M1 Info - Graphes et programmation dynamique

Cours 1 - Introduction

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

Christophe Crespelle
christophe.crespelle@univ-cotedazur.fr

UNIVERSITÉ CÔTE D'AZUR

DIGITAL SYSTEMS FOR HUMANS
GRADUATE SCHOOL AND RESEARCH

Graphes et programmation dynamique

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

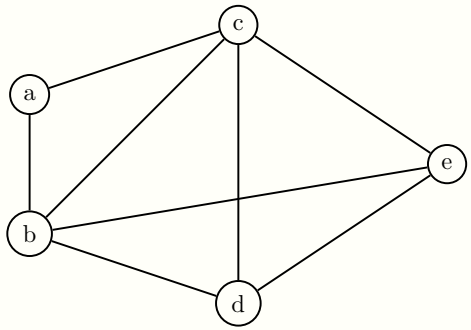


FIGURE – Un graphe a 5 sommets et 8 aretes.

Graphes et programmation dynamique

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

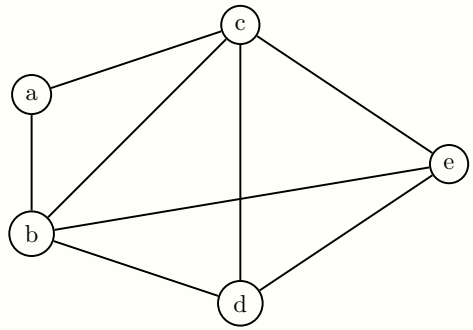


FIGURE – Un graphe a 5 sommets et 8 aretes.

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

Graphes et programmation dynamique

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

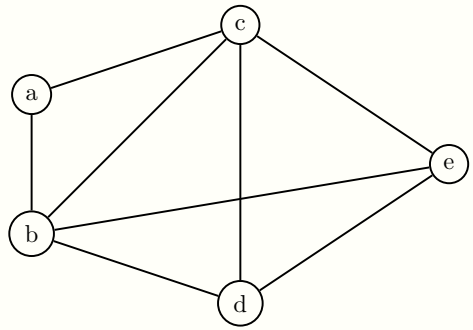


FIGURE – Un graphe a 5 sommets et 8 aretes.

  - **Programmation dynamique :** une approche algorithmique (parmi d'autres)
- A quoi ca sert ?
  - beaucoup de problemes rencontres en pratique sont formules 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

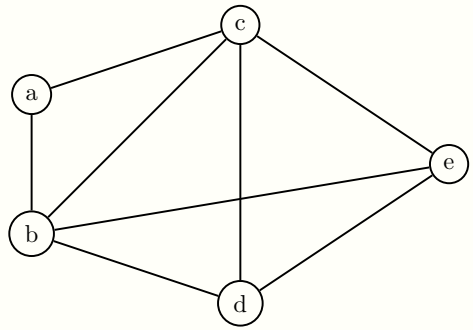


FIGURE – Un graphe a 5 sommets et 8 aretes.

  - **Programmation dynamique :** une approche algorithmique (parmi d'autres)
- A quoi ca sert ?
  - beaucoup de problemes rencontres en pratique sont formules 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)

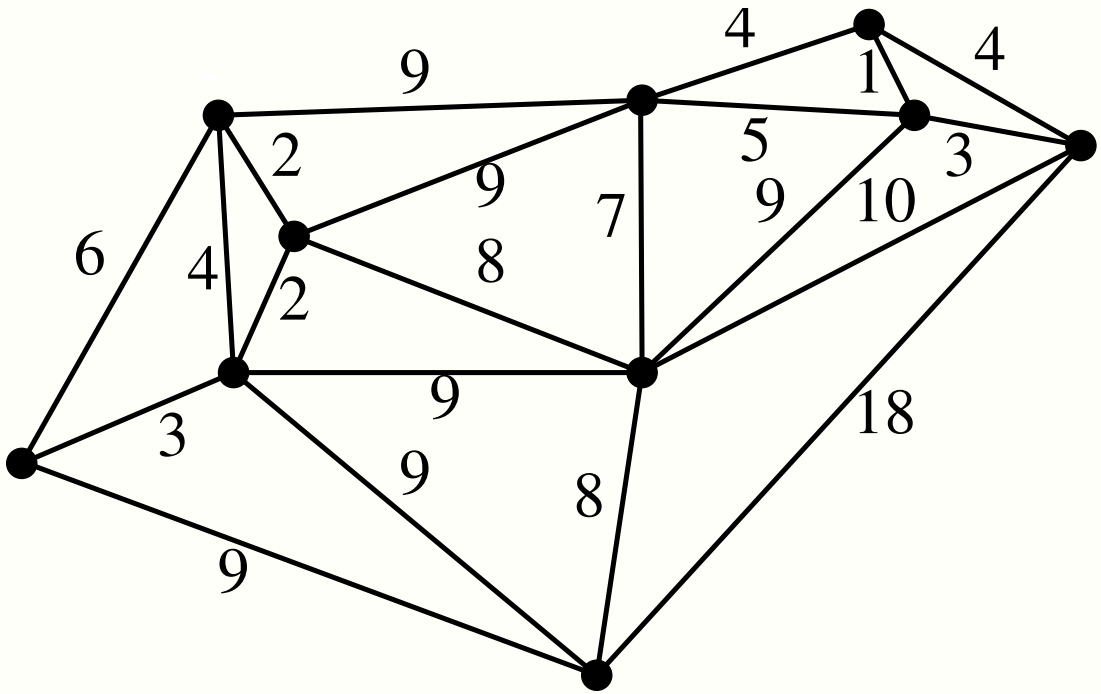


FIGURE – Un probleme d'implantation de ligne electrique.

- **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)

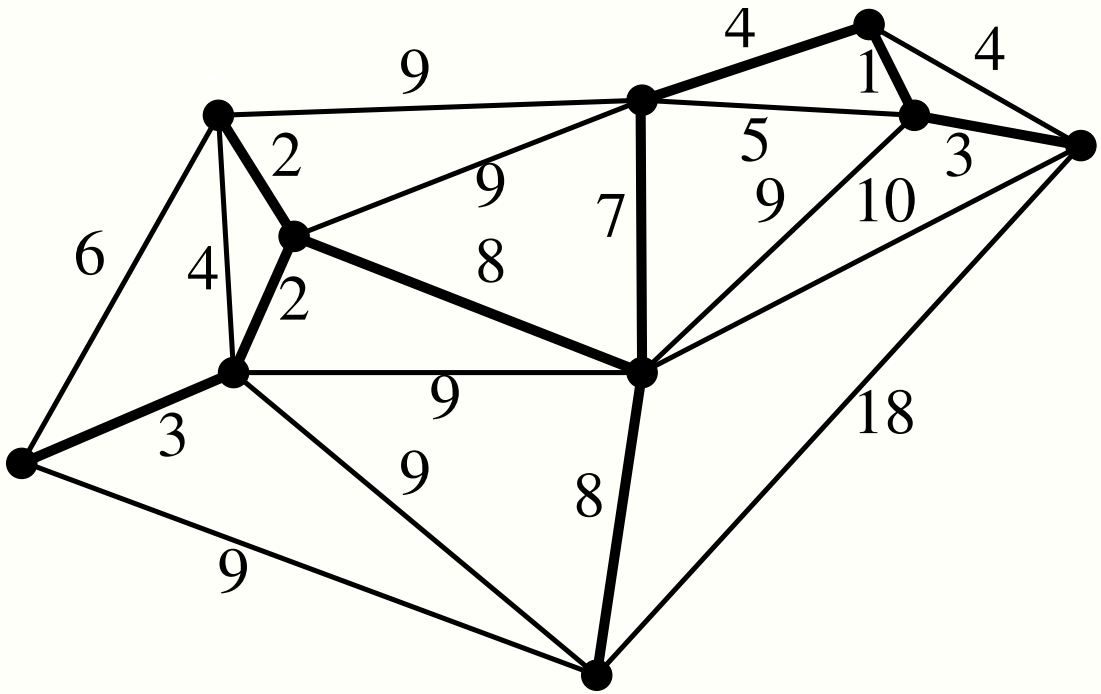


FIGURE – Un probleme d'implantation de ligne electrique.

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

Exemple 2 : equilibrage de charges

- **Donnee :** un ensemble de taches $T_1, T_2, \ldots, T_n$, 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 intermediare : 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 ?

