Histoire d'évasion

PCC - 2

Jean-Marc.Vincent@univ-grenoble-alpes.fr[1]

[1] Laboratoire LIG
Équipe-Projet INRIA POLARIS
Université Grenoble-Alpes

Algorithmique et Modélisation
L3-INFO

ÉVASION

Un problème antique (voire préhistorique)

Mosaïque romaine (-400 à -300 av. JC)
Villa romaine à Loigerfelder près de Salzburg

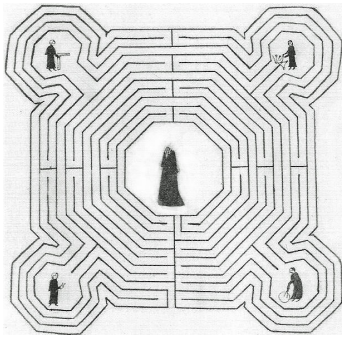
Kunsthistorisches Museum Wikipedia, lien

By Jacques Bailly, after Sébastien Leclerc. lien

Question

Trouver un chemin pour aller combattre le Minotaure.

Ici c'est évident, car il n'y a qu'un seul chemin qui remplit tout l'espace (courbe remplissante), on ne peut se tromper (chemin initiatique)



Par Jacques Cellier, Domaine public, lien

ÉVASION

Un problème antique (voire préhistorique)
Dans les jardins de Versailles (1672)

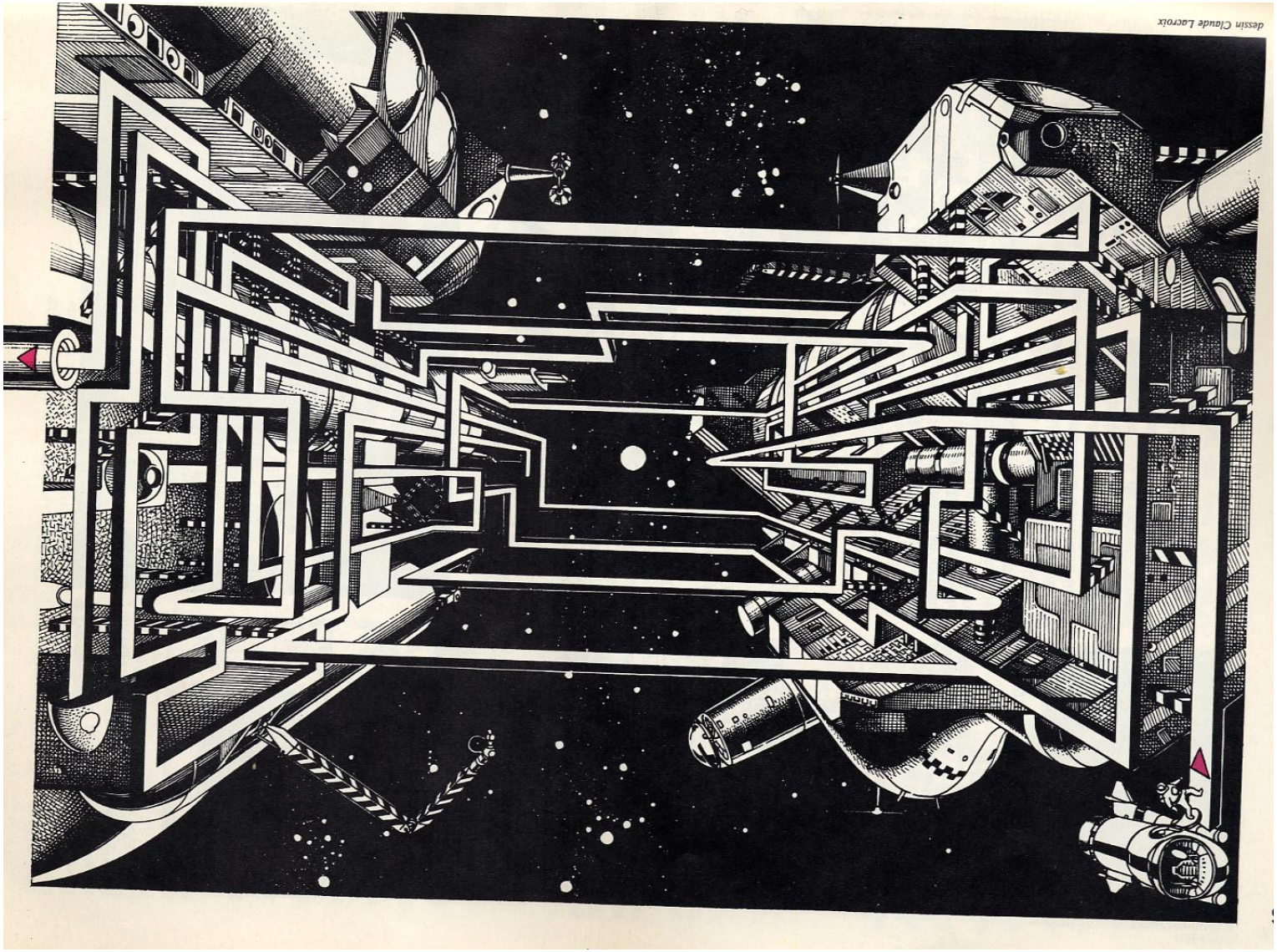
By C. Perrault - Bibliothèque nationale de France, Public Domain, lien

Question

Rendez-vous à la fontaine *Le lièvre et la tortue* (numéro 20)
Trouver un chemin de l'entrée du labyrinthe à toute fontaine à l'intérieur du labyrinthe (faire le plan).

Source Wikipedia The labyrinth of Versailles

ÉVASION (DANS L'ESPACE)



Source : Jeux & Stratégie n 13 (1982)

MODÉLISATION

La difficulté principale de ce problème est sa modélisation, c'est à dire sa représentation sous forme abstraite.

