

Cours 1 - Introduction

Semestre 1 – Année 2022-2023 – Université Côte D'azur

Christophe Crespelle

`christophe.crespelle@univ-cotedazur.fr`



Graphes et programmation dynamique

- De quoi s'agit-il ?
 - ▶ **Graphes** : un objet tres simple

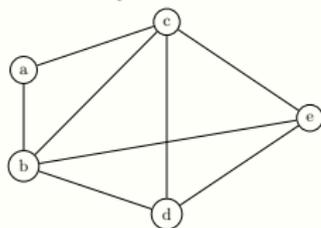


FIGURE – Un graphe a 5 sommets et 8 aretes.

Graphes et programmation dynamique

- De quoi s'agit-il ?
 - ▶ **Graphes** : un objet très simple

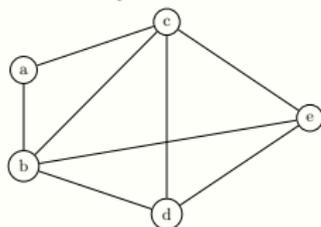


FIGURE – Un graphe a 5 sommets et 8 aretes.

- ▶ **Programmation dynamique** : une approche algorithmique (parmi d'autres)

Graphes et programmation dynamique

- De quoi s'agit-il ?
 - ▶ **Graphes** : un objet très simple

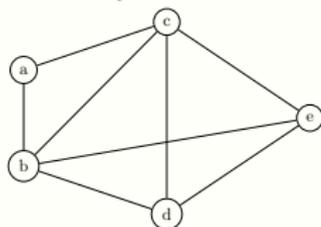


FIGURE – Un graphe a 5 sommets et 8 aretes.

- ▶ **Programmation dynamique** : une approche algorithmique (parmi d'autres)
- A quoi ca sert ?
 - ▶ beaucoup de problemes rencontres en pratique sont formulees en termes de graphes
 - ▶ les donnees sont organisees en graphe
 - ▶ la question a resoudre est un probleme de graphe

Graphes et programmation dynamique

- De quoi s'agit-il ?
 - ▶ **Graphes** : un objet tres simple

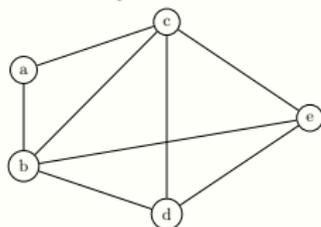


FIGURE – Un graphe a 5 sommets et 8 aretes.

- ▶ **Programmation dynamique** : une approche algorithmique (parmi d'autres)
- A quoi ca sert ?
 - ▶ beaucoup de problemes rencontres en pratique sont formulees en termes de graphes
 - ▶ les donnees sont organisees en graphe
 - ▶ la question a resoudre est un probleme de graphe
- Domaines et contextes d'application :
 - ▶ informatique, industrie, transports, finance, energie, biologie, medecine, etc.

Exemple 1 : implantation de lignes electriques

- **Donnee** : des sites electriques a relier, des voies possibles pour tirer les cables entre eux (avec leurs longueurs)

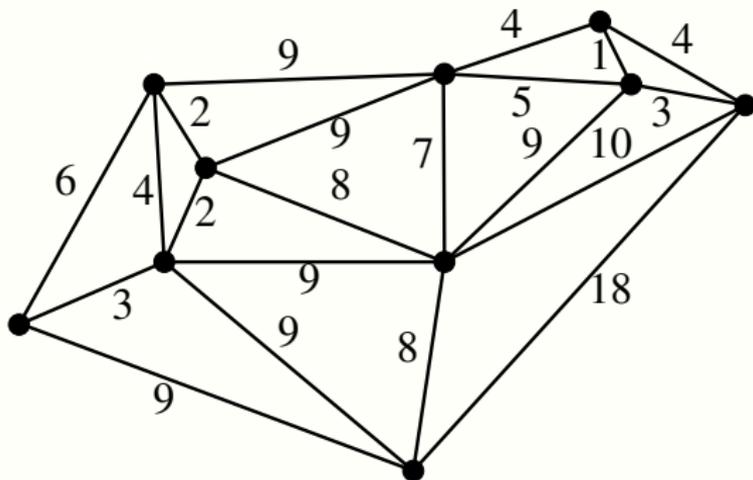


FIGURE – Un probleme d'implantation de ligne electrique.

- **Probleme 1** : alimenter tous les sites en minimisant la longueur de cable utilisee

Exemple 1 : implantation de lignes electriques

- **Donnee** : des sites electriques a relier, des voies possibles pour tirer les cables entre eux (avec leurs longueurs)

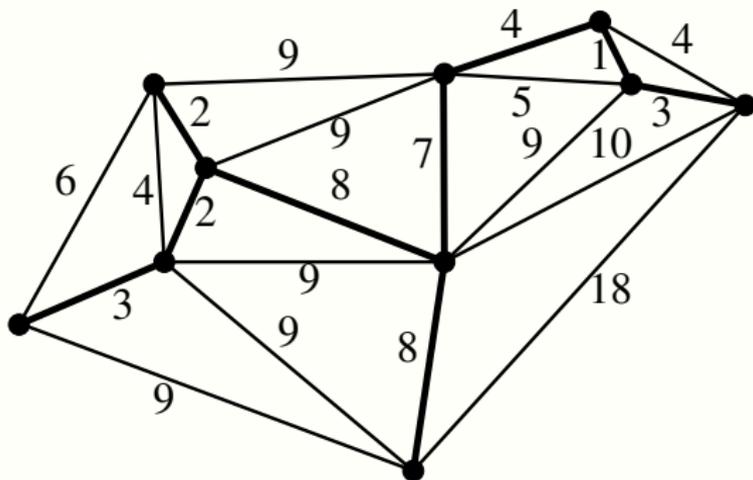


FIGURE – Un probleme d'implantation de ligne electrique.

- **Probleme 1** : alimenter tous les sites en minimisant la longueur de cable utilisee
⇒ **arbre couvrant minimum**

Exemple 2 : equilibrage de charges

- **Donnee** : un ensemble de taches T_1, T_2, \dots, T_n , chacune avec une duree $d(T_i)$, et k postes de travail

FIGURE – Un probleme d'equilibrage de charges avec 12 taches et 4 postes de travail.

- **Probleme** : trouver une affectation des taches sur les postes qui minimise le temps de fin d'execution

Objectifs du cours

- **Differents types de problemes d'optimisation**
 - ▶ Plus courts chemins
 - ▶ Chemin Hamiltonien et voyageur de commerce
 - ▶ Arbre couvrant de poids minimum
 - ▶ Coloration des sommets
 - ▶ Flots maximum / coupe minimum

- **Techniques algorithmiques generales pour les resoudre**
 - ▶ Algorithmes gloutons
 - ▶ Programmation dynamique
 - ▶ Diviser pour regner
 - ▶ ...d'autres qui ne rentrent pas (ou mal) dans ces categories

Organisation pratique du cours

Volume :

- CM : 12h
- TD : 6h
- TP : 6h

Evaluation (en controle continu partiel) : **NON CONTRACTUEL ;)**

- examen final : 60%
- test intermediaire : 20%
- TP note : 20%
 - ▶ un rendu en fin de seance (le 2eme TP)

Emploi du temps : ↙

<https://www.i3s.unice.fr/~ccrespelle/enseignements.html>

- CM, TD et TP aux lucioles, dans une des 2 salles dediees au M1

Reussir l'UE

Comment reussir l'UE ?