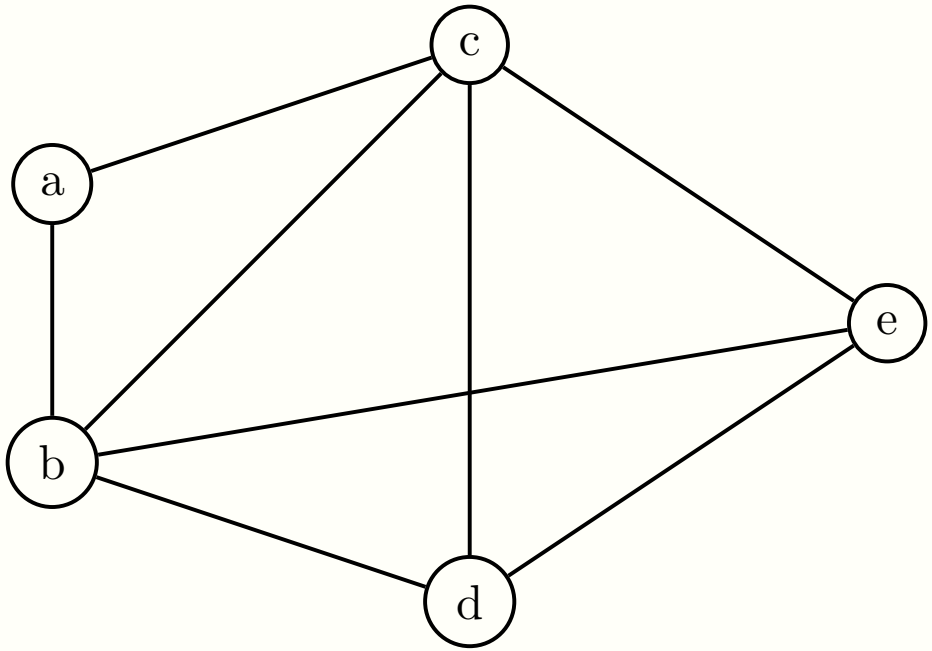
- Apprendre les CM par coeur
- Finir les 2 TP
- Savoir faire les TD (corrections integrales fournies)

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

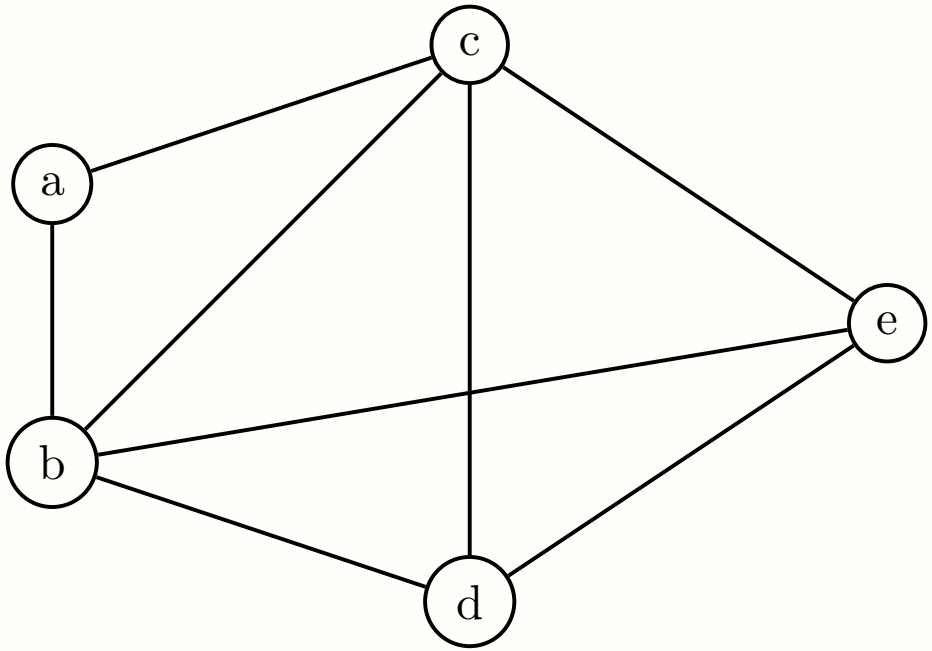
- Un objet simple



- Un ensemble de sommets : $\{a, b, c, d, e\}$
- Un ensemble d'aretes (= 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'aretes (= 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**

$\Longrightarrow$ **Reseaux complexes** ($=$ graphes)

Des graphes partout !

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

$$\Longrightarrow \textbf{Reseaux complexes } (= \text{graphes})$$

- 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 language, bon code,...)
  - Machines puissantes

$$\Longrightarrow \textbf{Ca suffit !}$$

Algorithmique

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

$\Longrightarrow$ **Ca suffit !**

Vraiment ? ? ?

Algorithmique

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

$$\Longrightarrow \textbf{Ca suffit !}$$

Vraiment ? ? ?

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

Fibonacci en recursif

Algorithme 1 : implementation recursive

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

Fibonacci en recursif

Algorithme 1 : implementation recursive

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

Analyse de complexite :

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

Fibonacci en recursif

Algorithme 1 : implementation recursive

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

Analyse de complexite :

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

Fibonacci en recursif

Algorithme 1 : implementation recursive

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

Analyse de complexite :

- T_n le temps de calcul de $fibo(n)$
- $T_n = T_{n-1} + T_{n-2} + 1$
- resolution par le polynome caractersitique : $x^2 - x - 1$
  - 2 racines : $\frac{1+\sqrt{5}}{2}$ et $\frac{1-\sqrt{5}}{2}$
  - la plus grande en valeur absolue est $\frac{1+\sqrt{5}}{2}$

Fibonacci en recursif

Algorithme 1 : implementation recursive

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

Analyse de complexite :

- T_n le temps de calcul de $fibo(n)$
- $T_n = T_{n-1} + T_{n-2} + 1$
- resolution par le polynome caractersitique : $x^2 - x - 1$
  - 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 $T_n = \Theta((\frac{1+\sqrt{5}}{2})^n)$

Fibonacci par la programmation dynamique

Algorithme 2 : programmation dynamique

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

Fibonacci par la programmation dynamique

Algorithme 2 : programmation dynamique

```
1 fibo(n)
2 begin
3 |   Reserver un tableau T de taille max{2, n};
4 |   T[0] ← 0 ; T[1] ← 1;
5 |   pour i de 2 a n faire
6 |   |   T[i] ← 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 $n \geq 2$:
$$\begin{bmatrix} fibo(n) \\ fibo(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} fibo(n-1) \\ fibo(n-2) \end{bmatrix}$$

Fibonacci par l'exponentiation rapide

- D'abord on remarque que pour $n \geq 2$:
$$\begin{bmatrix} fibo(n) \\ fibo(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} fibo(n-1) \\ fibo(n-2) \end{bmatrix}$$
- Ainsi, en posant $A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$, on a pour tout $n \geq 2$:
$$\begin{bmatrix} fibo(n) \\ fibo(n-1) \end{bmatrix} = A^{n-1} \cdot \begin{bmatrix} fibo(1) \\ fibo(0) \end{bmatrix}$$

Fibonacci par l'exponentiation rapide

- D'abord on remarque que pour $n \geq 2$:
$$\begin{bmatrix} fibo(n) \\ fibo(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} fibo(n-1) \\ fibo(n-2) \end{bmatrix}$$
- Ainsi, en posant $A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$, on a pour tout $n \geq 2$:
$$\begin{bmatrix} fibo(n) \\ fibo(n-1) \end{bmatrix} = A^{n-1} \cdot \begin{bmatrix} fibo(1) \\ fibo(0) \end{bmatrix}$$

Algorithme 3 : Fibonacci par exponentiation rapide de matrice

