Thierry Gil
thierry.gil@lecnam.net

Graphes et optimisation RCP101

Sommaire

- Introduction
- Historique
- Qu’est ce qu’un graphe
- Vocabulaire et concepts de base: Sommets, Arcs/Arêtes, Chemins/Chaînes, Cycles/Circuits, Connexité, Forte Connexité.
- Ordonnancement de projets: méthode MPM
- Fermeture transitive d’un graphe: Algorithme de Roy-Warshall

Pourquoi les graphes ?

Sommaire

- Plus courts chemins
  - Ford-Bellman, Dijkstra
- Flots maximaux: Ford-Fulkerson
- Arbre couvrant de poids extremal
- Méthode de parcours des graphes:
  - Parcours en largeur
  - Parcours en profondeur
- Programmation linéaire, simplexe

Les problèmes combinatoires

- Un problème combinatoire est un problème dans lequel il y a un grand nombre de combinaisons possibles.
  - Définir un emploi du temps dans une classe,
  - Organiser les tournées d'une équipe de livreurs,
  - Planifier un chantier,
  - Résoudre un jeu de Sudoku.
- Ce sont des problèmes pour lesquels une solution de bon sens n'est pas aisé à trouver.

Exemple: Combien faut il de repas à une famille de 8 personnes pour épuiser les diverses possibilités de se grouper autour de la table familiale ?

