G est (le dessin d') un graphe planaire connexe alors

$$|V| - |E| + |F| = 2$$

Corollaire Soit G un graphe simple planaire d'ordre au moins 3. alors $m \leq 3n - 6$.

Corollaire: K_5 n'est pas planaire

Corollaire: Tout graphe simple planaire a un sommet de de degré au plus 5

Cours 12 : Coloration des graphes

Définition, relations élémentaires

Déf : Soit (V, E) un graphe.

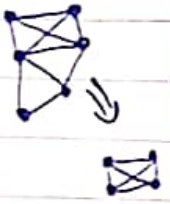
- Une K-coloration (des sommets) de G est une fonction $c : V \longrightarrow \{1, \dots, K\}$ telle que
$$uv \in E \quad \text{implique} \quad c(u) \neq c(v)$$

- Le nombre chromatique de G, noté $\chi(G)$, est le plus petit K tel qu'il existe une K-coloration de G.
$$\chi(G) = \min\{K \mid \text{il existe une } K\text{-coloration de } G\}$$

Rappels sur les graphes bipartis

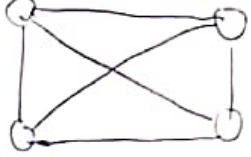
Soit C_n le cycles de longueur n. On a
$$\chi(C_n) = \begin{cases} 2, & n \text{ pair} \\ 3, & n \text{ impair} \end{cases}$$

Prop : Les faits suivants sont équivalents :
1 : $\chi(G) \leqslant 2$,
2 : G est un graphe biparti,
3 : G ne contient pas des cycles de longueur impair.



Cliques et stables :

- Une clique de G est un sous-ensemble $S \subseteq V$ tel que $uv \in E$, pour tout $u, v \in S$, on pose
$$\omega(G) = \max\{\text{card}(S) \mid S \subseteq V \text{ est une clique}\}$$
- Un stable (ou ensemble indépendant) de G est un sous ensemble $S \subseteq V$ tel que $uv \notin E$, pour tout $u, v \in S$, on pose
$$\alpha(G) = \max\{\text{Card}(S) \mid S \subseteq V \text{ est un stable}\}$$



Relation élémentaires.

Lemme: Une fct $c: V \longrightarrow \{1, \dots, K\}$ est une K-coloriation si et seulement si, pour tout $i = 1, \dots, K$, l'ensemble

$$c^{-1}(i) = \{v \in V \mid c(v) = i\}$$

est un stable

Rmq: Ces deux concepts sont donc équivalents:

- Une K-coloriation de (V, E),
- Une partition de V en K stables

Corollaire: on a:

$$\frac{n}{\alpha(G)} \leq \chi(G) \qquad \omega(G) \leq \chi(G).$$

Glouton: algo simple.

Th de Brooks:

Soit $\Delta(G)$ le degré max d'un sommets de G.
En utilisant l'algorithme glouton

$$\chi(G) \leq \Delta(G) + 1$$

Cette inegalité est une egalité si $G = C_n$, n impair, ou $G = K_n$

Th: (Brooks 1941). Si G est un graphe connexe, qui n'est pas cycle impair ni un graphe complet alors

$$\chi(G) \leq \Delta(G)$$

Graphe de Mycielski

Soit $G = (V, E)$ un graphe. Le graphe de Mycielski de G, $M(G) = (V', E')$ est ainsi défini:

$$V' = V \times \{0\} \times \{1\} \cup \{p\}$$

$$E' = \{(u, 0)(v, 0) \mid uv \in E\}$$
$$\cup \{(u, i)(v, 1-i) \mid uv \in E, i \in \{0, 1\}\}$$
$$\cup \{p(v, 1) \mid v \in V\}$$

Th:

- Si $\omega(G) = 2$, alors $\omega(M(G)) = 2$
- $\chi(M(G)) = \chi(G) + 1$

* Graphe de decalage SG_n

On defini $SG_n = (V, E)$ par :

$V = \{(i,j) \mid 1 \leq i < j \leq n\}$

$E = \{(i,j)(j,k) \mid i,j,k \in \{1,\dots,n\}\}$

Th : $\omega(SG_n) = 2$ et $\chi(SG_n) = \lceil \log_2 n \rceil$

* Graphe de Kneser $KG^n_{n,k}$

On definit $KG_{n,k} = (V, E)$ par :

$V = \binom{[n]}{k} = \{S \subseteq \{1,\dots,n\} \mid card(S) = k\}$

$E = \{\{S, S'\} \mid S \cap S' = \emptyset\}$

Rmq : si $k > \frac{n}{2}$ alors $E = \emptyset$

* La conjecture de Kneser

Lemme : Soit $d = n - 2k$. On a $\chi(KG_{n,k}) \leq d + 2$

Conjecture de Kneser (1955) : $\chi(KG_{n,k}) = d + 2$

* le th des quatres couleurs :

Th : il est possible de colorier, en n'utilisant que quatre couleurs differentes, une carte de coupée en region connexes, de sorte que deux regions adjacentes reçoivent toujours deux couleurs distinctes

De façon equivalentes :

Th : Si G est un graphe planaire, alors $\chi(G) \leq 4$

Rappel de planarité

Graphes parfaits

Def : Un graph $H = (S, E')$ est un sous-graphes induit de $G = (V, E)$ si $S \subseteq V$, et, pour $u, v \in S$, $uv \in E'$ ssi $uv \in E$

on note alors $H = G[S]$

Def Un graphe G est parfait si $\chi(H) = \omega(H)$ pour tout sous-graphe induit H de G

Graphes de comparabilité / incomp. —

Soit $(P, \leq)$ un ensemble ordonné (fini)

Def : Le graphe de comparabilité $C(P, \leq) = (V, E)$ est défini par $V = P$ et

$uv \in E$ ssi $u \leq v$ ou $v \leq u$.

Une chaîne C de $(P, \leq)$ est une clique de $C(P, \leq)$. Une anti-chaîne A de $(P, \leq)$ est stable de $C(P, \leq)$

Th : (Dilworth, 1950). Le nombre minimum K de chaînes telles que

$$P = C_1 \cup C_2 \cup \dots \cup C_K,$$

est égal à la largeur de $(P, \leq)$, c'est à dire la taille maximum d'une anti-chaîne

Graphe complémentaire :

Def Pour $G = (V, E)$ le graphe complémentaire de G est $\overline{G} = (V, E')$ avec

$$E' = \{uv \mid uv \in E\}$$

clique de G = stable de $\overline{G}$ stable de G - clique de $\overline{G}$

Th : (Dilworth 1950)

pour $(P, \leq)$ on a $\omega(C(P, \leq)) = \chi(\overline{C(P, \leq)})$

* Les graphes d'incomparabilités sont parfaits:
Un graphe d'incomparabilité est un graphe de la forme $C(P, \leq)$.

Lemme Tous sous-graphe induit d'un graphe d'incomparabilité est un graphe d'incomparabilité

Corollaire
Les graphes d'incomparabilités sont des graphes parfaits

Lemme:
$$\omega(C(P, \leq)) = \chi(C(P, \leq))$$

Th: (Lovasz)
Un graph parfait ssi son complémentaires $\overline{G}$ est parfait

Th (Hoginal, Lovasz) Un graphe G est parfait ssi $card(S) \leq \alpha(H) \cdot \omega(H)$

Cours 12: résumé Coloriage des graphes

Coloration: chaque sommet a une couleur
Coloration propre: chaque sommet a une couleur différente de celles de ses voisins.
Donné $G = (V, E)$, Trouver une coloration propre de G utilisant le nombre minimal de couleurs

Algorithme: