P'tit IndeX d'InformatiquE

304 entrées à la date du samedi 14 novembre 2026

11 verres de sirop, 4 cookies, 52h heures de travail

N. Tillier, P. Chaudagne

« The Pixies, or the Devonshire fairies, are just like the northern elves. The popular idea of them is that they are small creatures—pigmies—dressed in green, and are fond of dancing. Some of them live in the mines, where they show the miners the richest veins of metal just like the German dwarfs; others live on the moors, or under the shelter of rocks; others take up their abode in houses, and, like the Danish and Swedish elves, are very cross if the maids do not keep the places clean and tidy others, like the will-o'-the-wisps, lead travellers astray, and then laugh at them. »

John Thackray Bunce, *Fairy tales, their origin and meaning*

« À bien prendre les choses, le dictionnaire est le livre par excellence : tous les autres livres sont dedans, il ne s'agit plus que de les en tirer. »

Anatole France, *La Vie littéraire, 2e série*

« Programs must be written for people to read, and only incidentally for machines to execute. »

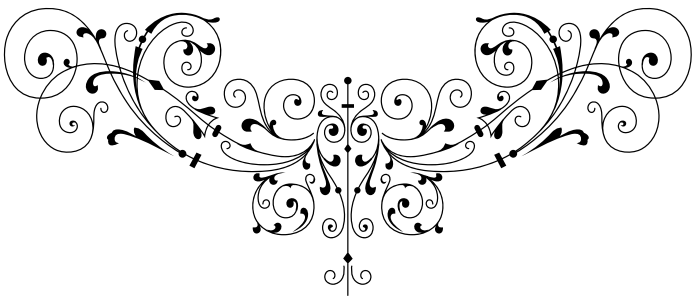
H. Abelson, G. Sussman, J. Sussman, *Structure and Interpretation of Computer Programs*

« An algorithm must be seen to be believed. »

Donald Knuth, *The Art of Computer Programming, Volume 1*

Afin d'améliorer ce document n'hésitez pas à nous transmettre les éventuelles critiques, remarques ou corrections qui vous viendraient à l'esprit à sa lecture, à l'adresse :

contact@paulch.fr ou nathanti@crans.org



Nous avons réalisé ce dictionnaire comme une synthèse des notions abordées au cours des deux années de classe préparatoire MP2I puis MPI. Sans prétendre à une exhaustivité complète, nous espérons qu'il représente un condensé de ce à quoi vous serez confrontés, avec des explications simples et des exemples afin de vous aider à rapidement appréhender les différentes définitions.

Ce projet tire son inspiration du *Petit Dico de Maths* de M. Appel. Le contenu s'inspire quant à lui des cours d'informatique et exercices de M. Rebout et M. Bianquis. Nous souhaitons les remercier pour cette inspiration ainsi que pour leurs enseignements des années passées.

Ce dictionnaire est encore en cours de rédaction.

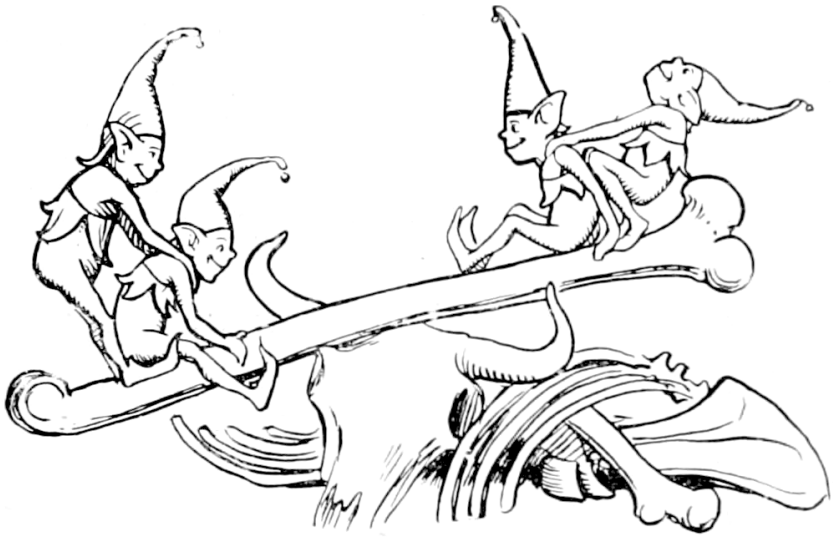


Illustration par John Dickson Batten pour *More English Fairy Tales*, Joseph Jacobs

A

Absence d'interblocage Se dit d'un verrou pour lequel, lorsqu'au moins un fil tente d'acquérir ou de relacher le verrou, au moins un fil finit par y parvenir.

Absence de famine Se dit d'un verrou pour lequel, lorsqu'un certain fil tente d'acquérir ou de relâcher le verrou, ce fil finit par y arriver.

Acyclique Un graphe $G = (V, E)$ est dit acyclique s'il ne contient aucun cycle.

Algorithme Programme fini prenant en entrée une quantité finie (non nécessairement bornée) de données.
On considérera que les algorithmes s'exécutent sur un ordinateur idéal disposant d'une quantité illimitée de mémoire, et que la pile d'appel ne déborde jamais.
Un algorithme peut
- renvoyer un résultat en temps fini
- diverger, c'est-à-dire ne pas renvoyer de résultat quand le programme plante ou ne pas terminer

Exemple : Cet algorithme ne termine pas :

Algorithme long

Entrée : un entier n
Sortie : 42 ?
  $i \leftarrow 0$
  tant que $i < 861$ **faire**
    $i \leftarrow (i + n) \bmod 861$
  retourner 42

Algorithme 1. – Algorithme long

Algorithme glouton Classe d'algorithme pour des problèmes d'optimisation, qui consiste à réaliser des choix localement optimaux afin de potentiellement aboutir à une solution optimale globale.
Si l'algorithme donne une solution optimale il sera souvent le meilleur pour le problème. S'il donne une solution pas trop mauvaise sur un problème difficile à résoudre exactement, il peut être utile.

Exemple : L'algorithme de Kruskal est un algorithme glouton.
↪ Kruskal (algorithme de)

Alphabet Ensemble fini Σ dont les éléments sont appelées lettres, symboles ou caractères.

Exemple : $\Sigma = \{a, b\}$ est un alphabet.

Ambigu
↪ Grammaire ambiguë

Ancêtres d'un nœud Dans un arbre, tous les nœuds hiérarchiquement supérieur à ce nœud.

Antilogie Formule propositionnelle P telle qu'aucune valuation la satisfait :

$$\mathrm{Mod}(P) = \varnothing$$

Antisymétrique
↪ Relation antisymétrique

Arbre
- Graphe acyclique et connexe.

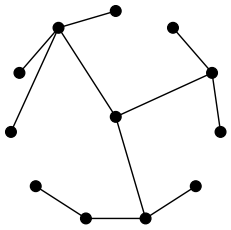


Figure 1. – Arbre

- Structure de données informatique, ensemble de nœuds qui ont chacun tout au plus un seul ancêtre chacun et un nombre quelconque de descendants.
- Ensemble fini A muni d'une relation binaire $\to$ telle que
  - il existe un unique élément r (la racine) vérifiant
  $$\forall x \in A, x \not\to r$$
  - pour tout élément x de A, il existe un unique père :
  $$\forall x \in A \setminus \{r\}, \exists! y \in A, y \to x$$
  - tout élément $x \in A \setminus \{r\}$ est relié à la racine :

  $\forall x \in A \setminus \{r\}, \exists n \in \mathbb{N}, x_1, ..., x_n \in A, r \to x_1 \to ... \to x$
- Végétal haut de plus de 7 mètres, dont la tige, appelée fût ou tronc, ne se garnit de branches et de feuilles qu'à une certaine hauteur. La partie constituée par les branches et les feuilles s'appelle houppier.