Solution: $8! = 40320$ repas

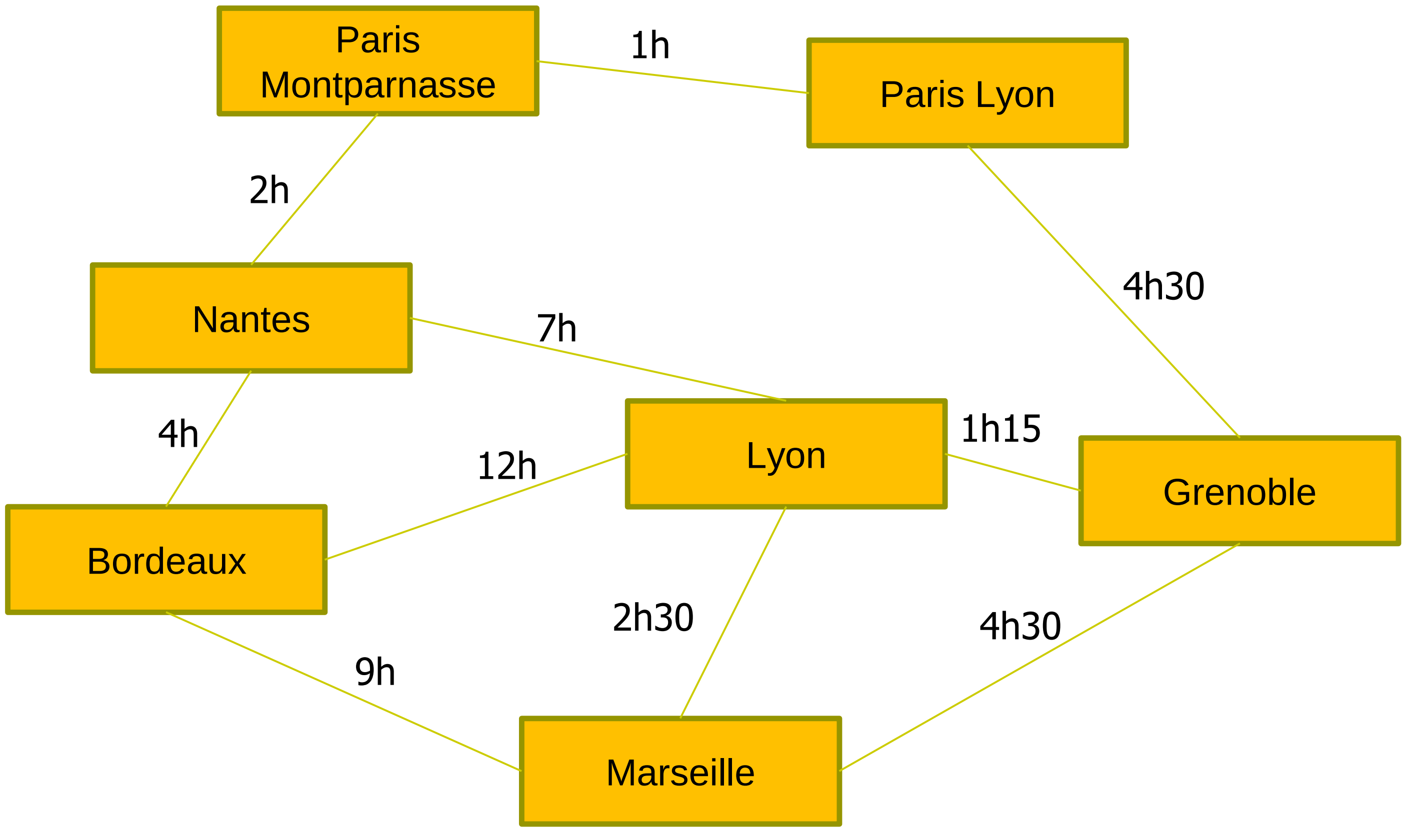
Les problèmes combinatoires

- Il est inconcevable de chercher les 40320 possibilités.
- Un ordinateur pourrait chercher les possibilités à notre place ?
  - Oui mais l’énumération de 20! solutions à raison d’un milliard d’affectations par seconde prendrait 77 ans.
- Les compagnies aériennes sont confrontées à de tels problèmes mais à la dimension 100!
- Ces problèmes font partie de ce que l’on appelle la recherche opérationnelle.
- La théorie des graphes est l’un des instruments les plus efficaces pour résoudre de nombreux problèmes de recherche opérationnelle.

Exemple de problèmes formalisables par les graphes

1. Choix d'un itinéraire.

- Les données du problème sont faciles à représenter par un graphe dont les arêtes sont étiquetées par les durées des trajets:



- Il s'agit de déterminer, dans ce graphe, le plus court chemin (ou l'un des plus courts chemins, s'il existe plusieurs solutions) entre Bordeaux et Grenoble.

Exemple de problèmes formalisables par les graphes

1. Choix d'un itinéraire.

- Sachant qu'une manifestation d'étudiants bloque la gare de Poitiers et connaissant la durée des trajets suivants:

Bordeaux – Nantes :	4h
Bordeaux – Marseille:	9h
Bordeaux – Lyon:	12h
Nantes – Paris Montparnasse:	2h
Nantes – Lyon:	7h
Paris Montparnasse – Paris Lyon:	1h (en bus)
Paris Lyon – Grenoble:	4h30
Marseille – Lyon:	2h30
Marseille – Grenoble:	4h30
Lyon – Grenoble:	1h15

Comment faire pour aller le plus rapidement possible de Bordeaux à Grenoble ?

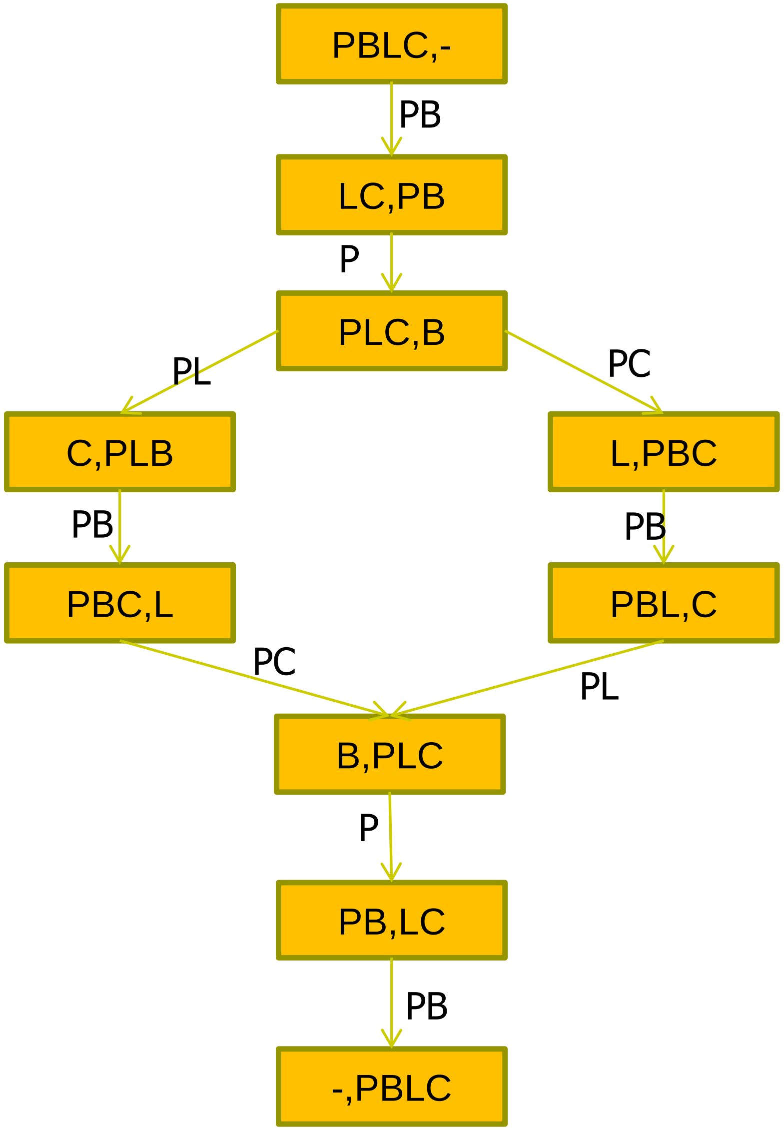
Exemple de problèmes formalisables par les graphes

2. Le loup, le bouc et le chou

- Un passeur se trouve sur le bord d'un fleuve. Il doit faire passer de l'autre côté un loup, un bouc et un chou, mais ne peut transporter qu'un seul client à la fois. Peut-il y arriver sachant qu'il ne peut laisser seuls ensemble ni le loup et le bouc, ni le bouc et le chou. Si oui, comment procéder de la façon la plus rapide ?