```
1 fibo(n)
2 begin
3 |   Calculer [x_{n-1}  y_{n-1}; z_{n-1}  t_{n-1}] = A^{n-1} par l'exponentiation rapide;
4 |   retourner x_{n-1}
5 end
```

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 : $O(\log n)$

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

- ecriture 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

- ecriture 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

- ecriture 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 :

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

Comparaison des trois methodes

Complexites temporelles :

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

Application numerique :

Notions de bases dans les graphes (non orientes)

- Notation $G = (V, E)$ avec
  - V l'ensemble des sommets
  - E l'ensemble des aretes, paires $\{u, v\} \subseteq V$
    - notees indifferemment 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 $\{u, v\} \subseteq V$
    - notees 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 $\{u, v\} \subseteq V$
    - notees indifferemment uv ou vu
- Deux sommets adjacents (ou voisins)
  - relies 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 $\{u, v\} \subseteq V$
    - notees indifferemment uv ou vu
- Deux sommets adjacents (ou voisins)
  - relies par une arete
- Le voisinage d'un sommet u
  - L'ensemble des sommets voisins de u
- Un chemin
  - une sequence de sommets $P = x_1 x_2 \dots x_k$ tels que
  - $\forall i \in [\![1, k-1]\!], x_i x_{i+1} \in E$

Notions de bases dans les graphes (non orientes)

- Notation $G = (V, E)$ avec
  - V l'ensemble des sommets
  - E l'ensemble des aretes, paires $\{u, v\} \subseteq V$
    - notees indifferemment uv ou vu
- Deux sommets adjacents (ou voisins)
  - relies par une arete
- Le voisinage d'un sommet u
  - L'ensemble des sommets voisins de u
- Un chemin
  - une sequence de sommets $P = x_1 x_2 \dots x_k$ tels que
  - $\forall i \in [\![1, k-1]\!], x_i x_{i+1} \in E$