- ce que l'on a à disposition : une entrée de labyrinthe
- des opérations : déplacements dans le labyrinthe (identification des endroits/segments/...)

Problème : une méthode pour construire des chemins pour parcourir le labyrinthe.

On a ainsi un problème de cheminement dans un graphe

- $\mathcal{G} = (\mathcal{X}, \mathcal{A})$ graphe
- $\mathcal{X}$ ensemble de n sommets ($\mathcal{X}$, n ne sont pas forcément connus), positions dans le labyrinthe
- $\mathcal{A}$ un ensemble de m arcs ($\mathcal{A}$, m ne sont pas forcément connus), passage d'une position à une autre.
- Un sommet x_0, entrée du labyrinthe

Les ensembles peuvent être construits à la volée lorsque l'on en a besoin (fonction de voisinage).

UGA Université Grenoble Alpes

FORMALISATION

Ensemble des chemins du graphe $\mathcal{G}$ d'origine x_0 noté $\mathcal{C}_{x_0}$

Questions de même nature sur $\mathcal{C}_{x_0}$

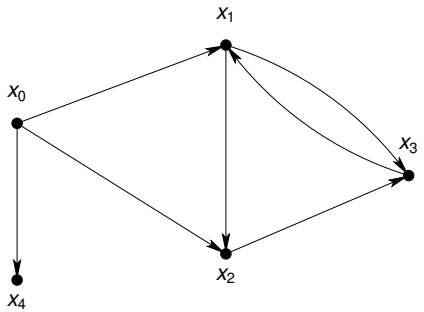
- Existe-t'il un chemin de x_0 à x_k dans $\mathcal{C}_{x_0}$?
- Quelles sont les valeurs de k pour lesquelles il existe un chemin de x_0 à x_k dans $\mathcal{C}_{x_0}$? (accessibilité à partir de x_0)
- Pour tout couple (x_i, x_j) existe-t'il un chemin de x_i à x_j ?
- Quelles sont les composantes fortement connexes du graphe ?
- ...