Exemple de problèmes formalisables par les graphes

2. Le loup, le bouc et le chou



Le problème peut être formalisé par un graphe dont les sommets sont les configurations acceptables du jeu, et dont les arêtes sont les traversées possibles du passeur.

Exemple de problèmes formalisables par les graphes

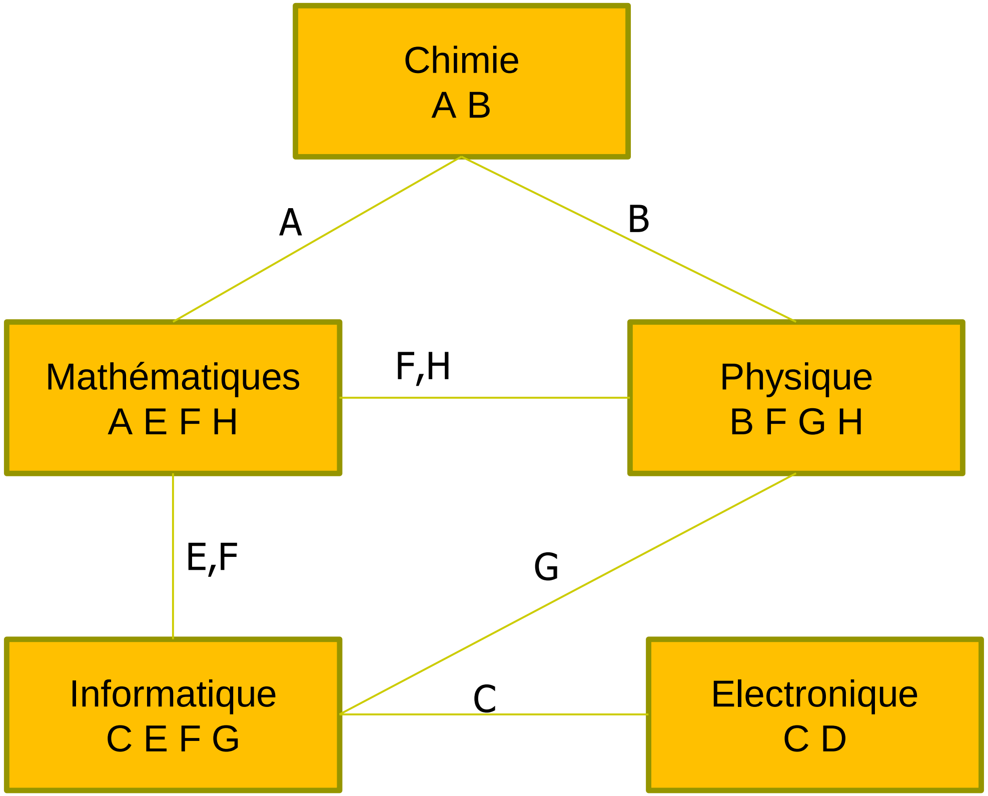
3. Organisation d'une session d'examen

- Des étudiants A, B, C, D, E et F doivent passer des examens dans différentes disciplines, chaque examen occupant une demi-journée:
  - Chimie: étudiants A et B
  - Electronique: étudiants C et D
  - Informatique: étudiants C, E, F et G
  - Mathématiques: étudiants A, E, F et H
  - Physique: étudiants B, F, G et H

- On cherche à organiser la session d'examens la plus courte possible.