Figure 2. – Arbre dans la nature

↪ Graphe

Arbre binaire parfait Arbre binaire strict dans lequel toutes les feuilles sont à la même distance de la racine.
Autrement dit, c'est un arbre qui est
- soit une feuille
- soit de la forme $N(x, g, d)$ où g et d sont des arbres binaires parfaits de même hauteur

Arbre binaire strict Arbre dont tous les nœuds possèdent zéro ou deux fils.

Remarque :
- Un arbre binaire strict de hauteur h a au moins $2h + 1$ nœud (cas particulier d'un peigne).

Arbre couvrant Soit $G = (V, E)$ un graphe. Un arbre couvrant de G est un sous-graphe $H = (V_H, E_H)$ de G tel que :
- H est un arbre
- $V_H = V$

Remarque : Tout graphe connexe possède un arbre couvrant.

Construction d'un arbre couvrant

Entrées : un graphe connexe $G = (V, E)$
Sorties : un arbre couvrant T de G

```
T ← (V, E)
pour xy ∈ E faire
    si T − xy est connexe alors
        T ← T − xy
renvoyer T
```

Algorithme 2. – Construction d'un arbre couvrant

Ceci est une des algorithmes possibles, voir l'algorithme de Kruskal pour une autre proposition.
↪ Kruskal (algorithme de)

Arbre couvrant minimal Soit H un sous-graphe de G. H est un arbre couvrant minimal de G si $\rho(H)$ est minimal parmi les poids des arbres couvrants de G.
↪ Kruskal (algorithme de)

Arène Graphe orienté biparti représentant les états d'un jeu.

Arête
- Dans un graphe non orienté, paire de deux sommets $\{x, y\}$ avec $x \neq y$ si on n'autorise pas la présence de boucle.
- Dans un graphe orienté, 2-uplet de deux sommets (x, y) avec $x \neq y$ si on n'autorise pas la présence de boucle.

Arité L'arité d'un nœud d'un arbre est son nombre de fils.

Arrêt
↪ Problème de l'arrêt

Assainisseur
↪ Sanitizer

Asymétrique
↪ Relation asymétrique

Atomicité (base de données) Garantie que chaque opération s'exécute entièrement ou pas du tout. C'est important dans le cas d'une panne.

Attracteur En théorie des jeux, l'attracteur d'un joueur i est l'ensemble des positions gagnantes pour i.

Remarque : On peut calculer l'attracteur des deux joueurs en $O(|E| + |V|)$ avec l'algorithme suivant.

Calcul des attracteurs

Entrée : Un jeu (G, s_0, T_0, T_1)
Sortie : les attracteurs des joueurs 0 et 1

```
A_0 ← ∅
A_1 ← ∅
d ← (d_+(s); s ∈ V)
G_M ← miroir(G)
fonction Explore(s, joueur)
    A_joueur ← A_joueur ∪ {s}
    pour s' successeur de s dans G_M tel que
    s' ∉ A_0 ⊔ A_1 faire
        si s' contrôlé par joueur faire
            Explore(s', joueur)
        sinon
            d[s'] ← d[d'] − 1
            si d[s'] = 0 alors
                Explore(s', joueur)
pour s ∈ T_0 faire
    Explore(s, 0)
pour s ∈ T_1 faire
    Explore(s, 1)
renvoyer A_0, A_1
```

Algorithme 3. – Calcul des attracteurs

On part des états gagnants d'un joueur pour remonter aux sommets de son attracteur, que ce soit par un choix de ce joueur ou en forçant la main de son adversaire, dans le cas où ce dernier ne peut qu'aboutir sur une position gagnante pour notre joueur.