UGA Université Grenoble Alpes

STRUCTURE DE $\mathcal{C}_{x_0}$

L'objectif est de parcourir tout l'ensemble $\mathcal{C}_{x_0}$

Un petit exemple

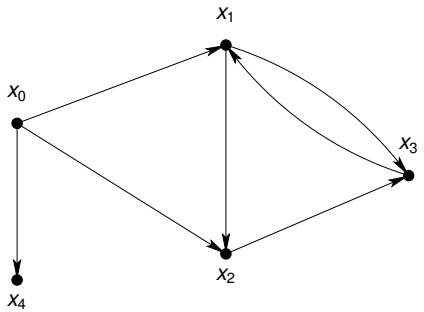


Arbre des chemins

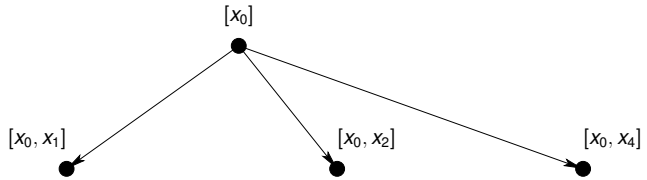
STRUCTURE DE $\mathcal{C}_{x_0}$

L'objectif est de parcourir tout l'ensemble $\mathcal{C}_{x_0}$

Un petit exemple



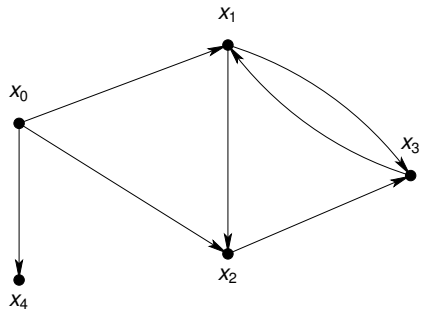
Arbre des chemins



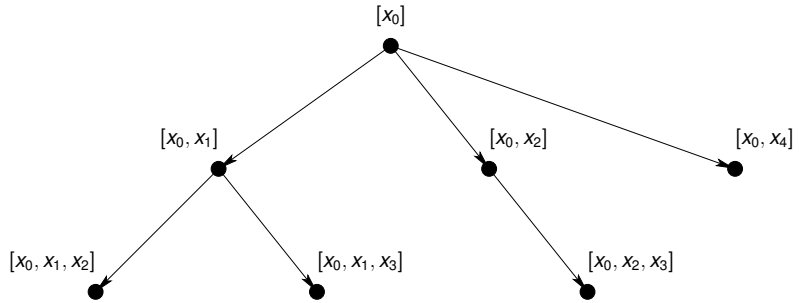
STRUCTURE DE $\mathcal{C}_{x_0}$

L'objectif est de parcourir tout l'ensemble $\mathcal{C}_{x_0}$

Un petit exemple



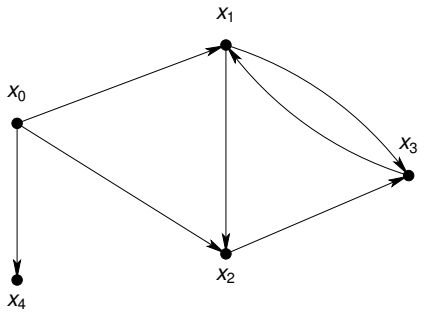
Arbre des chemins



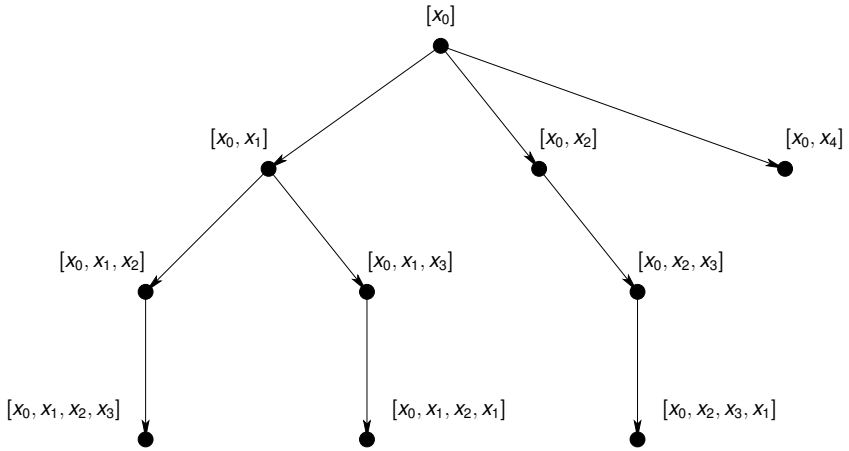
STRUCTURE DE $\mathcal{C}_{x_0}$

L'objectif est de parcourir tout l'ensemble $\mathcal{C}_{x_0}$

Un petit exemple



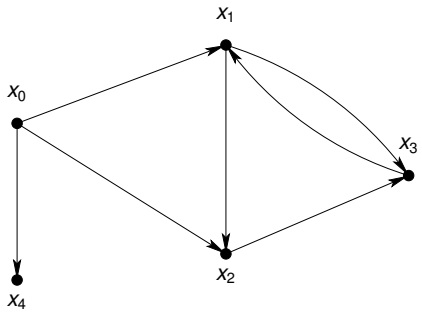
Arbre des chemins