Exemple de problèmes formalisables par les graphes

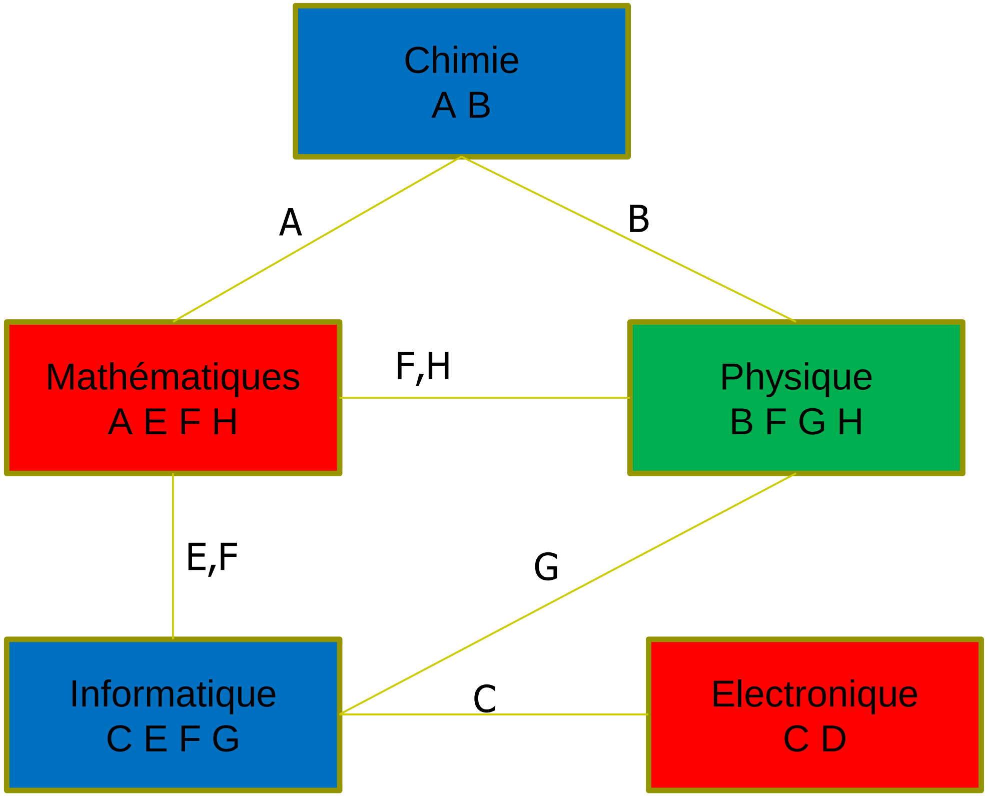
3. Organisation d'une session d'examen



On peut représenter chacune des disciplines par un sommet, et relier par des arêtes les sommets correspondant aux examens incompatibles (ayant des étudiants en commun).

Exemple de problèmes formalisables par les graphes

3. Organisation d'une session d'examen

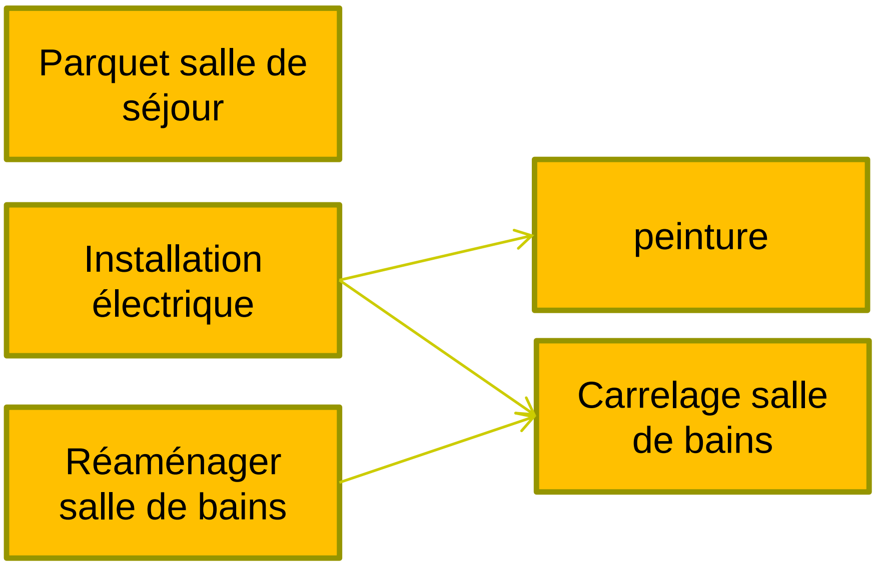


Coloration de graphe:
Il suffit alors de colorier chacun des sommets du graphe en utilisant le moins de couleurs possibles. Les sommets voisins (reliés par une arête) seront nécessairement de couleurs différentes.