Backtracking Le backtracking consiste à effectuer un parcours en profondeur de l'arbre des solutions partielles, en élaguant une branche (abandonnant sa recherche) dès qu'on est sûr qu'elle ne peut contenir aucune solution complète admissible.
Dans un problème d'optimisation, on peut également élaguer lorsqu'on est sûr de ne pas avoir de meilleure solution admissible dans la branche que la meilleur solution trouvée à cet instant. On a alors souvent besoin d'une heuristique h prenant en entrée une solution partielle, et vérifiant

$$\text{score}(x) \leq h(y)$$

si la solution complète x étend la solution partielle y.

Base de données Une base de données est une collection organisée d'objets, gérée par un SGBD (Système de Gestion de Base de Données).
↪ Système de Gestion de Bases de Données

Bellman-Ford (algorithme de) Algorithme de recherche d'un plus court chemin dans un graphe pondéré.

Algorithme de Bellman-Ford

Entrée : un graphe $G = (V, E)$
Sortie :

Algorithme 4. – Algorithme de Bellman-Ford

BFS
↪ Parcours en largeur

Bidimensionel
↪ Tableau bidimensionel statique

Bit Unité binaire d'information la plus simple, codant un 0 ou un 1, ou bien `true` ou `false`.

Bit de signe Idée d'implémentation naïve d'entiers signés, consistant à utiliser un bit en plus valant 0 dans le cas d'un nombre positif, et 1 dans le cas d'un nombre négatif.

Bon ordre Ordre sur un ensemble E pour lequel toute partie non vide admet un minimum.

Exemple : La relation d'inférierité est un bon ordre sur $\mathbb{N}$ et $\mathbb{N}^*$ mais pas sur $\mathbb{Z}$.

Booléen Vrai ou faux. En C comme en OCaml, on les note `true` et `false`. Il faut utiliser l'en-tête `<stdbool.h>` en C pour utiliser le type `bool`.

Bottom-up Approche ascendante pour résoudre un problème, consistant à résoudre des instances de taille de plus en plus grande, jusqu'à résoudre le problème initial.
C'est une solution itérative, pouvant réduire la complexité spatiale.
↪ Fibonacci (suite de)

Boucle for Sert à répéter l'exécution d'un séquence d'instructions. Ces boucles sont surtout utilisée pour parcourir des tableaux.

- En C, on utilise la syntaxe

```c
for (int i = 0; i < n; i++) {
  /* code */
}
```

pour parcourir tous les entiers de 0 à $n-1$.
Remarque :
- Pour sortir précipitamment de la boucle sans vérifier la condition, on peut utiliser le mot-clé `break`.
- Pour parcourir ces entiers dans l'ordre décroissant, on pourra commencer à $i = n-1$ puis décrémenter i tant que $i \geq 0$.

```c
for (int i = n - 1; i >= 0; i--) {
  /* code */
}
```

- Pour revenir à la condition en sautant le reste des instructions dans la boucle, on peut utiliser le mot-clé `continue`.

- En OCaml, on utilise la syntaxe

```ocaml
for i = 0 to n - 1 do
  (* code *)
done
```

Remarque :
- Attention : le dernier indice est inclus.
- Pour parcourir ces entiers dans l'ordre décroissant, ou pourra utiliser `downto` et intervertir les bornes.

```OCaml
for i = n - 1 downto 0 do
  (* code *)
done
```

- Contrairement au C, il n'y a pas moyen de sortir précipitamment de la boucle.

Fondamentalement, en C une boucle `for` n'est rien d'autre qu'une boucle `while` déguisée. Cette boucle `for`

```C
for (initialisation; condition; itération) {
  /* code */
}
```

Peut se réécrire comme ceci

```C
{
  initialisation
  while (condition) {
    /* code */
    itération
  }
}
```

Ce point de vue est pratique pour parcourir une chaîne de caractères jusqu'à sa fin, sans utiliser `strlen` (qui est en $O(n)$ où n est la taille de la chaîne de caractères).

```C
char* texte = "Lorem ipsum.";
for (int i = 0; texte[i] != '\0'; i++) {
  /* code */
}
```

Boucle while Sert à répéter l'exécution d'une séquence d'instructions tant qu'une condition est vérifiée.