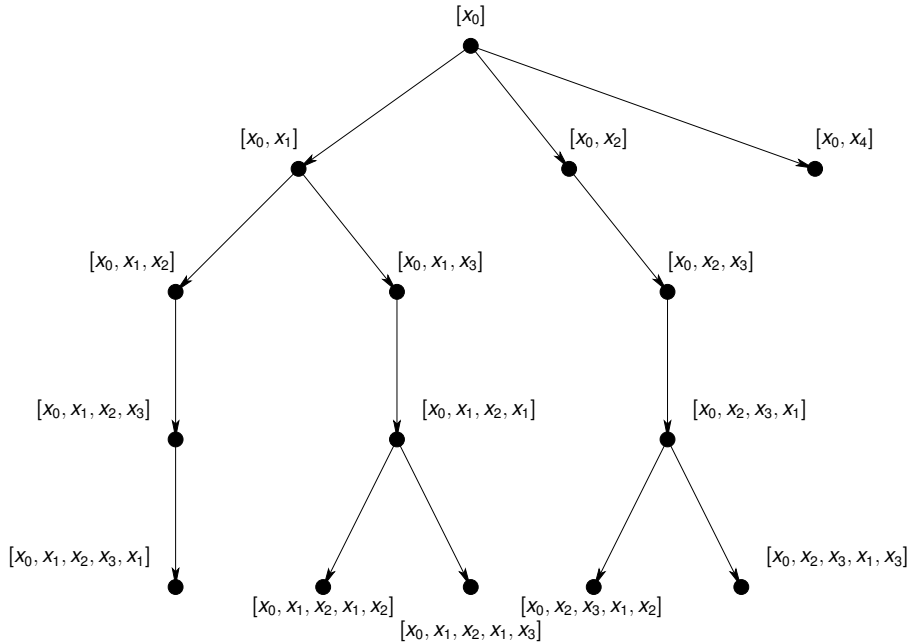
STRUCTURE DE $\mathcal{C}_{x_0}$

L'objectif est de parcourir tout l'ensemble $\mathcal{C}_{x_0}$

Un petit exemple



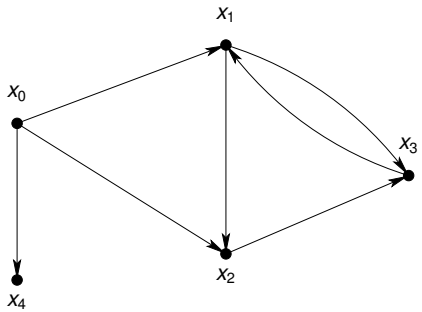
Arbre des chemins



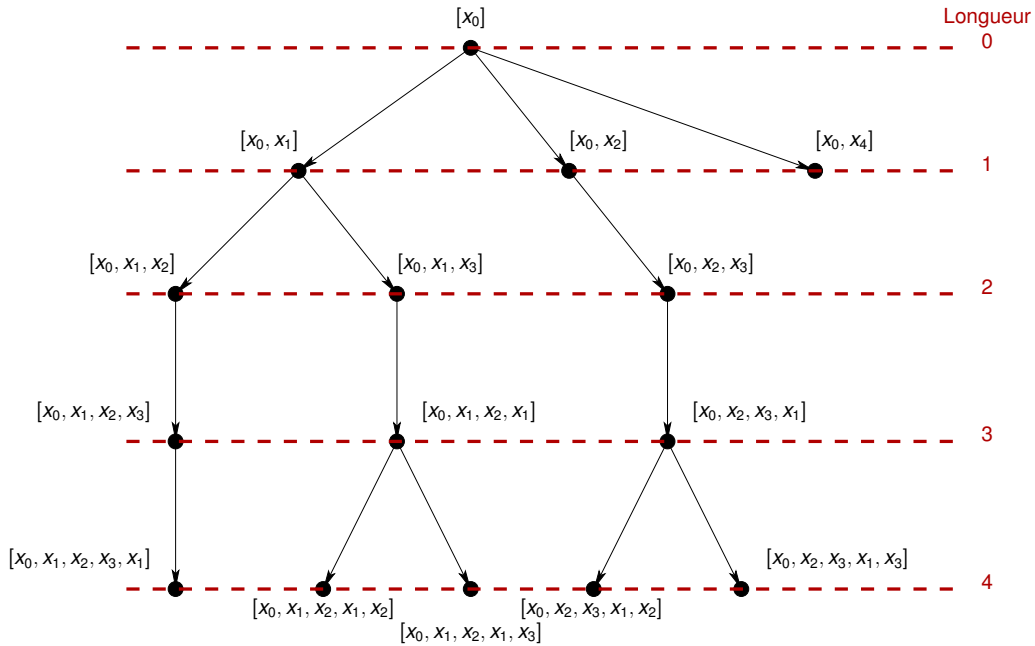
STRUCTURE DE $\mathcal{C}_{x_0}$

L'objectif est de parcourir tout l'ensemble $\mathcal{C}_{x_0}$

Un petit exemple



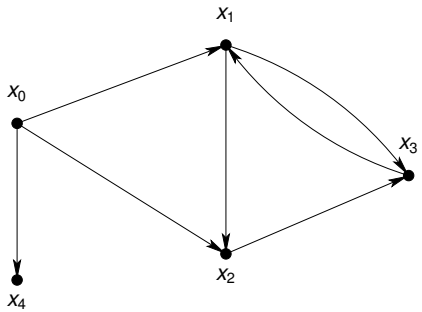
Arbre des chemins



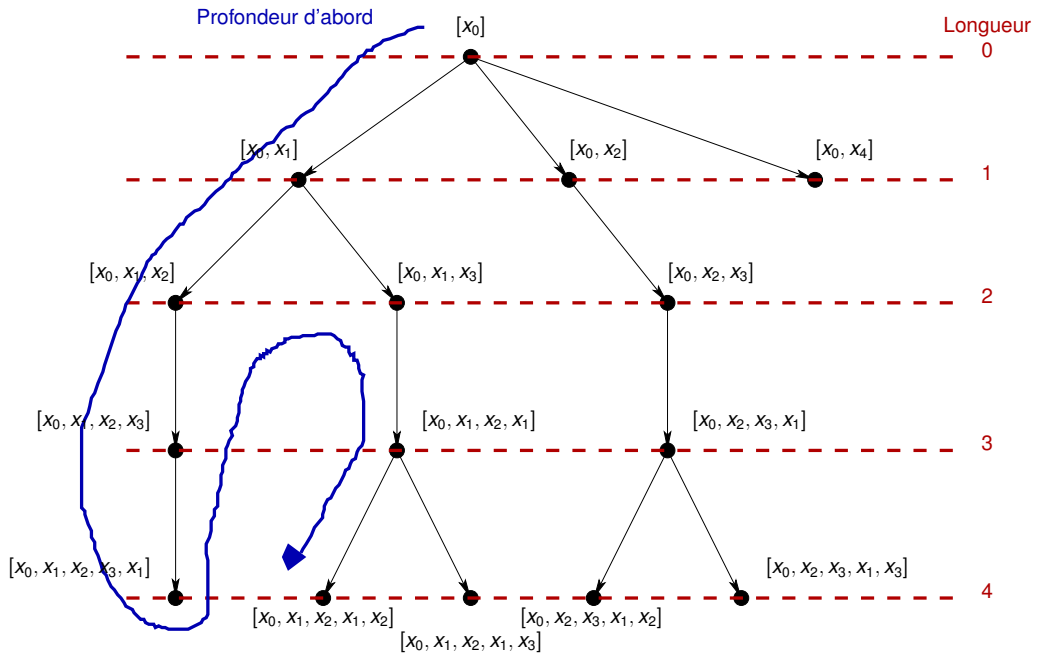
STRUCTURE DE $\mathcal{C}_{x_0}$

L'objectif est de parcourir tout l'ensemble $\mathcal{C}_{x_0}$

Un petit exemple



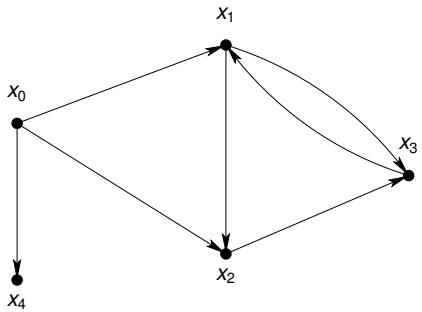
Arbre des chemins



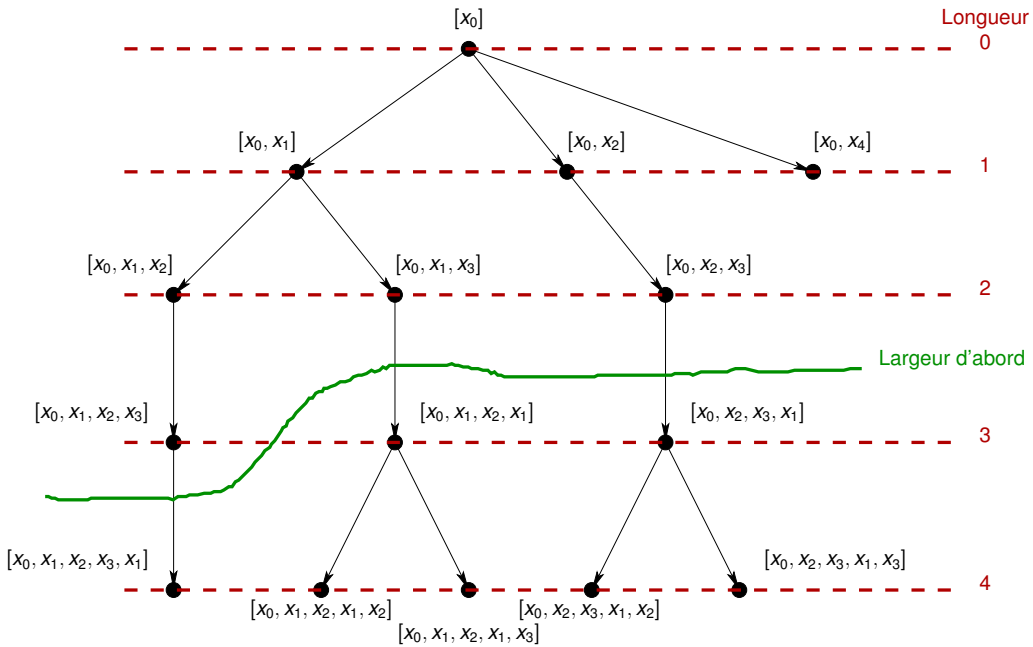
STRUCTURE DE $\mathcal{C}_{x_0}$

L'objectif est de parcourir tout l'ensemble $\mathcal{C}_{x_0}$

Un petit exemple



Arbre des chemins



UN ALGORITHME GÉNÉRIQUE

PARCOURS_CHEMIN($\mathcal{G}$,x_0)