Exemple de problèmes formalisables par les graphes

4. Planification de travaux

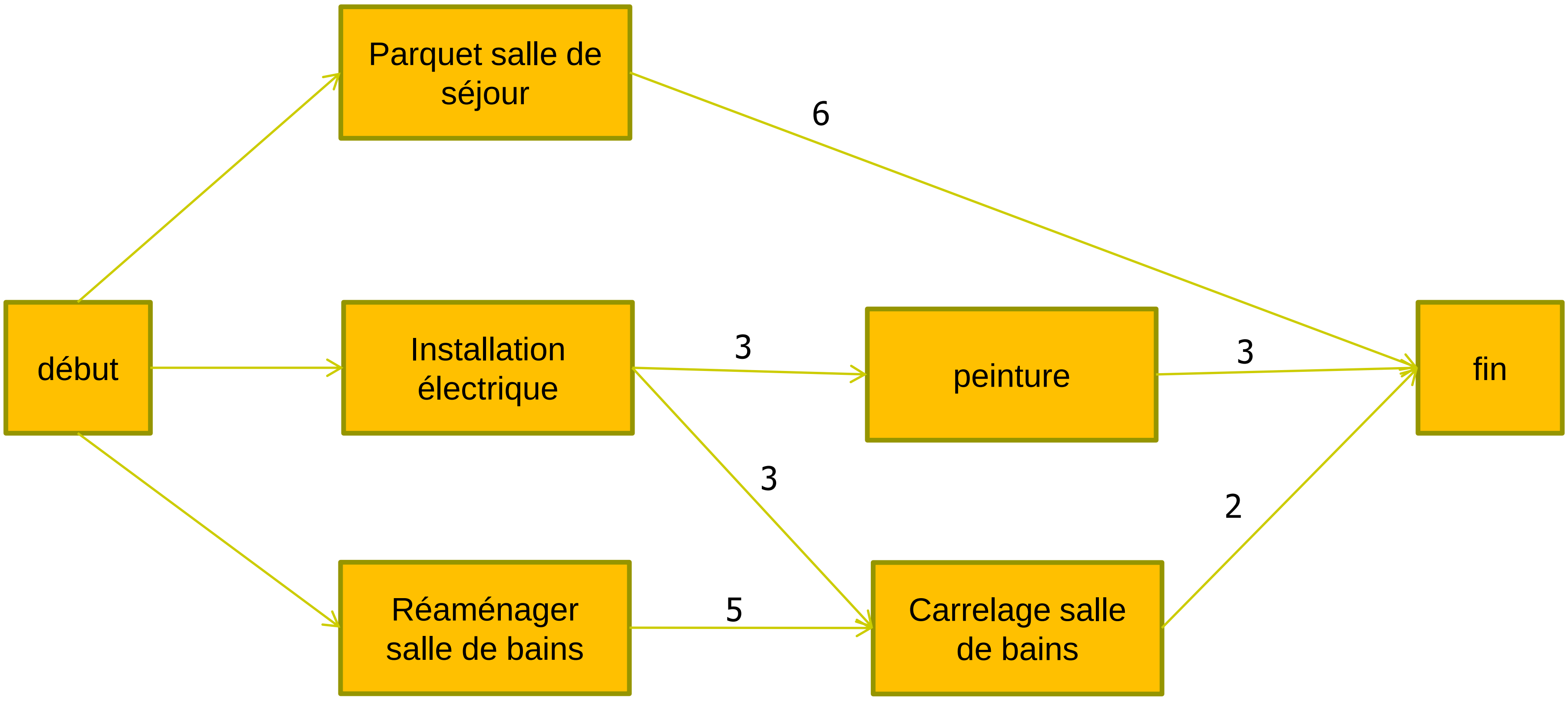
- Pour rénover une maison, il est prévu de refaire l'installation électrique (3 jours), de réaménager la salle de bains (5 jours) et de carreler (2 jours) la salle de bains, de refaire le parquet de la salle de séjour (6 jours) et de repeindre les chambres (3 jours), la peinture et le carrelage ne devant être faits qu'après réfection de l'installation électrique.
- Si le propriétaire décide de tout faire lui-même, dans quel ordre doit-il procéder ?
- Si la rénovation est faite par une entreprise et que chacune des tâches est accomplie par un employé différent, quelle est la durée minimale des travaux ?



On peut représenter les différentes étapes de la rénovation sur un graphe dont les arcs sont étiquetés par la durée minimale séparant deux étapes.

Exemple de problèmes formalisables par les graphes

4. Planification de travaux



Il s’agit de déterminer la durée du plus long chemin du début à la fin des travaux.

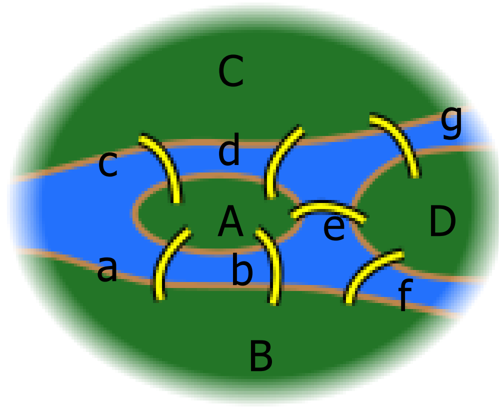
Histoire des graphes

Naissance de la théorie des graphes

- L’origine de la théorie des graphes se situe en 1736, lorsque Leonhard Euler essaya de résoudre le problème des 7 ponts de Königsberg.

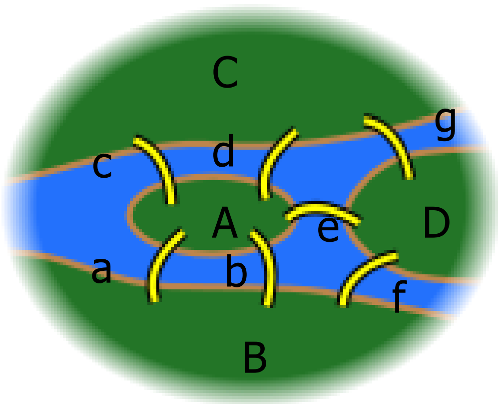
- Le problème était de trouver une route commençant et finissant au même endroit et traversant tous les ponts exactement une fois.