Reussir l'UE

Comment reussir l'UE ?

- Apprendre les CM par coeur

Reussir l'UE

Comment reussir l'UE ?

- Apprendre les CM par coeur
- Finir les 2 TP

Reussir l'UE

Comment reussir l'UE ?

- Apprendre les CM par coeur
- Finir les 2 TP
- Savoir faire les TD (corrections integrales fournies)

Reussir l'UE

Comment reussir l'UE ?

- Apprendre les CM par coeur
- Finir les 2 TP
- Savoir faire les TD (corrections integrales fournies)

Que faire si vous etes perdu ?

Reussir l'UE

Comment reussir l'UE ?

- Apprendre les CM par coeur
- Finir les 2 TP
- Savoir faire les TD (corrections integrales fournies)

Que faire si vous etes perdu ?

- Poser des questions en CM

Reussir l'UE

Comment reussir l'UE ?

- Apprendre les CM par coeur
- Finir les 2 TP
- Savoir faire les TD (corrections integrales fournies)

Que faire si vous etes perdu ?

- Poser des questions en CM
- Poser des questions en CM

Reussir l'UE

Comment reussir l'UE ?

- Apprendre les CM par coeur
- Finir les 2 TP
- Savoir faire les TD (corrections integrales fournies)

Que faire si vous etes perdu ?

- Poser des questions en CM
- Poser des questions en CM
- Revoir les notions manquante dans *Introduction a l'algorithmique*, Cormen et al.

Reussir l'UE

Comment reussir l'UE ?

- Apprendre les CM par coeur
- Finir les 2 TP
- Savoir faire les TD (corrections integrales fournies)

Que faire si vous etes perdu ?

- Poser des questions en CM
- Poser des questions en CM
- Revoir les notions manquante dans *Introduction a l'algorithmique*, Cormen et al.
- Retravailler les CM a posteriori

Reussir l'UE

Comment reussir l'UE ?

- Apprendre les CM par coeur
- Finir les 2 TP
- Savoir faire les TD (corrections integrales fournies)

Que faire si vous etes perdu ?

- Poser des questions en CM
- Poser des questions en CM
- Revoir les notions manquante dans *Introduction a l'algorithmique*, Cormen et al.
- Retravailler les CM a posteriori
- Travailler les corrections des TD

Reussir l'UE

Comment reussir l'UE ?

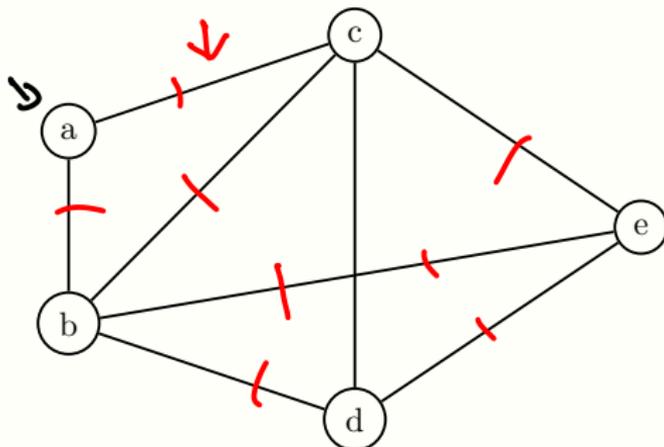
- Apprendre les CM par coeur
- Finir les 2 TP
- Savoir faire les TD (corrections integrales fournies)

Que faire si vous etes perdu ?

- Poser des questions en CM
- Poser des questions en CM
- Revoir les notions manquante dans *Introduction a l'algorithmique*, Cormen et al.
- Retravailler les CM a posteriori
- Travailler les corrections des TD
- Consultez les passages de livre sur le cours (versions electroniques)
 - ▶ *Introduction a l'algorithmique*, Cormen et al.
 - ▶ *Algorithm Design*, Kleinberg et Tardos

Graphe

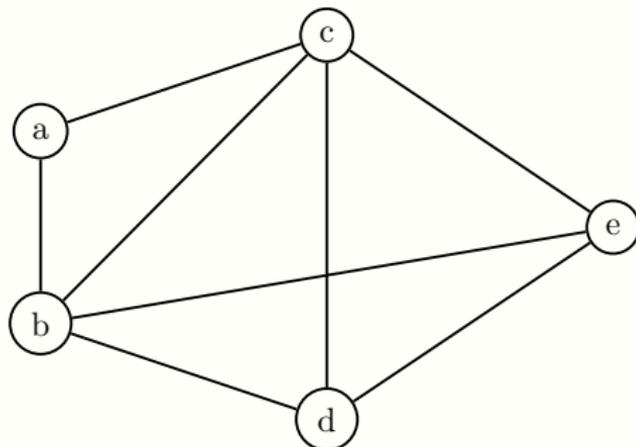
- Un objet simple



- ▶ Un ensemble de sommets : $\{a, b, c, d, e\}$
- ▶ Un ensemble d'arêtes (= paires de sommets) : $\{ab, ac, bc, bd, be, cd, ce, de\}$

Graphe

- Un objet simple au fort pouvoir de modelisation



- ▶ Un ensemble de sommets : $\{a, b, c, d, e\}$
- ▶ Un ensemble d'arêtes (= paires de sommets) : $\{ab, ac, bc, bd, be, cd, ce, de\}$

Des graphes partout !

- **Big data**
- **Systemes complexes**

Des graphes partout !

- **Big data**
- **Systemes complexes**

⇒ **Reseaux complexes** (= graphes)

Des graphes partout !

- **Big data**
- **Systemes complexes**

⇒ **Reseaux complexes** (= graphes)

- dans presque tous les domaines :
 - ▶ informatique, communications, industrie, transports, finance, economie, energie, physique, biologie, genetique, medecine, sciences sociales, poilitique, linguistique, etc.

Algorithmique

- Besoin d'algorithmique ?

Algorithmique

- Besoin d'algorithmique?
 - ▶ Bien implementer (bon langage, bon code,...)
 - ▶ Machines puissantes

⇒ **Ca suffit !**

Algorithmique

- Besoin d'algorithmique?
 - ▶ Bien implementer (bon langage, bon code,...)
 - ▶ Machines puissantes

⇒ **Ca suffit !**

Vraiment ???