Données: Un graphe $\mathcal{G}$ et un sommet x_0 de $\mathcal{G}$

Résultat: Visite tous les chemins du graphe

$E \leftarrow \{[x_0]\}$

`// ensemble de séquences de sommets`

repeat

1 $c =$ `Extraire` (E)

2 `Visiter` (c)

3 **for** $y \in$ `Voisins` **(**`Extremite` **(**c**))**

4 `Insère` $(E,c.y)$

until $E=\emptyset$

PREUVE DE L'ALGORITHME GÉNÉRIQUE

Preuve : correction partielle

Invariant $\mathbb{I}$ de l'itération :
Tout chemin du graphe d'origine x_0 non encore visité a un unique préfixe dans E.

- $\mathbb{I}$ est vrai lors de la première itération
- $\mathbb{I}$ est un invariant. En effet, soit c' un chemin d'origine x_0 non encore visité en fin d'itération alors
  - soit c est préfixe de c' et il existe un unique sommet y tel que $c.y$ soit préfixe de c', et $c.y$ est inséré dans E et est l'unique préfixe de c' dans E
  - soit c n'est préfixe pas de c', $\mathbb{I}$ vrai en début d'itération entraine qu'il existe un unique préfixe de c' dans E, comme c n'est pas préfixe de c', à fortiori $c.y$. Donc l'unicité est assurée
- si l'itération se termine alors tous les chemins d'origine x_0 ont été visités

PREUVE DE L'ALGORITHME GÉNÉRIQUE

Preuve : correction partielle

Invariant $\mathbb{I}$ de l'itération :
Tout chemin du graphe d'origine x_0 non encore visité a un unique préfixe dans E.

- $\mathbb{I}$ est vrai lors de la première itération
- $\mathbb{I}$ est un invariant. En effet, soit c' un chemin d'origine x_0 non encore visité en fin d'itération alors
  - soit c est préfixe de c' et il existe un unique sommet y tel que $c.y$ soit préfixe de c', et $c.y$ est inséré dans E et est l'unique préfixe de c' dans E
  - soit c n'est préfixe pas de c', $\mathbb{I}$ vrai en début d'itération entraine qu'il existe un unique préfixe de c' dans E, comme c n'est pas préfixe de c', à fortiori $c.y$. Donc l'unicité est assurée
- si l'itération se termine alors tous les chemins d'origine x_0 ont été visités

Preuve : terminaison

Tel que, rien ne prouve la terminaison de l'algorithme

SPÉCIALISATION

Recherche des sommets accessibles à partir de x_0 (parcours des sommets du graphe)

PARCOURS_CHEMIN_SOMMETS($\mathcal{G}$,x_0)

Données: Un graphe $\mathcal{G}$ et un sommet x_0 de $\mathcal{G}$
Résultat: Visite tous les sommets du graphe

```
Visiter (x0)
E ← {[x0]}
// ensemble de séquences de sommets
repeat
    c = Extraire (E)
    for y ∈ Voisins (Extremité (c))
        if y non visité
            Visiter (y)
            Insère (E,c.y)
until E=∅
```

Questions

Transformer l'invariant de l'algorithme générique pour l'adapter à cette situation
Prouver la terminaison
Est-il besoin de stocker l'ensemble du chemin ?

SPÉCIALISATION DE E

Lorsque l'on prend une structure particulière pour E on obtient différents parcours des sommets du graphe

PARCOURS_CHEMIN_SOMMETS($\mathcal{G}$,x_0)
Données: Un graphe $\mathcal{G}$ et un sommet x_0 de $\mathcal{G}$
Résultat: Visite tous les sommets du graphe

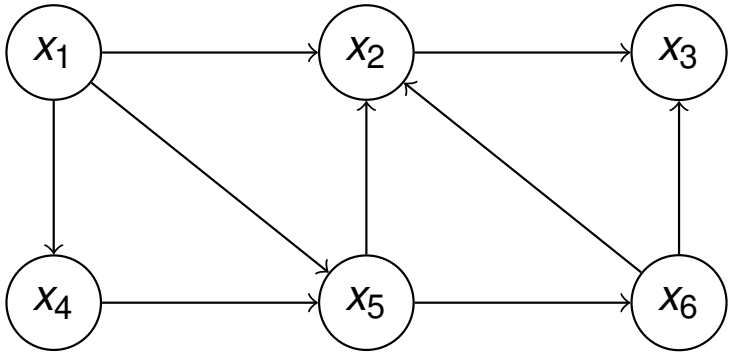
```
Visiter (x0)
E ← {[x0]}
// ensemble de séquences de sommets
repeat
    c = Extraire (E)
    for y ∈ Voisins (Extremité (c))
        if y non visité
            Visiter (y)
            Insère (E,c.y)
until E=∅
```

Questions

Si E est une pile, quel est le parcours obtenu ? (algorithme DFS : Depth-First Search)
Si E est une file, quel est le parcours obtenu ? (algorithme BFS : Breadth-First Search)

CONSTRUIRE UNE ROUTE

Graphe $\mathcal{G} = (\mathcal{X}, \mathcal{A}, \mathcal{V})$ avec $\mathcal{V}$ valuation positive des arcs