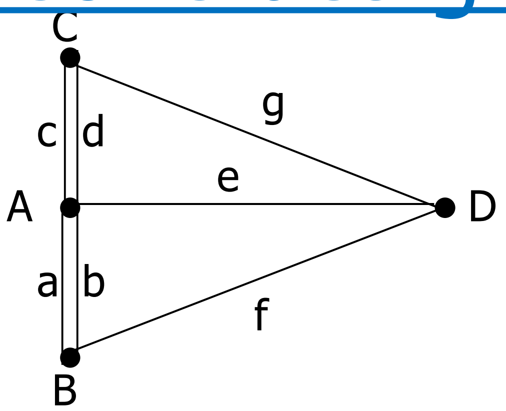
Naissance de la théorie des graphes



- Ce problème peut être modélisé au moyen d’un graphe.
- Les sommets A, B, C et D représentent les berges de la rivière et l’ilot.
- Les arêtes a, b, c, d, e, f et g représentent les sept ponts.
- La route cherchée correspond au parcours des arêtes du graphe d’une façon telle que chaque arête soit parcourue exactement une fois.

Naissance de la théorie des graphes





- Un graphe dans lequel il existe une route commençant et finissant au même sommet et qui emprunte chaque arête exactement une fois est appelé graphe eulérien.

- Est-ce que le graphe des 7 ponts de Konigsberg est un graphe Eulérien ?

Complexité des algorithmes

Complexité des algorithmes

- La complexité d'un algorithme est le nombre d'opérations élémentaires (affectations, comparaisons, opérations mathématiques) effectuées par un algorithme.

Notation	Type de complexité
$O(1)$	Complexité constante
$O(\log_2(n))$	Complexité logarithmique
$O(n)$	Complexité linéaire
$O(n\log_2(n))$	Complexité linéarithmétique
$O(n^2)$	Complexité polynomiale quadratique
$O(n^3)$	Complexité polynomiale cubique
$O(2^n)$	Complexité exponentielle

- Les algorithmes de complexité $O(n)$ ou $O(n\log_2 n)$ sont considérés comme rapides.
- Les algorithmes de complexité $O(n^2)$, $O(n^3)$ ou $O(2^n)$ sont considérés comme lents.
- Un algorithme polynomial (le degré est constant et indépendant de n) est un algorithme efficace.

Complexité d'un algorithme Recherche du plus petit élément

```
int k = 0;
for (int i=1; i < n; i++)
{
    if ( x[i] < x[k] )
    {
        k = i;
    }
}
min = x[k];
```

- Dans cette fonction, on exécute n-1 tests de comparaison. La complexité est donc n-1 = O(n).

Complexité d'un algorithme
Multiplication de matrices

- Soient 2 matrices A et B de dimension (n,n), leur produit P est une matrice de même dimension définie par:

$$P(i,j) = \sum_{k=1}^{k=n} A(i,k) * B(k,j)$$

- Si on écrit l'algorithme:

```
for ( i=1; i <= n; i++)
{
    for ( j=1; j <= n; j++)
    {
        P(i,j)=0;
        for (k=1; k <= n; k++)
        {
            P(i,j) = P(i,j) + A(i,k)*B(k,j);
        }
    }
}
```

- Il y a 3 boucles imbriquées dans lesquelles on passe n fois, on dit que la complexité de l'algorithme de calcul est de l'ordre de n^3.

Complexité d’un algorithme
Dichotomie *(= "couper en deux" en grec)*

- Soit un tableau d’entiers triés en ordre croissant. On veut savoir à quelle position de ce tableau se trouve un élément d’une valeur précise.

```
indice = INDICE_INVALIDE;
plafond = TAILLE-1;
plancher = 0;
trouve = false;
while ( !trouve && plancher <= plafond)
{
    milieu = (plancher + plafond)/2;
    if (val == tab[milieu])
    {
        trouve = true;
        indice = milieu;
    }
    else if (val < tab[milieu] ) plafond = milieu - 1;
    else plancher = milieu + 1;
}
```

- Si le tableau a 10 éléments, on aura besoin au pire de 4 itérations (le tableau débutera à 10 éléments, puis 5, puis 2, puis 1).
- Si le tableau a 100 éléments, on aura besoin au pire de 7 itérations (le tableau débutera à 100 éléments, puis 50, puis 25, 12, 6, 3, 1).
- Si le tableau a 1000 éléments, on aura besoin au pire de 10 itérations (le tableau débutera à 1000 éléments, puis 500, puis 250, 125, 62, 31, 15, 7, 3, 1).