Algorithmique

- Besoin d'algorithmique ?
 - ▶ Bien implementer (bon langage, bon code,...)
 - ▶ Machines puissantes

⇒ **Ca suffit !**

Vraiment ???

- Un exemple (basique) : Fibonacci
 - ▶ $fibonacci(0) = 0, fibonacci(1) = 1$
 - ▶ $\forall n \geq 2, fibonacci(n) = fibonacci(n - 1) + fibonacci(n - 2)$

Fibonacci en recursif

Algorithme 1 : implementation recursive

```
1 fibonacci(n)  
2 begin  
3   si  $n = 0$  alors retourner 0;  
4   si  $n = 1$  alors retourner 1;  
5   sinon retourner  $\text{fibonacci}(n-1)$  +  $\text{fibonacci}(n-2)$ ;  
6 end
```

Fibonacci en recursif

Algorithme 1 : implementation recursive

```
1 fibonacci(n)  
2 begin  
3   | si  $n = 0$  alors retourner 0;  
4   | si  $n = 1$  alors retourner 1;  
5   | sinon retourner fibonacci(n-1)+fibonacci(n-2);  
6 end
```

Analyse de complexite :

- T_n le temps de calcul de $fibonacci(n)$

Fibonacci en recursif

Algorithme 1 : implementation recursive

```
1 fibonacci(n)
2 begin
3   si  $n = 0$  alors retourner 0;
4   si  $n = 1$  alors retourner 1;
5   sinon retourner fibonacci(n-1)+fibonacci(n-2);
6 end
```

Analyse de complexite :

- T_n le temps de calcul de $fibonacci(n)$
- $T_n = T_{n-1} + T_{n-2} + 1$

Fibonacci en recursif

Algorithme 1 : implementation recursive

```
1 fibonacci(n)
2 begin
3   si n = 0 alors retourner 0;
4   si n = 1 alors retourner 1;
5   sinon retourner fibonacci(n-1)+fibonacci(n-2);
6 end
```

Analyse de complexite :

- T_n le temps de calcul de $fibonacci(n)$
- $T_n = T_{n-1} + T_{n-2} + 1$
- resolution par le polynome caractéristique : $x^2 - x - 1$

▶ 2 racines : $\frac{1+\sqrt{5}}{2}$ et $\frac{1-\sqrt{5}}{2}$

▶ la plus grande en valeur absolue est $\frac{1+\sqrt{5}}{2}$

$\propto \left(\frac{1+\sqrt{5}}{2}\right)^n + O\left(\frac{1-\sqrt{5}}{2}\right)^n$

Fibonacci en recursif

Algorithme 1 : implementation recursive

```
1 fibonacci(n)
2 begin
3   si n = 0 alors retourner 0;
4   si n = 1 alors retourner 1;
5   sinon retourner fibonacci(n-1)+fibonacci(n-2);
6 end
```

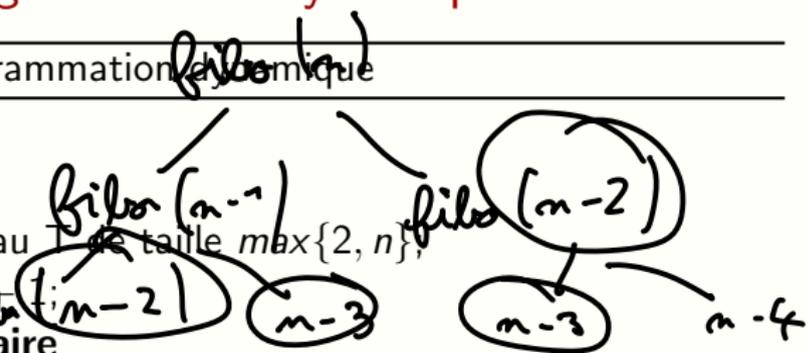
Analyse de complexite :

- T_n le temps de calcul de $fibonacci(n)$
- $T_n = T_{n-1} + T_{n-2} + 1$
- resolution par le polynome caractéristique : $x^2 - x - 1$
 - ▶ 2 racines : $\frac{1+\sqrt{5}}{2}$ et $\frac{1-\sqrt{5}}{2}$
 - ▶ la plus grande en valeur absolue est $\frac{1+\sqrt{5}}{2}$
- on a donc $T_n = \Theta\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right) \approx 1.63^n$

Fibonacci par la programmation dynamique

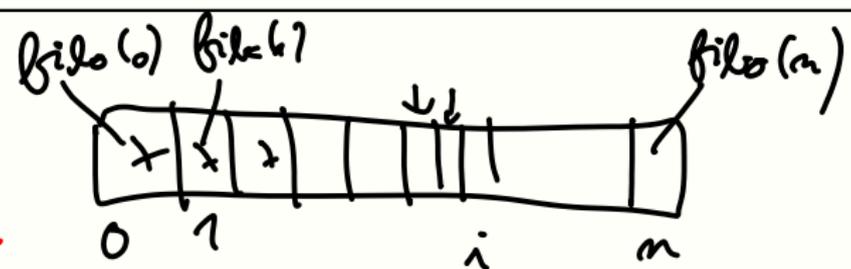
Algorithme 2 : programmation dynamique

```
1 fibonacci(n)
2 begin
3   Reserver un tableau T de taille  $\max\{2, n\}$ ;
4    $T[0] \leftarrow 0$ ;  $T[1] \leftarrow 1$ ;
5   pour i de 2 a n faire
6     |  $T[i] \leftarrow T[i-1] + T[i-2]$ ;
7   fin
8   retourner T[n]
9 end
```



→ temps $O(1)$

$O(n)$
total.



Fibonacci par la programmation dynamique

Algorithme 2 : programmation dynamique

```
1 fibonacci(n)
2 begin
3   Reserver un tableau T de taille  $\max\{2, n\}$ ;
4    $T[0] \leftarrow 0$ ;  $T[1] \leftarrow 1$ ;
5   pour  $i$  de 2 a  $n$  faire
6     |  $T[i] \leftarrow T[i - 1] + T[i - 2]$ ;
7   fin
8   retourner  $T[n]$ 
9 end
```

Analyse de complexite :

- chaque passage dans la boucle en temps $O(1)$
- $n - 1$ passages dans la boucle
- Total : $O(n)$

Fibonacci par l'exponentiation rapide

- D'abord on remarque que pour $n \geq 2$:

$$\begin{bmatrix} \text{fibonacci}(n) \\ \text{fibonacci}(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \text{fibonacci}(n-1) \\ \text{fibonacci}(n-2) \end{bmatrix}$$

Handwritten diagram illustrating the matrix multiplication for Fibonacci numbers. It shows a 2x2 matrix with entries 1, 1, 1, 0. The top row is multiplied by the top element of a vector, f_{n-1} , and the bottom row is multiplied by the bottom element, f_{n-1} . The results are summed to give $f_{n-1} + f_{n-2}$ in the top position of the resulting vector, and f_{n-1} in the bottom position. The final result is labeled f_n .