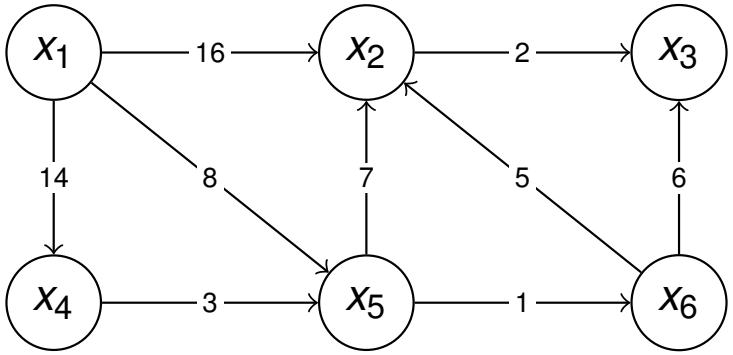


Notations :
$v(x_i, x_j) \geqslant 0$ est la valeur de l'arc (x_i, x_j)
source s : par exemple $s = x_1$

Exemple extrait du cours de Kevin Wayne

CONSTRUIRE UNE ROUTE

Graphe $\mathcal{G} = (\mathcal{X}, \mathcal{A}, \mathcal{V})$ avec $\mathcal{V}$ valuation positive des arcs



Notations :
$v(x_i, x_j) \geqslant 0$ est la valeur de l'arc (x_i, x_j)
source s : par exemple $s = x_1$

Exemple extrait du cours de Kevin Wayne

ALGORITHME DE DIJKSTRA(1956)

Algorithme_Dijkstra_V0($\mathcal{G}$,s)

Données: $\mathcal{G} = (\mathcal{X}, \mathcal{A}, \mathcal{V})$ graphe orienté valué positif,[a] s sommet de $\mathcal{G}$

Résultat: Visite tous les sommets accessibles x du graphe à partir de s et calcule la valeur minimale des chemins de s à x

foreach $x \in \mathcal{X}$ **do**
  $d(x) = +\infty$
$S \leftarrow \{s\}$
$d(s) = 0$
repeat
  `Choisir` $y \notin S$ minimisant

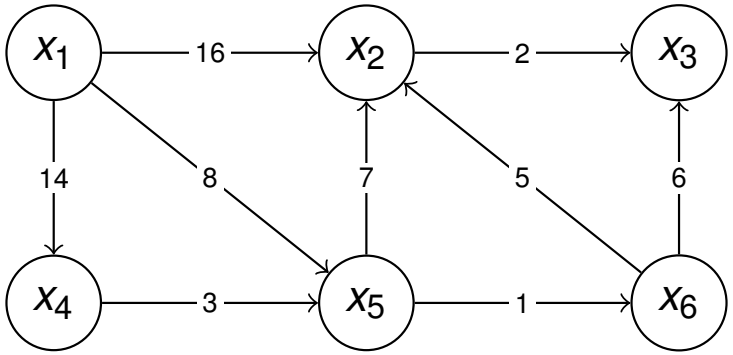
$$\pi(y) = \min_{x \in S, (x,y) \in \mathcal{A}} d(x) + v(x, y)$$

  $S \leftarrow S \cup \{y\}$
  $d(y) = \pi(y)$
until il n'existe pas $y \notin S$ **et** $x \in S$ **tels que** $(x, y) \in \mathcal{A}$

a. $\mathcal{A}$ ensemble des arcs, $v(x, x) = 0$, $v(x, y) \geqslant 0$ si $(x, y) \in \mathcal{A}$, $v(x, y) = +\infty$ si $(x, y) \notin \mathcal{A}$

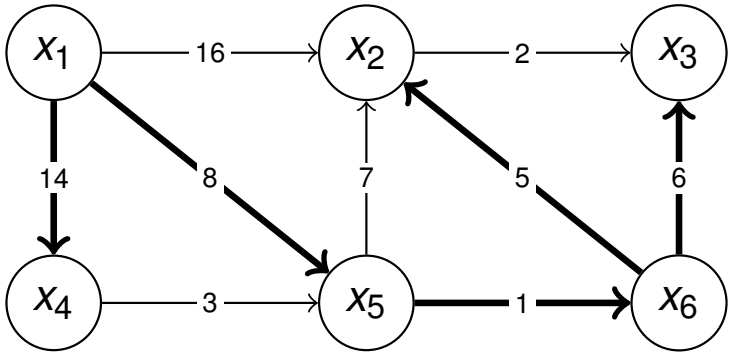
Algorithme glouton : (S, d) est une partie d'une solution optimale

EXEMPLE



	S	$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$d(6)$
0	$\{x_1\}$	0	/	/	/	/	/
1							
2							
3							
4							
5							

EXEMPLE



	S	$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$d(6)$
0	$\{x_1\}$	0	/	/	/	/	/
1	$\{x_1, x_5\}$	0	/	/	/	8	/
2	$\{x_1, x_5, x_6\}$	0	/	/	/	8	9
3	$\{x_1, x_5, x_6, x_4\}$	0	/	/	14	8	9
4	$\{x_1, x_5, x_6, x_4, x_2\}$	0	14	/	14	8	9
5	$\{x_1, x_5, x_6, x_4, x_2, x_3\}$	0	14	15	14	8	9

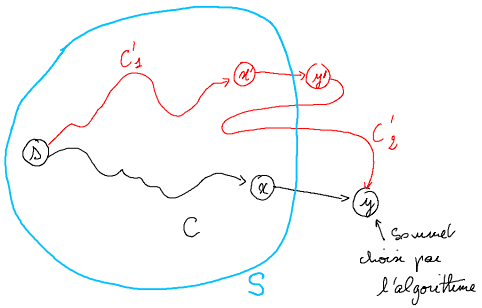
PREUVE

Invariant $\mathbb{I}$

Pour tout sommet $x \in S$ $d(x)$ est la valeur minimale d'un chemin de s à x

Preuve par induction sur $|S|$

- Cas de base $|S| = 1$, $d(s) = 0$
- Induction : Supposons (I) vérifié pour $|S| \geqslant 1$