Complexité d'un algorithme
Dichotomie

- Si le tableau a 10 éléments, on aura besoin au pire de 4 itérations (le tableau débutera à 10 éléments, puis 5, puis 2, puis 1).
- Si le tableau a 100 éléments, on aura besoin au pire de 7 itérations (le tableau débutera à 100 éléments, puis 50, puis 25, 12, 6, 3, 1).
- Si le tableau a 1000 éléments, on aura besoin au pire de 10 itérations (le tableau débutera à 1000 éléments, puis 500, puis 250, 125, 62, 31, 15, 7, 3, 1).

n	# itérations	$\log_2(n)$
10	4	$\log_2(10) = 3{,}32$
100	7	$\log_2(100) = 6{,}64$
1000	10	$\log_2(1000) = 9{,}96$
10000	14	$\log_2(10000) = 13{,}28$
100000	17	$\log_2(100000) = 16{,}61$

10000, 5000, 2500,1250, 625, 312, 156, 78, 39, 20, 10, 5, 2, 1 = 14 itérations
100000, 50000, 25000, 12500, 6250, 3125, 1562, 781, 390, 195, 97, 48, 24, 12, 6, 3, 1 = 17 itérations.

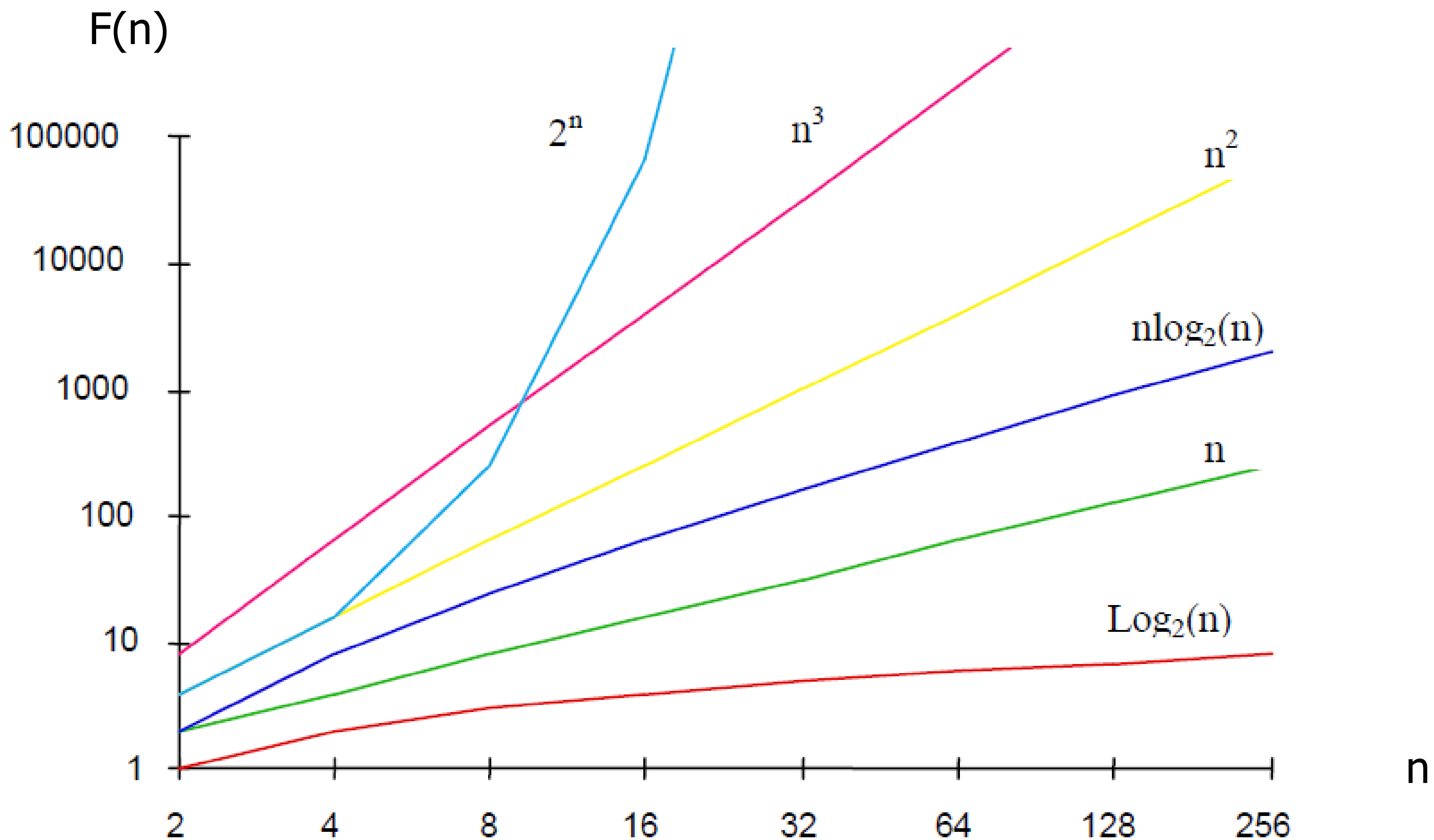
Complexité des algorithmes