- Un cycle
  - une sequence de sommets $C = x_1 x_2 \dots x_k$ tels que
  - C est un chemin et
  - $x_k x_1 \in E$

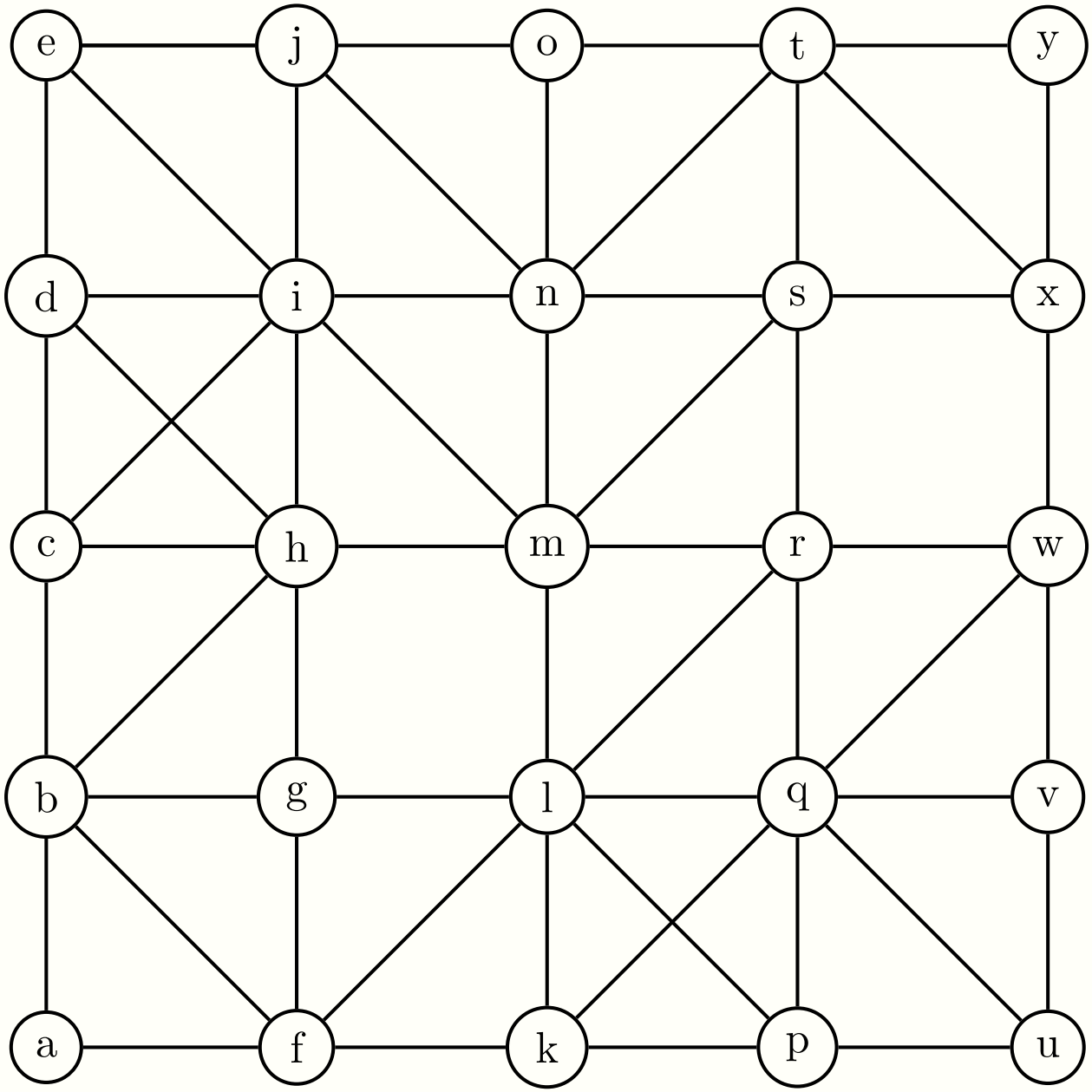
Notions de bases dans les graphes (non orientes)

- Notation $G = (V, E)$ avec
  - V l'ensemble des sommets
  - E l'ensemble des aretes, paires $\{u, v\} \subseteq V$
    - notees indifferemment uv ou vu
- Deux sommets adjacents (ou voisins)
  - relies par une arete
- Le voisinage d'un sommet u
  - L'ensemble des sommets voisins de u
- Un chemin
  - une sequence de sommets $P = x_1 x_2 \dots x_k$ tels que
  - $\forall i \in [\![1, k-1]\!], x_i x_{i+1} \in E$
- Un cycle
  - une sequence de sommets $C = x_1 x_2 \dots x_k$ 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 $\{u, v\} \subseteq V$
    - notees indifferemment uv ou vu
- Deux sommets adjacents (ou voisins)
  - relies par une arete
- Le voisinage d'un sommet u
  - L'ensemble des sommets voisins de u
- Un chemin
  - une sequence de sommets $P = x_1 x_2 \dots x_k$ tels que
  - $\forall i \in [\![1, k-1]\!], x_i x_{i+1} \in E$
- Un cycle
  - une sequence de sommets $C = x_1 x_2 \dots x_k$ 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



Autres notions de bases

- Graphe complementaire
  - $\overline{G} = (V, \overline{E})$ avec
  - $\overline{E} = \{uv | u, v \in V \text{ et } uv \notin E\}$

Autres notions de bases

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

Autres notions de bases

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

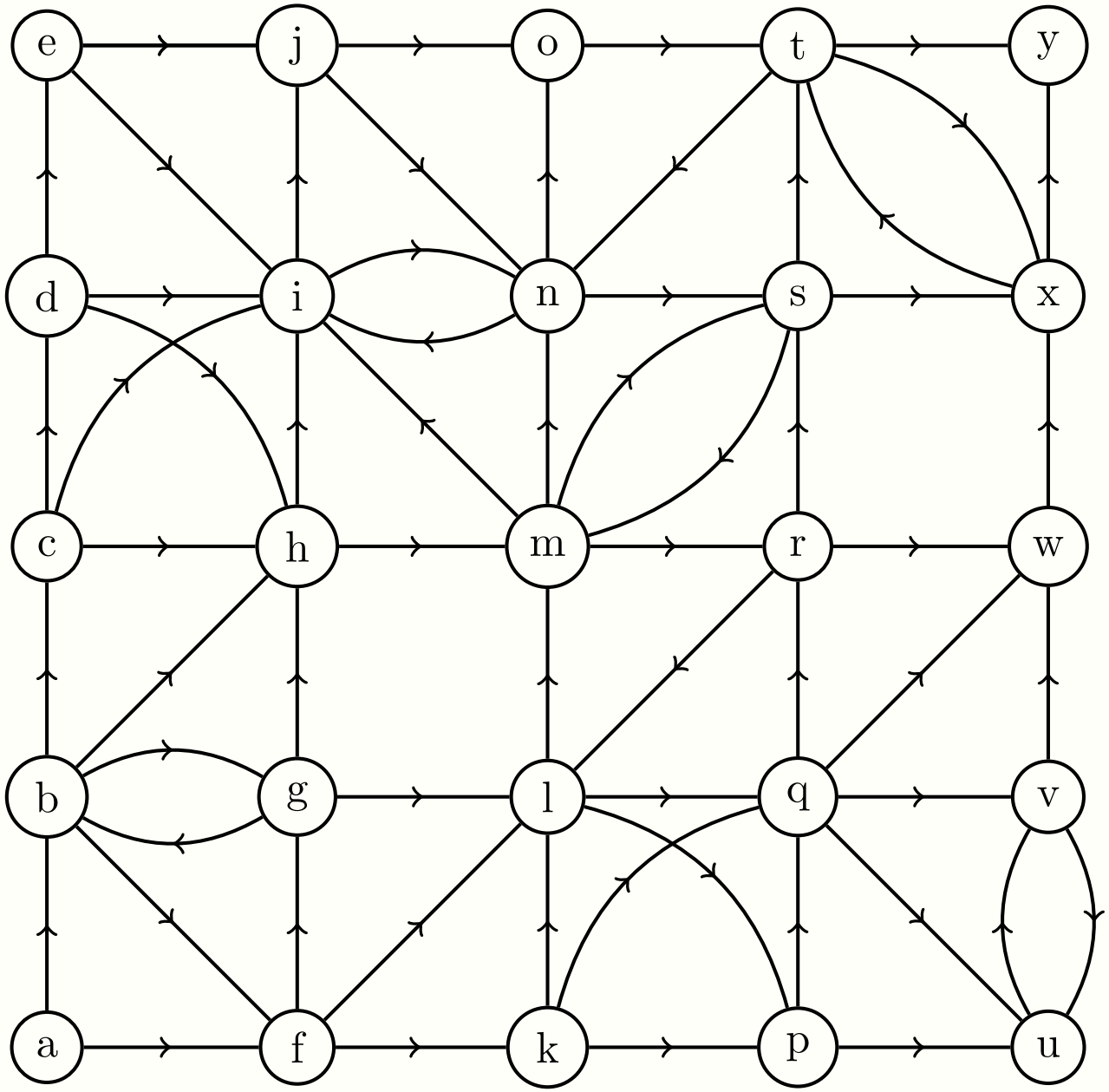
Autres notions de bases

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

Notions de bases dans les graphes orientes

- 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 oriente
  - une sequence de sommets $P = x_1 x_2 \dots x_k$ tels que
  - $\forall i \in [\![1, k-1]\!], x_i x_{i+1} \in A$
- Un circuit
  - une sequence de sommets $C = x_1 x_2 \dots x_k$ tels que