Fibonacci par l'exponentiation rapide

- D'abord on remarque que pour $n \geq 2$:

$$\begin{bmatrix} \text{fibonacci}(n) \\ \text{fibonacci}(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \text{fibonacci}(n-1) \\ \text{fibonacci}(n-2) \end{bmatrix}$$

- Ainsi, en posant $A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$, on a pour tout $n \geq 2$:

$$\begin{bmatrix} \text{fibonacci}(n) \\ \text{fibonacci}(n-1) \end{bmatrix} = A^{n-1} \cdot \begin{bmatrix} \text{fibonacci}(1) \\ \text{fibonacci}(0) \end{bmatrix}$$

$$a^2 + a^2 = a^3$$

$n=2$ $A \times A > A + A \rightarrow \underbrace{(n-1 \text{ fois})}$

$$O(n)$$

Induct. mult. $a^2 = a^{2^{2-1}} \times a^{2^{2-1}}$

$$a \times a = a^2 \times a^2 = a^4 \times a^4 = a^8 \times a^8$$

Fibonacci par l'exponentiation rapide

- D'abord on remarque que pour $n \geq 2$:

$$\begin{bmatrix} \text{fibonacci}(n) \\ \text{fibonacci}(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \text{fibonacci}(n-1) \\ \text{fibonacci}(n-2) \end{bmatrix}$$

- Ainsi, en posant $A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$, on a pour tout $n \geq 2$:

$$\begin{bmatrix} \text{fibonacci}(n) \\ \text{fibonacci}(n-1) \end{bmatrix} = A^{n-1} \cdot \begin{bmatrix} \text{fibonacci}(1) \\ \text{fibonacci}(0) \end{bmatrix}$$

$$1.63^n$$



$a^n \rightarrow O(\log^m)$
 $n = 2$

ln multi.

$$\ln = \log_2 n$$

$$\begin{array}{ccccccc} & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ & 1 & 1 & 0 & 1 & 0 & 1 \\ a & - & - & - & - & - & - \end{array}$$

$$a^{2^1}$$

$$\begin{array}{cccc} a & a^2 & a^4 & a^8 \\ \wedge & 0 & 1 & 0 \end{array}$$

$$\begin{array}{c} a^{16} \\ \wedge \\ 1 \end{array}$$

Fibonacci par l'exponentiation rapide

- D'abord on remarque que pour $n \geq 2$:

$$\begin{bmatrix} \text{fibonacci}(n) \\ \text{fibonacci}(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \text{fibonacci}(n-1) \\ \text{fibonacci}(n-2) \end{bmatrix}$$

- Ainsi, en posant $A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$, on a pour tout $n \geq 2$:

$$\begin{bmatrix} \text{fibonacci}(n) \\ \text{fibonacci}(n-1) \end{bmatrix} = A^{n-1} \cdot \begin{bmatrix} \text{fibonacci}(1) \\ \text{fibonacci}(0) \end{bmatrix}$$

Algorithme 3 : Fibonacci par exponentiation rapide de matrice

1 fibonacci(n)

2 begin

3 Calculer $\begin{bmatrix} x_{n-1} & y_{n-1} \\ z_{n-1} & t_{n-1} \end{bmatrix} = A^{n-1}$ par l'exponentiation rapide;

4 retourner x_{n-1}

5 end

log n

Fibonacci par l'exponentiation rapide

Analyse de complexite :

- multiplication de matrices 2×2 en temps $O(1)$

Fibonacci par l'exponentiation rapide

Analyse de complexite :

- multiplication de matrices 2×2 en temps $O(1)$
- exponentiation rapide en temps $O(\log n)$

Fibonacci par l'exponentiation rapide

Analyse de complexite :

- multiplication de matrices 2×2 en temps $O(1)$
- exponentiation rapide en temps $O(\log n)$

• Total : $O(\log n)$ pour le calcul de $\text{fib}(n)$

Fibonacci par l'exponentiation rapide

Analyse de complexite :

- multiplication de matrices 2×2 en temps $O(1)$
- exponentiation rapide en temps $O(\log n)$
- Total : $O(\log n)$

Rappel de l'exponentiation rapide : calcul de a^n

- ecriture binaire de $n = \sum_{0 \leq i \leq \lfloor \log_2 n \rfloor} b_i 2^i$

Fibonacci par l'exponentiation rapide

Analyse de complexite :

- multiplication de matrices 2×2 en temps $O(1)$
- exponentiation rapide en temps $O(\log n)$
- Total : $O(\log n)$

Rappel de l'exponentiation rapide : calcul de a^n

- écriture binaire de $n = \sum_{0 \leq i \leq \lfloor \log_2 n \rfloor} b_i 2^i$
- calcul des a^{2^i} grace a $a^{2^{i+1}} = a^{2^i} * a^{2^i}$

Fibonacci par l'exponentiation rapide

Analyse de complexite :

- multiplication de matrices 2×2 en temps $O(1)$
- exponentiation rapide en temps $O(\log n)$
- Total : $O(\log n)$

Rappel de l'exponentiation rapide : calcul de a^n

- écriture binaire de $n = \sum_{0 \leq i \leq \lfloor \log_2 n \rfloor} b_i 2^i$
- calcul des a^{2^i} grace a $a^{2^{i+1}} = a^{2^i} * a^{2^i}$
- $a^n = \prod_{0 \leq i \leq \lfloor \log_2 n \rfloor \text{ et } b_i=1} a^{2^i}$

Fibonacci par l'exponentiation rapide

Analyse de complexite :

- multiplication de matrices 2×2 en temps $O(1)$
- exponentiation rapide en temps $O(\log n)$
- Total : $O(\log n)$

Rappel de l'exponentiation rapide : calcul de a^n

- écriture binaire de $n = \sum_{0 \leq i \leq \lfloor \log_2 n \rfloor} b_i 2^i$
- calcul des a^{2^i} grace a $a^{2^{i+1}} = a^{2^i} * a^{2^i}$
- $a^n = \prod_{0 \leq i \leq \lfloor \log_2 n \rfloor \text{ et } b_i=1} a^{2^i}$
- Total : $O(\log n)$

Comparaison des trois methodes

Complexites temporelles :

pour le calcul de fibo(n)

1. Récursif : $O(1.62^n)$
2. Programmation dynamique : $O(n)$
3. Exponentiation rapide : $O(\log n)$

Comparaison des trois methodes

Complexites temporelles :

1. Récursif : $O(1.62^n)$ $n = 575$
2. Programmation dynamique : $O(n)$
3. Exponentiation rapide : $O(\log n)$

$\log_m \rightarrow 10^{11} \rightarrow \boxed{ms}$

Application numerique :