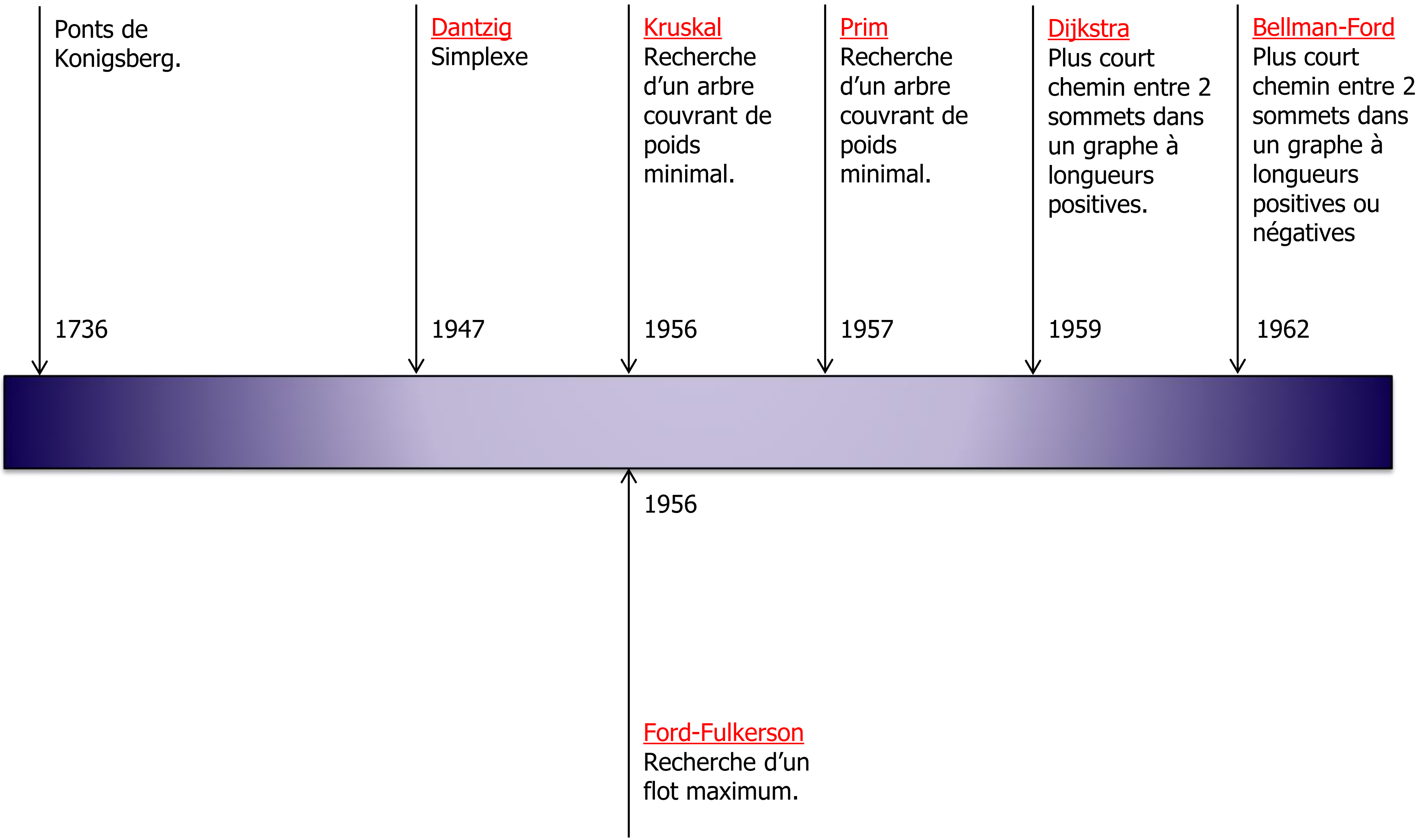
	1	$\log_2(n)$	n	$n\log_2(n)$	n^2	n^3	2^n	n!
n=10	1µs	3µs	10µs	30µs	100µs	1 ms	1,024ms	3,6 ms
n=100	1µs	6µs	100µs	600µs	10ms	1 s	$4\ 10^{16}$ an	$3\ 10^{141}$ an
n=1000	1µs	10µs	1 ms	10 ms	1 s	16 mn	□	□
n=10000	1µs	13µs	10 ms	130 ms	1mn 40s	11,5 jour	□	□
n=100000	1µs	17µs	100 ms	1,7 s	2,7h	31 an	□	□

Complexité 2n: recherche du Rubik’s cube

Complexité n!: recherche du problème du voyageur de commerce.

Complexité des algorithmes




Ponts de Konigsberg.
1736
Dantzig
Simplexe
1947
Kruskal
Recherche d'un arbre couvrant de poids minimal.
1956
Prim
Recherche d'un arbre couvrant de poids minimal.
1957
Dijkstra
Plus court chemin entre 2 sommets dans un graphe à longueurs positives.
1959
Bellman-Ford
Plus court chemin entre 2 sommets dans un graphe à longueurs positives ou négatives
1962
1956
Ford-Fulkerson
Recherche d'un flot maximum.