  - C est un chemin oriente et
  - $x_k x_1 \in A$
- (Une clique)
- Un stable
  - Un sous-ensemble de sommets deux a deux non adjacents

Exemples orientes



Autres notions de bases dans les graphes orientes

- Graphe complementaire
  - $\overline{G} = (V, \overline{A})$ avec
  - $\overline{A} = \{uv | u, v \in V \text{ et } uv \notin A\}$

Autres notions de bases dans les graphes orientes

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

Autres notions de bases dans les graphes orientes

- Graphe complementaire
  - $\overline{G} = (V, \overline{A})$ avec
  - $\overline{A} = \{uv | u, v \in V \text{ et } uv \notin A\}$
- Graphe induit par un sous ensemble X de sommets
  - $G[X] = (X, A_X)$ avec
  - $A_X = \{uv | u, v \in X \text{ et } uv \in A\}$
- Graphe fortement connexe
  - Un graphe $G = (V, A)$ tq $\forall u, v \in V, \exists$ un chemin oriente de u a v

Autres notions de bases dans les graphes orientes

- Graphe complementaire
  - $\overline{G} = (V, \overline{A})$ avec
  - $\overline{A} = \{uv | u, v \in V \text{ et } uv \notin A\}$
- Graphe induit par un sous ensemble X de sommets
  - $G[X] = (X, A_X)$ avec
  - $A_X = \{uv | u, v \in X \text{ et } uv \in A\}$
- 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 complementaire
  - $\overline{G} = (V, \overline{A})$ avec
  - $\overline{A} = \{uv | u, v \in V \text{ et } uv \notin A\}$
- Graphe induit par un sous ensemble X de sommets
  - $G[X] = (X, A_X)$ avec
  - $A_X = \{uv | u, v \in X \text{ et } uv \in A\}$
- 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 : $uv \to w(uv)$

Quelques variations tres utiles des graphes

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

- Multigraphes
  - plusieurs aretes entre u et v

Quelques variations tres utiles des graphes

- Graphes ponderes (ou values)
  - poids sur les aretes : $uv \to w(uv)$
- Multigraphes
  - plusieurs aretes entre u et v
- Boucles
  - une arete entre u et lui meme