- En C, on utilise la syntaxe

```C
while (condition) {
  /* code */
}
```

Remarque :

- Pour sortir précipitamment de la boucle sans vérifier la condition, on peut utiliser le mot-clé `break`.
- Pour revenir à la condition en sautant le reste des instructions à l'intérieur de la boucle, on peut utiliser le mot-clé `continue`.

- En OCaml, on utilise la syntaxe

```OCaml
while condition do
  (* code *)
done
```

Remarque : Contrairement au C, il n'y a pas moyen de sortir arbitrairement de la boucle. On pourra utiliser une référence pour reproduire ce comportement.

Boulangerie de Lamport
↪ Lamport (algorithme de la boulangerie de)

Boutisme On le désigne à la fois pour :

- désigner l'ordre des bits dans la représentation binaire d'un chiffre
- un entier étant souvent stocké sur plusieurs octets, désigner l'orientation des octets de poids fort et de poids faible pour le représenter

On distingue

- le gros-boutisme (*big endian*), avec les bits/octets de poids fort en premier
- le petit-boutisme (*little endian*), avec les bits/octets de poids faible en premier

Bozo-sort Algorithme de tri consistant à intervertir au hasard deux éléments de la séquence jusqu'à ce qu'elle soit triée.
À n'utiliser qu'en dernier recours.

Branch and Bound Technique accélérant la résolution d'un problème d'optimisation par backtracking.

Byte Aussi appelé multiplet, c'est la plus petite unité logiquement adressable par un programme sur un ordinateur. Sur la plupart des systèmes, ce sont des bytes de 8 bits qui sont utilisés (octet).



C Le langage C est un langage de programmation impératif inventé au début des années 1970.

Calculable Fonction f telle qu'il existe un algorithme A terminant en temps fini pour toute entrée valide x, en renvoyant $f(x)$.
On dit que A réalise f.

Remarque : Il existe des fonctions $f : \mathbb{N} \to \mathbb{N}$ non calculables. En effet, $\mathbb{N}^{\mathbb{N}}$ n'est pas dénombrable, tandis que l'ensemble des algorithmes est dénombrable.
↪ Castor affairé

Calloc En C, la fonction `calloc` permet de réserver une zone mémoire et de mettre tous ses bis à zéro. Elle est déclarée dans l'en-tête `<stdlib.h>`.

```c
void *calloc(size_t nb_octets);
```

Cast
↪ Transtypage

Castor affairé Désigne
- un programme dont l'exécution termine, calculant une valeur la plus grande possible parmi tous les programmes de même taille.
- le problème de décision indécidable défini par
  - **Instance :** le code source d'un programme OCaml (par exemple) P
  - **Question :** P est-il un castor affairé ?

Remarque : Vous avez un excellent exercice à ce sujet.

Catalan (nombres de) Suite intervenant dans des problèmes combinatoires, définie récursivement par :

$$c_0 = 1 \text{ et } c_{n+1} = \sum_{k=0}^{n} c_k c_{n-k}$$

On peut montrer que

$$c_{n+1} = \frac{2(2n+1)}{n+2} c_n$$

Et

$$c_n = \frac{1}{n+1}\binom{2n}{n}$$

Exemple : Il y a c_n arbres binaires stricts avec n nœuds internes.

Chaîne de caractères Suite ordonnée de caractères stockant du texte.
Une chaîne de caractère est de type `char*` en C, et le caractère `\0` marque la fin de la chaîne. En OCaml, on utilise le module `String`.

Implémentation : En C, on peut utiliser `strlen` après avoir inclus `<string.h>`.

```c
#include <string.h>

int main() {
  size_t len = strlen("Hello, world!");
}
```

On peut également l'implémenter soi-même de la fonction suivante

```c
int strlen(char* str) {
  int len = 0;
  while (str[len] != '\0') {
    len += 1;
  }
  return len;
}
```

En OCaml, il suffit d'utiliser `String.length`.

```ocaml
let length = String.length "Hello, world!"
```

Chaîne eulérienne Chemin d'un graphe parcourant chaque arête du graphe une unique fois.