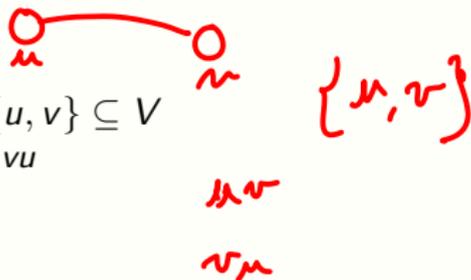
Tous les atomes de l'univers calculent pour vous,
Comme une supercalculateur, depuis le début de l'univers

nb d'atomes	supercalculateur	nb de s.	total
10^{80}	10^{24}	10^{18}	10^{122}

\rightarrow

Notions de bases dans les graphes (non orientés)

- Notation $G = (\overline{V}, E)$ avec
 - ▶ V l'ensemble des sommets
 - ▶ E l'ensemble des arêtes, paires $\{u, v\} \subseteq V$
 - ▶ notées indifféremment uv ou vu



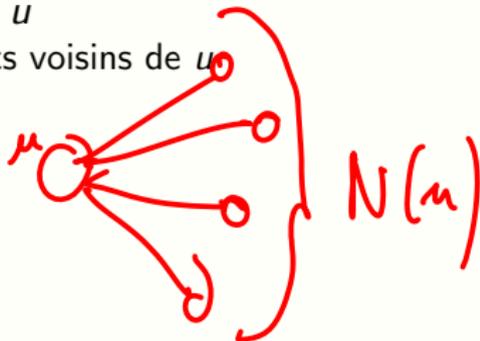
Notions de bases dans les graphes (non orientes)

- Notation $G = (V, E)$ avec
 - ▶ V l'ensemble des sommets
 - ▶ E l'ensemble des aretes, paires $\{u, v\} \subseteq V$
 - ▶ notes indifferemment uv ou vu
- Deux sommets adjacents (ou voisins)
 - ▶ relies par une arete



Notions de bases dans les graphes (non orientes)

- Notation $G = (V, E)$ avec
 - ▶ V l'ensemble des sommets
 - ▶ E l'ensemble des aretes, paires $\{u, v\} \subseteq V$
 - ▶ notes indifferemment uv ou vu
- Deux sommets adjacents (ou voisins)
 - ▶ relie par une arete
- Le voisinage d'un sommet u
 - ▶ L'ensemble des sommets voisins de u



Notions de bases dans les graphes (non orientes)

- Notation $G = (V, E)$ avec
 - ▶ V l'ensemble des sommets
 - ▶ E l'ensemble des aretes, paires $\{u, v\} \subseteq V$
 - ▶ notes indifferemment uv ou vu
- Deux sommets adjacents (ou voisins)
 - ▶ relie par une arete
- Le voisinage d'un sommet u
 - ▶ L'ensemble des sommets voisins de u
- Un chemin
 - ▶ une sequence de sommets $P = x_1x_2 \dots x_k$ tels que
 - ▶ $\forall i \in \llbracket 1, k - 1 \rrbracket, x_i x_{i+1} \in E$

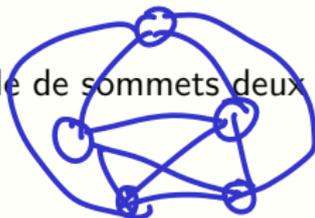


Notions de bases dans les graphes (non orientes)

- Notation $G = (V, E)$ avec
 - ▶ V l'ensemble des sommets
 - ▶ E l'ensemble des aretes, paires $\{u, v\} \subseteq V$
 - ▶ notes indifferemment uv ou vu
- Deux sommets adjacents (ou voisins)
 - ▶ relie par une arete
- Le voisinage d'un sommet u
 - ▶ L'ensemble des sommets voisins de u
- Un chemin
 - ▶ une sequence de sommets $P = x_1x_2 \dots x_k$ tels que
 - ▶ $\forall i \in \llbracket 1, k-1 \rrbracket, x_i x_{i+1} \in E$
- Un cycle
 - ▶ une sequence de sommets $C = x_1x_2 \dots x_k$ tels que
 - ▶ C est un chemin et
 - ▶ $x_k x_1 \in E$

Notions de bases dans les graphes (non orientes)

- Notation $G = (V, E)$ avec
 - ▶ V l'ensemble des sommets
 - ▶ E l'ensemble des aretes, paires $\{u, v\} \subseteq V$
 - ▶ notes indifferemment uv ou vu
- Deux sommets adjacents (ou voisins)
 - ▶ relie par une arete
- Le voisinage d'un sommet u
 - ▶ L'ensemble des sommets voisins de u
- Un chemin
 - ▶ une sequence de sommets $P = x_1x_2 \dots x_k$ tels que
 - ▶ $\forall i \in \llbracket 1, k - 1 \rrbracket, x_i x_{i+1} \in E$
- Un cycle
 - ▶ une sequence de sommets $C = x_1x_2 \dots x_k$ tels que
 - ▶ C est un chemin et
 - ▶ $x_k x_1 \in E$
- Une clique
 - ▶ Un sous-ensemble de sommets deux a deux adjacents



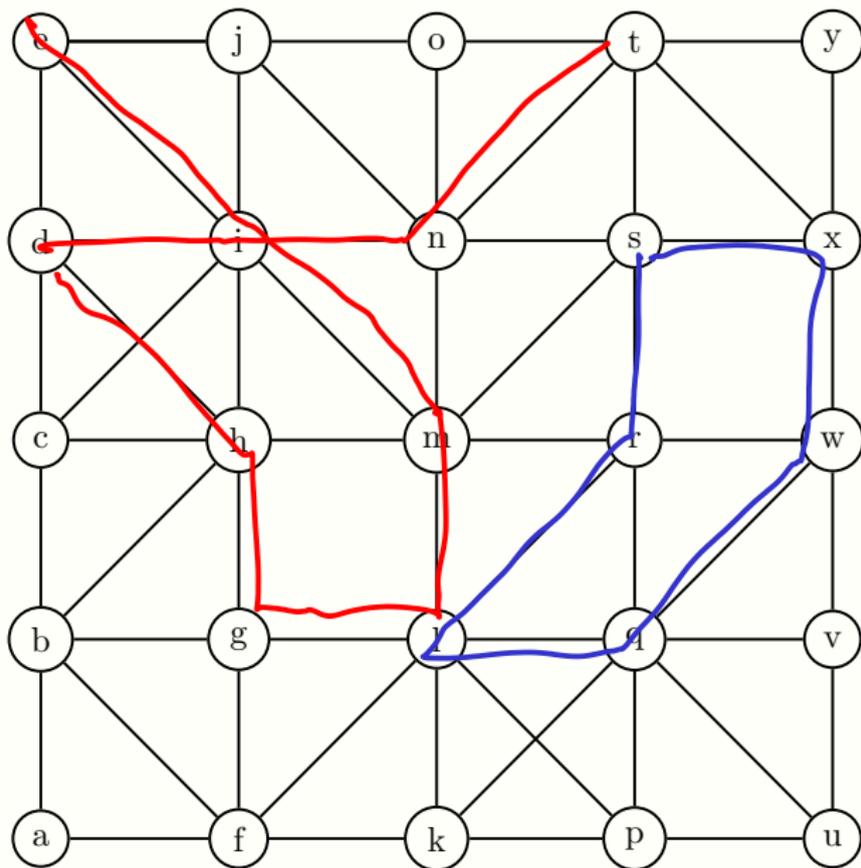
Notions de bases dans les graphes (non orientes)