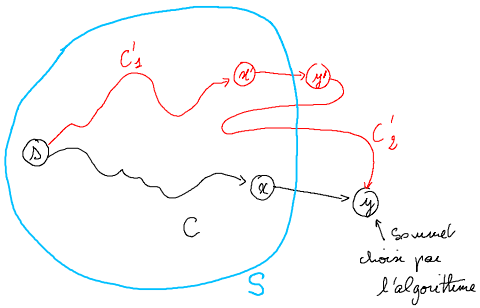
PREUVE

Invariant $\mathbb{I}$

Pour tout sommet $x \in S$ $d(x)$ est la valeur minimale d'un chemin de s à x

Preuve par induction sur $|S|$

- Cas de base $|S| = 1$, $d(s) = 0$
- Induction : Supposons (I) vérifié pour $|S| \geqslant 1$



$$
\begin{aligned}
v(C') &\geqslant v(C'_1) + v(x', y') \text{ les valeurs des arcs sont } \geqslant 0 \\
&\geqslant d(x') + v(x', y') \text{ par hypothèse de récurrence} \\
&\geqslant \pi(y') \text{ par définition de } \pi(y') \\
&\geqslant \pi(y) \text{ car l'algorithme a choisi } y \text{ plutôt que } y' \\
&\geqslant v(C) = d(y)
\end{aligned}
$$

Donc $d(y)$ est la valeur d'un chemin de s à y minorant toute autre valeur de chemin, c'est donc la valeur minimale recherchée.

UGA Université Grenoble Alpes

COMPLEXITÉ

La complexité dépend fondamentalement du coût de l'opérateur `Choisir`

- Il faut donc choisir une structure de donnée telle que la recherche du min soit efficace
- Le calcul de $\pi(z)$ peut être redondant, on peut actualiser la valeur de π à chaque étape

au pire en $\mathcal{O}((m+n)\log n)$ le $\log n$ apparaît en utilisant un tas binaire

Algorithme_Dijkstra($\mathcal{G}$,s)

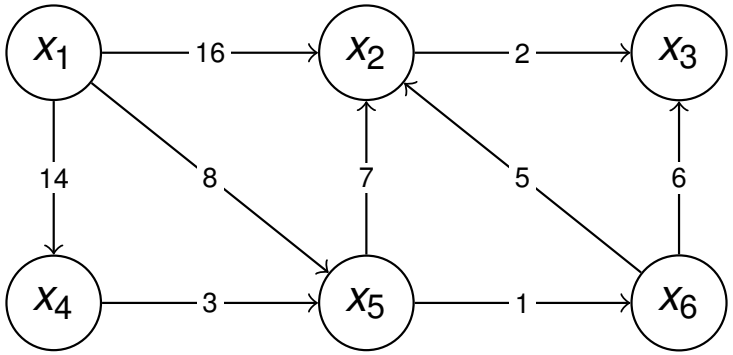
Données: Un graphe $\mathcal{G}$ valué et un sommet s de $\mathcal{G}$
Résultat: Visite tous les sommets du graphe et calcule la valeur minimale des chemins de s à tout autre sommet du graphe

$d(s) = 0$
$S \leftarrow \{s\}$
for $y \in \mathcal{X} - S$
  $d(y) = v(s, y)$
`// ensemble de sommets (chemins) dont la valeur minimale` d `est établie`
repeat
  `Choisir` $y \notin S$ minimisant

$$\pi(y) = \min_{x \in S, (x,y) \in \mathcal{A}} d(x) + v(x, y)$$

  $S \leftarrow S \cup \{y\}$
  $d(y) = \pi(y)$
  for $z \in \mathcal{X} - S$
    $d(z) = \min(d(z), d(y) + v(y, z))$
until il existe $y \notin S$ **et** $x \in S$ **tels que** $(x, y) \in \mathcal{A}$

EXEMPLE



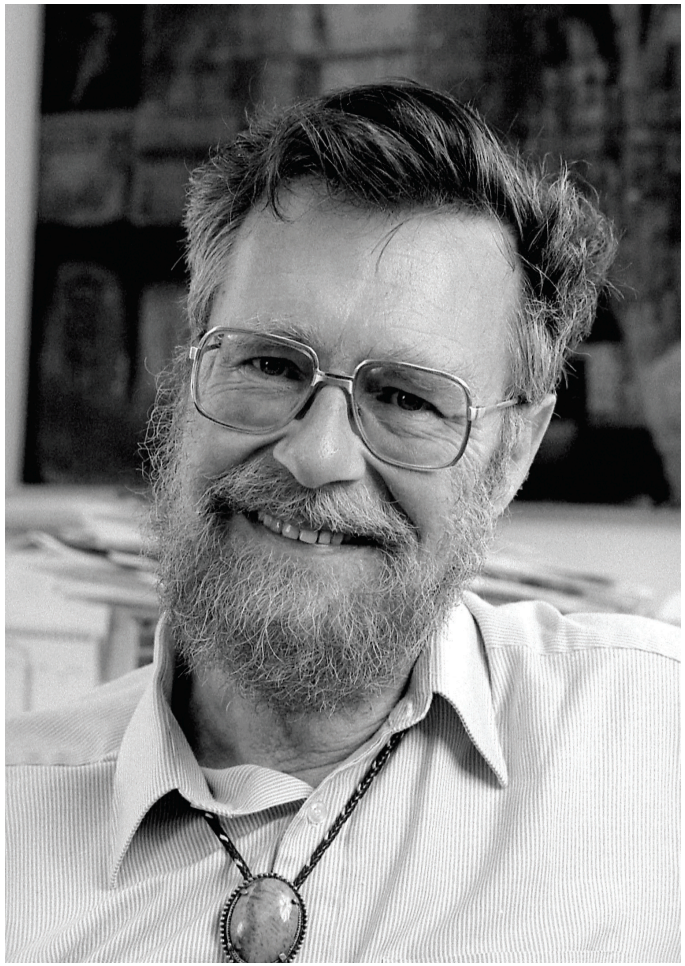
	S	$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$d(6)$
0	$\{x_1\}$	0	16	$+\infty$	14	8	$+\infty$
1							
2							
3							
4							
5							

HISTORIQUE

Extrait Interstice

Cet algorithme est dû à Edsger W. Dijkstra (1930-2002), qui est l'un des trois ou quatre informaticiens les plus importants du XXe siècle : il reçut, en 1972, le prestigieux Turing Award, équivalent du Prix Nobel ou de la médaille Fields. Il publia l'algorithme décrit ici en 1959 (« A Note on Two Problems in Connexion with Graphs », Numerische Mathematik, vol.1, PP. 269-271, 1959), l'année où il soutint sa thèse à l'université d'Amsterdam.