Chaînée
↪ Liste simplement chaînée, Liste doublement chaînée

Chemin
↪ Graphe chemin

Chemin alternant Soit $G = (V, E)$ un graphe non orienté et C un couplage de G. Un chemin alternant pour C est un chemin élémentaire dont les arêtes sont alternativement dans C et dans $E \setminus C$

Chemin augmentant Soit $G = (V, E)$ un graphe non orienté et C un couplage de G. Un chemin augmentant

pour C est un chemin alternant dont les deux extrémités sont libres pour C.

Chemin élémentaire Chemin n'utilisant pas deux fois le même sommet. $x_0, ..., x_n$ est un tel chemin si les x_i sont distincts.

Chemin eulérien
↪ Chaîne eulérienne

Chemin simple Chemin n'utilisant pas deux fois la même arête. $x_0, ..., x_n$ est un tel chemin si les $\{x_i, x_{i+1}\}$ sont distincts.

Church-Turing (thèse de) Thèse du mathématicien Alonzo Church portant sur la calculabilité. Elle formalise la notion de fonction calculable : les fonctions calculables sont celles qu'on peut représenter par une machine de Turing.

Classe d'équivalence Pour une relation d'équivalence $\sim$, la classe d'équivalence d'un élément $x \in E$ est l'ensemble des éléments en relation avec x :

$$\mathrm{Cl}(x) = [x] = \{y \in E \mid x \sim y\}$$

Exemple : Dans un graphe non orienté, pour la relation « être accessible depuis », la classe d'équivalence d'un sommet est sa composante connexe.

Clé étrangère Dans une table d'une base de données, clé permettant de référencer la clé primaire d'une autre table. Cela permet de relier différentes tables.

Clé primaire Dans une table d'une base de données, clé permettant d'identifier de manière unique une ligne.

Clustering hiérarchique ascendant Technique d'apprentissage non supervisé, regroupant les objets en classe de manière arborescente.

Clusterin hiérarchique ascendant

Entrée : un ensemble de N échantillons $x_1, ..., x_N$ non étiquetés
Sortie : bonne question

```
créer N classes {x_1}, ..., {x_N}
tant que nécessaire faire
    déterminer C_i et C_j les deux classes les plus proches
    remplacer C_i et C_j par Fusion(C_i, C_j)
```

Algorithme 5. – Clusterin hiérarchique ascendant

Pour calculer la distance entre deux classes à partir d'une distance d entre les échantillons, on peut utiliser plusieurs liaisons.

Nom	Distance $\delta(C, C')$
Liaison simple	$\min(d(x,y), (x,y) \in C \times C')$
Liaison complète	$\max(d(x,y), (x,y) \in C \times C')$
Liaison Hausdorff	$\max_{x \in C \cup C'} \min_{y \in C \cup C'} d(x,y)$
Liaison moyenne	$\frac{1}{\lvert C\rvert \cdot \lvert C'\rvert} \sum_{x \in C, y \in C'} d(x,y)$
Liaison barycentrique	$d(x_C, x_{C'})$ où $x_C, x_{C'}$ isobarycentres

On appelle dendrogramme le diagramme représentant les regroupements hiérarchiques successifs effectués.

Code à longueur fixe Un code binaire f est à longueur fixe s'il existe $n \in \mathbb{N}^*$ tel que f soit à valeur dans $\{0,1\}^n$. Autrement dit tous les $f(a)$ ou $a \in \Sigma$ sont de même longueur. Il est dit à longueur variable sinon.

Code ambigu Un code binaire est ambigu s'il n'est pas uniquement déchiffrable : il existe deux mots différents donnant le même code.