- Notation $G = (V, E)$ avec
 - ▶ V l'ensemble des sommets
 - ▶ E l'ensemble des aretes, paires $\{u, v\} \subseteq V$
 - ▶ notes indifferemment uv ou vu
- Deux sommets adjacents (ou voisins)
 - ▶ relie par une arete
- Le voisinage d'un sommet u
 - ▶ L'ensemble des sommets voisins de u
- Un chemin
 - ▶ une sequence de sommets $P = x_1x_2 \dots x_k$ tels que
 - ▶ $\forall i \in \llbracket 1, k - 1 \rrbracket, x_i x_{i+1} \in E$
- Un cycle
 - ▶ une sequence de sommets $C = x_1x_2 \dots x_k$ tels que
 - ▶ C est un chemin et
 - ▶ $x_k x_1 \in E$
- Une clique
 - ▶ Un sous-ensemble de sommets deux a deux adjacents
- Un stable
 - ▶ Un sous-ensemble de sommets deux a deux non adjacents

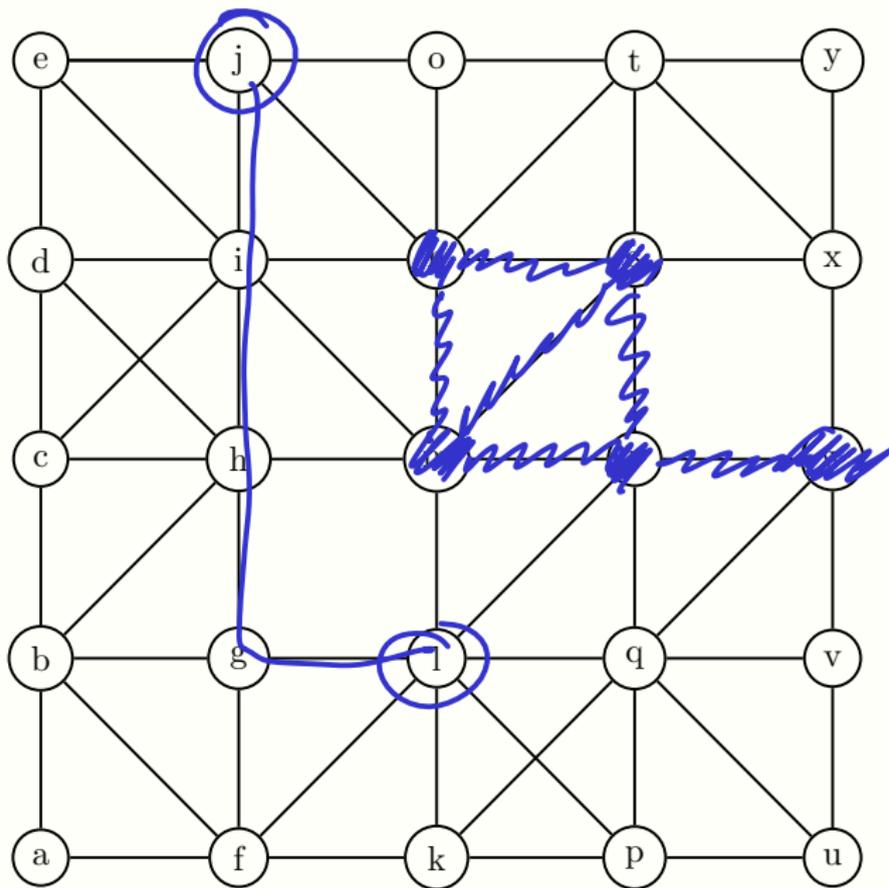


Exemples

e, i, m, l, g, h, d

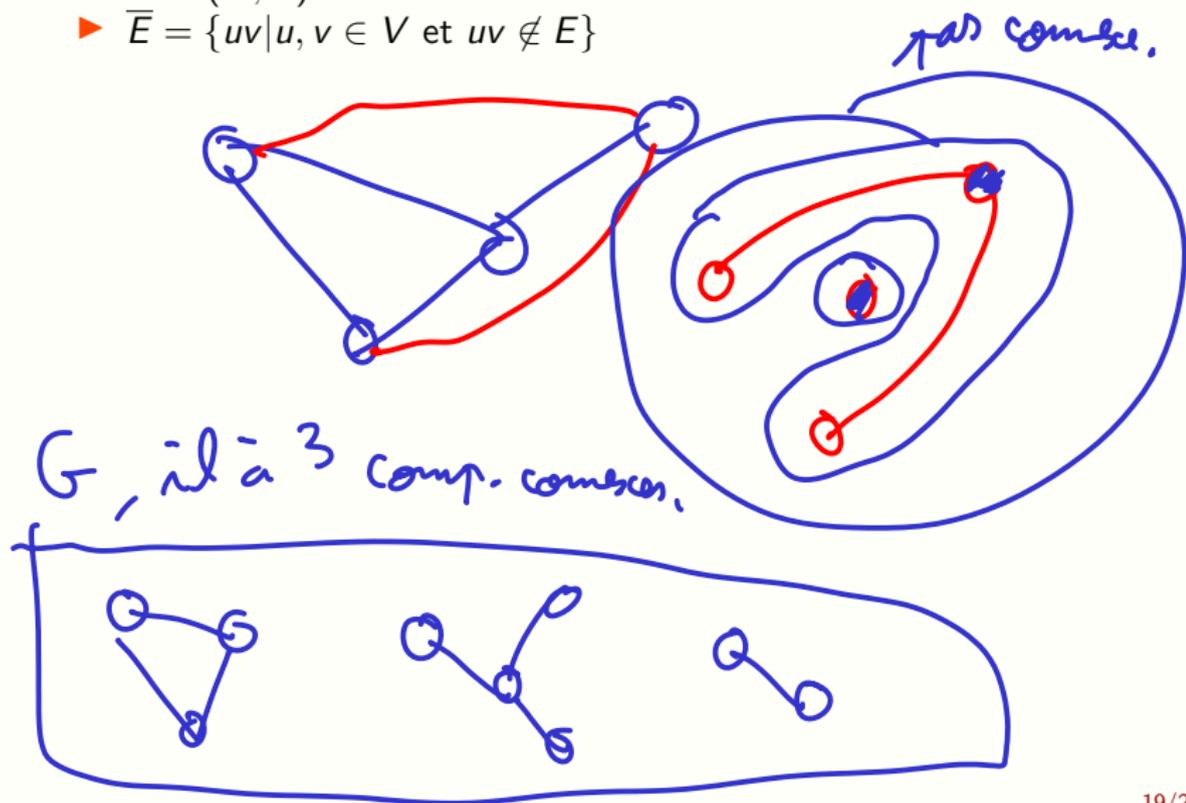


Examples



Autres notions de bases

- Graphe complémentaire
 - ▶ $\bar{G} = (V, \bar{E})$ avec
 - ▶ $\bar{E} = \{uv \mid u, v \in V \text{ et } uv \notin E\}$