Extrait de communications of the ACM 2010 vol. 53 no. 8



EDSGER WYBE DIJKSTRA

Turing Award 1972

- First compiler for Algol-60
- Minimum Spaning Tree, Shortest path (1956)
- Notes on Structured Programming,(Go To statement considered harmful)

EWD84[illegible] 1

Dijkstra E W. A Note on Two Problems in Connexion with Graphs
Numerische Mathematik 1: 269-271, 1959
[Mathematisch Centrum, Amsterdam]

Abstract. We consider a graph with n vertices, all pairs of which are connected by an edge; each edge is of given positive length. The following two basic problems are solved.

Problem 1. Construct the tree of minimal total length between the n vertices. (A tree is a graph with one and only one path between any two vertices.)

Problem 2. Find the path of minimal total length between two given vertices.

Edsger W.Dijkstra
Burroughs/Eindhoven University of Technology
Plataanstraat 5
5671 AL Nuenen, The Netherlands

I found my solution to the second problem in 1956 after having invented the problem first. At the time I was the main programmer of the Mathematical Centre in Amsterdam, where the construction of its second automatic computer was on the verge of completion, and for the celebration of its inauguration we needed a nice demonstration: it should solve a problem that could be easily stated to a predominantly lay audience. For the purpose of the demonstration I drew a slightly simplified map of the Dutch railroad system, someone in the audience could ask for the shortest connection between, say, Harlingen and Maastricht, and the ARMAC would print out that shortest route town by town. The demonstration was a great success; I remember that I could show that inversion of source and destination could influence the computation time required. The speed of the ARMAC and the size of the map were such that one minute always sufficed.

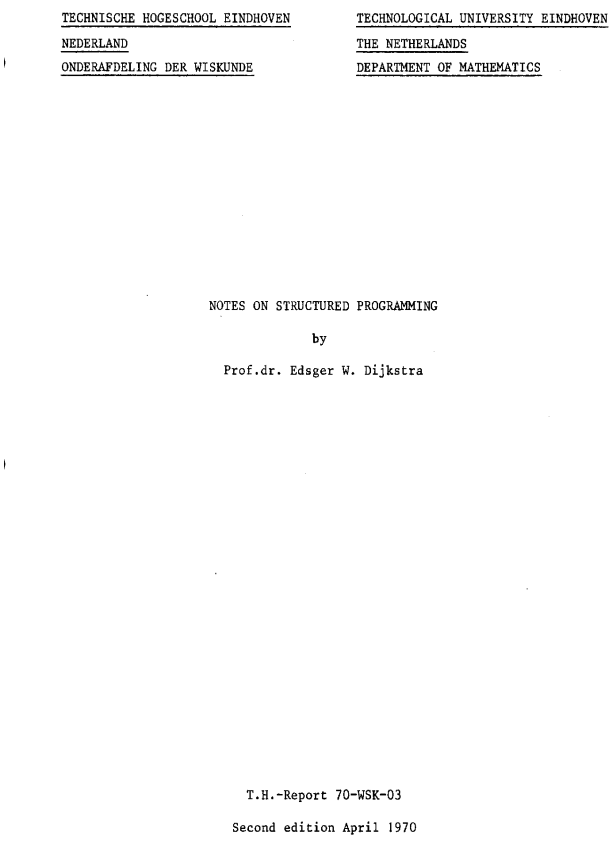
I found the solution to the first problem about a year later, when the next machine was under construction and I was asked whether I could minimize the amount of cable needed for the back-panel wiring, which had to connect pins tree-fashion. I remember its discovery more vividly than that for the shortest route: my wife and I were having a cup of coffee in front of some café at the sunny side of the Damrak. I found both solutions

../2

EDSGER WYBE DIJKSTRA

Turing Award 1972

- First compiler for Algol-60
- Minimum Spaning Tree, Shortest path (1956)
- Notes on Structured Programming,(Go To statement considered harmful)



TECHNISCHE HOGESCHOOL EINDHOVEN
NEDERLAND
ONDERAFDELING DER WISKUNDE

TECHNOLOGICAL UNIVERSITY EINDHOVEN
THE NETHERLANDS
DEPARTMENT OF MATHEMATICS

NOTES ON STRUCTURED PROGRAMMING

by

Prof.dr. Edsger W. Dijkstra

T.H.-Report 70-WSK-03

Second edition April 1970

EDSGER WYBE DIJKSTRA

Turing Award 1972

- First compiler for Algol-60
- Minimum Spaning Tree, Shortest path (1956)
- Notes on Structured Programming,(Go To statement considered harmful)

EWD 316:

A SHORT INTRODUCTION
TO THE ART
OF PROGRAMMING

by prof. dr. Edsger W. Dijkstra

Dictaatnr. 2.268 Prijs f 4,--

ÉCHELLES DE MOTS

Étant donnés 2 mots de même longueur, trouver une séquence de mots allant de l'un à l'autre en ne changeant qu'une seule lettre à la fois.
De bleu à noir
bleu -> blés -> bues -> buis -> luis -> lois -> loir -> noir
De Jean à Marc
jean -> jeun -> jeux -> feux -> feus -> fers -> fars -> mars -> marc.


Doublets de Lewis Carroll, repris par Knuth