Autres notions de bases

- Graphe complémentaire
 - ▶ $\overline{G} = (V, \overline{E})$ avec
 - ▶ $\overline{E} = \{uv \mid u, v \in V \text{ et } uv \notin E\}$
- Graphe induit par un sous ensemble X de sommets
 - ▶ $G[X] = (X, E_X)$ avec
 - ▶ $E_X = \{uv \mid u, v \in X \text{ et } uv \in E\}$

Autres notions de bases

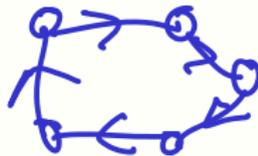
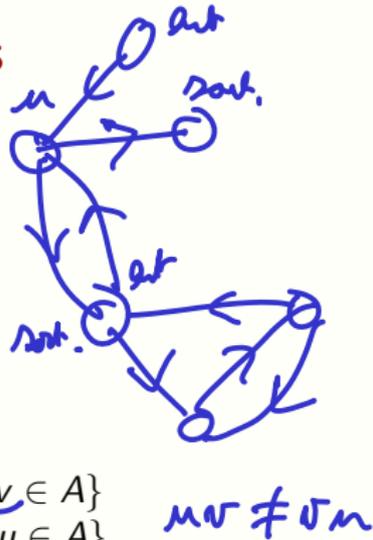
- Graphe complémentaire
 - ▶ $\overline{G} = (V, \overline{E})$ avec
 - ▶ $\overline{E} = \{uv \mid u, v \in V \text{ et } uv \notin E\}$
- Graphe induit par un sous ensemble X de sommets
 - ▶ $G[X] = (X, E_X)$ avec
 - ▶ $E_X = \{uv \mid u, v \in X \text{ et } uv \in E\}$
- Graphe connexe
 - ▶ Un graphe $G = (V, E)$ tq $\forall u, v \in V, \exists$ un chemin de u a v

Autres notions de bases

- Graphe complémentaire
 - ▶ $\overline{G} = (V, \overline{E})$ avec
 - ▶ $\overline{E} = \{uv \mid u, v \in V \text{ et } uv \notin E\}$
- Graphe induit par un sous ensemble X de sommets
 - ▶ $G[X] = (X, E_X)$ avec
 - ▶ $E_X = \{uv \mid u, v \in X \text{ et } uv \in E\}$
- Graphe connexe
 - ▶ Un graphe $G = (V, E)$ tq $\forall u, v \in V, \exists$ un chemin de u a v
- Composante connexe
 - ▶ Un sous-ensemble de sommets X tq. $G[X]$ est connexe et
 - ▶ X est maximal pour l'inclusion (parmi les sous-ensembles ayant cette propriété)

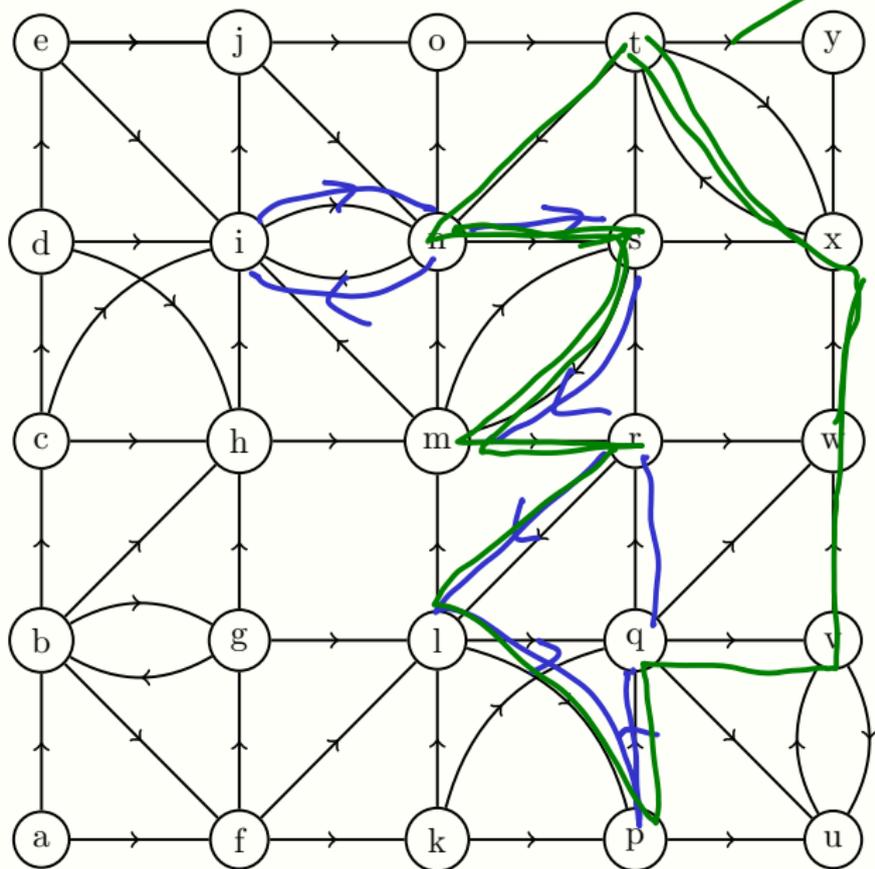
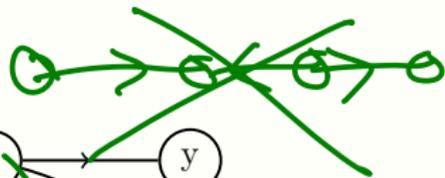
Notions de bases dans les graphes orientés

- Notation $G = (V, A)$ avec
 - ▶ V l'ensemble des sommets
 - ▶ A l'ensemble des arcs, couples $(u, v) \in V^2$
 - ▶ Attention! Dans ce contexte, $uv \neq vu$.
- Sommets adjacents si $uv \in A$ ou $vu \in A$
 - ▶ v est un voisin sortant de u si $uv \in A$
 - ▶ v est un voisin entrant de u si $vu \in A$
- Voisinage d'un sommet u
 - ▶ Voisinage sortant, note $N^+(u) : \{v \in V \mid uv \in A\}$
 - ▶ Voisinage entrant, note $N^-(u) : \{v \in V \mid vu \in A\}$
- Un chemin orienté
 - ▶ une séquence de sommets $P = x_1 x_2 \dots x_k$ tels que
 - ▶ $\forall i \in \llbracket 1, k-1 \rrbracket, x_i x_{i+1} \in A$
- Un circuit
 - ▶ une séquence de sommets $C = x_1 x_2 \dots x_k$ tels que
 - ▶ C est un chemin orienté et
 - ▶ $x_k x_1 \in A$
- (Une clique)
- Un stable
 - ▶ Un sous-ensemble de sommets deux à deux non adjacents



Exemples orientes

m, n, m



Autres notions de bases dans les graphes orientés

- Graphe complémentaire
 - ▶ $\bar{G} = (V, \bar{A})$ avec
 - ▶ $\bar{A} = \{uv \mid u, v \in V \text{ et } uv \notin A\}$

Autres notions de bases dans les graphes orientés

- Graphe complémentaire
 - ▶ $\bar{G} = (V, \bar{A})$ avec
 - ▶ $\bar{A} = \{uv \mid u, v \in V \text{ et } uv \notin A\}$
- Graphe induit par un sous ensemble X de sommets
 - ▶ $G[X] = (X, A_X)$ avec
 - ▶ $A_X = \{uv \mid u, v \in X \text{ et } uv \in A\}$

Autres notions de bases dans les graphes orientés

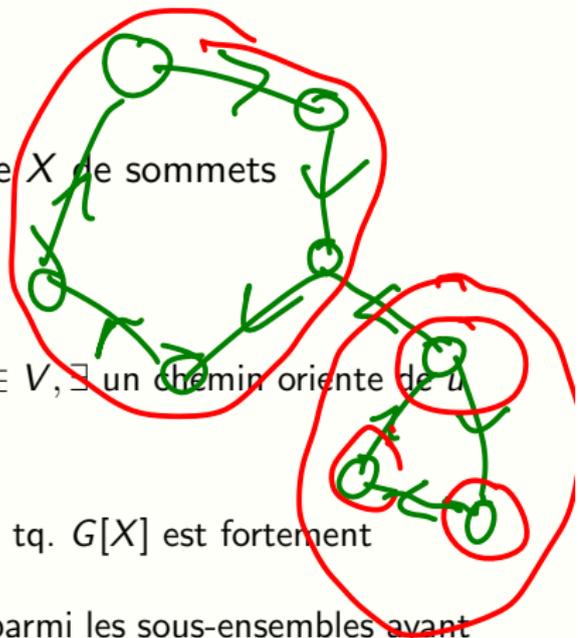
- Graphe complémentaire
 - ▶ $\bar{G} = (V, \bar{A})$ avec
 - ▶ $\bar{A} = \{uv \mid u, v \in V \text{ et } uv \notin A\}$
- Graphe induit par un sous ensemble X de sommets
 - ▶ $G[X] = (X, A_X)$ avec
 - ▶ $A_X = \{uv \mid u, v \in X \text{ et } uv \in A\}$
- Graphe fortement connexe
 - ▶ Un graphe $G = (V, A)$ tq $\forall u, v \in V, \exists$ un chemin orienté de u à v

Autres notions de bases dans les graphes orientes

- Graphe complémentaire
 - ▶ $\bar{G} = (V, \bar{A})$ avec
 - ▶ $\bar{A} = \{uv \mid u, v \in V \text{ et } uv \notin A\}$
- Graphe induit par un sous ensemble X de sommets
 - ▶ $G[X] = (X, A_X)$ avec
 - ▶ $A_X = \{uv \mid u, v \in X \text{ et } uv \in A\}$
- Graphe fortement connexe
 - ▶ Un graphe $G = (V, A)$ tq $\forall u, v \in V, \exists$ un chemin oriente de u a v
- Composante fortement connexe
 - ▶ Un sous-ensemble de sommets X tq. $G[X]$ est fortement connexe et
 - ▶ X est maximal pour l'inclusion (parmi les sous-ensembles ayant cette propriete)

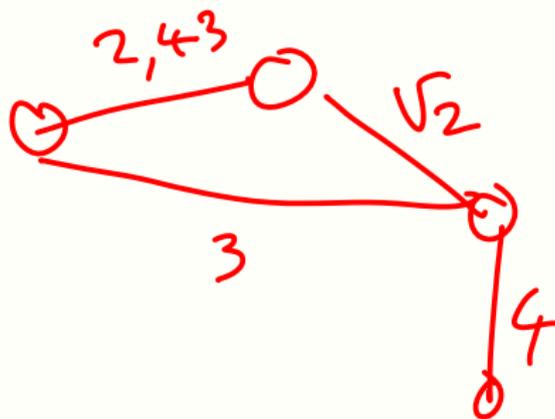
Autres notions de bases dans les graphes orientes

- Graphe complémentaire
 - ▶ $\bar{G} = (V, \bar{A})$ avec
 - ▶ $\bar{A} = \{uv \mid u, v \in V \text{ et } uv \notin A\}$
- Graphe induit par un sous ensemble X de sommets
 - ▶ $G[X] = (X, A_X)$ avec
 - ▶ $A_X = \{uv \mid u, v \in X \text{ et } uv \in A\}$
- Graphe fortement connexe
 - ▶ Un graphe $G = (V, A)$ tq $\forall u, v \in V, \exists$ un chemin oriente de u a v
- Composante fortement connexe
 - ▶ Un sous-ensemble de sommets X tq. $G[X]$ est fortement connexe et
 - ▶ X est maximal pour l'inclusion (parmi les sous-ensembles ayant cette propriete)
- Graphe non oriente sous-jacent
 - ▶ $\tilde{G} = (V, E)$ avec
 - ▶ $E = \{\{u, v\} \subseteq V \mid uv \in A \text{ ou } vu \in A\}$



Quelques variations tres utiles des graphes

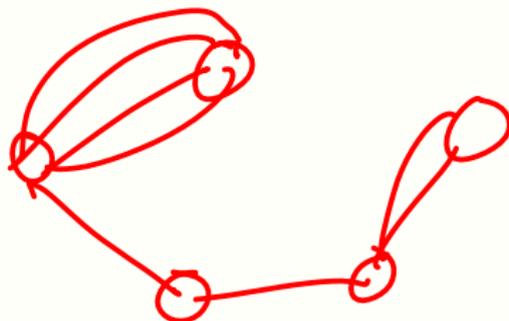
- Graphes ponderes (ou values)
 - ▶ poids sur les aretes : $uv \rightarrow w(uv)$



Quelques variations tres utiles des graphes

- Graphes ponderes (ou values)
 - ▶ poids sur les aretes : $uv \rightarrow w(uv)$

- Multigraphes
 - ▶ plusieurs aretes entre u et v



Quelques variations tres utiles des graphes

- Graphes ponderes (ou values)
 - ▶ poids sur les aretes : $uv \rightarrow w(uv)$

- Multigraphes
 - ▶ plusieurs aretes entre u et v

- Boucles
 - ▶ une arete entre u et lui meme