Exemple : Le code binaire f sur $\{a, b, c\}$ défini par

$$f(a) = 1$$
$$f(b) = 0$$
$$f(c) = 10$$

est ambigu. Notamment

$$f(ab) = f(c)$$

Code binaire Un code binaire sur l'alphabet Σ est une application injective

$$f : \Sigma \to \{0,1\}^* \setminus \{\varepsilon\}$$

Elle code un caractère par une suite finie non vide de 0 et de 1. On lui associe son extension

$$\tilde{f} : \Sigma^* \to \{0,1\}^*$$
$$a_1 ... a_n \mapsto f(a_1) ... f(a_n)$$

Code préfixe Code binaire f sur Σ^* dont le code d'aucun caractère n'a pour préfixe le code d'un autre caractère : il n'existe pas de couple $(a, b) \in \Sigma^2$ tels que

$$a \neq b \text{ et } f(a) \text{ préfixe de } f(b)$$

Il est dit sans préfixe sinon.

Exemple : Le code binaire f sur $\{a, b, c, d\}$ défini par

$$f(a) = 00$$
$$f(b) = 01$$
$$f(c) = 10$$
$$f(d) = 11$$

est un code préfixe.

Code uniquement déchiffrable Un code binaire est uniquement déchiffrable si son extension est injective. Il est ambigu sinon.

Exemple : Le code binaire f sur $\{a, b\}$ défini par

$$f(a) = 0 \text{ et } f(b) = 1$$

est uniquement déchiffrable.

Cohérence (base de données) Garantie que toutes les copies ou instances d'une base de donnée sont exactes et à jour.

Coloriage
↪ Problème de coloriage

Commentaire Texte écrit à côté du code, afin de le documenter.
- En C, on l'écrit derrière deux barres obliques `//` jusqu'à la fin de la ligne, ou bien entre les caractères `/*` (ouvrant une section de commentaires) et `*/` (fermant une section de commentaires).
- En OCaml les écrit entre les caractères `(*` (ouvrant une section de commentaires) et `*)` (fermant une section de commentaires).

Compilation Action de compiler du code source grâce à un compilateur.
- En C, on utilise la commande `gcc`, avec
  - `-O1`, `-O2` ou `-O3` pour optimiser le code de manière plus ou moins agressive.
  - `-Wall` et `-Wextra` pour afficher les avertissements
  - `-Werror` pour transformer les avertissements en erreurs
  - `-fsanitize=` pour utiliser le sanitizer
  - `-lm` si l'en-tête `<math.h>` est utilisé
- En OCaml, ... à faire

↪ Sanitizer

Complément à deux Représentation des entiers signés utile pour réaliser des opérations dessus.
Étant donné une largeur ω et une suite de bits $(b_{\omega-1}, ..., b_0)$, sa valeur en complément à deux est

$$-b_{\omega-1}2^{\omega-1} + \sum_{i=0}^{\omega-2} b_i 2^i$$

Cela revient à multiplier par -1 le bit de poids fort.

Remarque :
- Le plus grand entier signé représentable est alors $2^{\omega-1} - 1$, et le plus petit entier signé représentable $-2^{\omega-1}$.
- L'intérêt de cette représentation, en notant $v_\omega(\mathrm{b})$ la valeur non signée et $vs_\omega(\mathrm{b})$ la valeur signée de la suite de bits b, est qu'on peut représenter les entiers non signés de manière usuelle :
$$v_\omega(b) \equiv vs_\omega(b) \bmod 2^w$$
- Pour obtenir le complément à deux d'un nombre négatif, on peut inverser les bits de l'écriture en binaire de sa valeur absolue, avant de lui additionner 1.
  Cela vient du point précédent et du fait que pour tout $x \geq 0$, en notant x_{inv} le nombre dont on a inversé les bits dans la représentation en complément à deux, on a
  $$x + x_{\text{inv}} = -1$$
  et donc avec b la représentation de x en complément à deux
  $$\begin{aligned} -x &\equiv 1 + x_{\text{inv}} && \bmod 2^\omega \\ &\equiv 1 + vs_\omega(b_{\text{inv}}) && \bmod 2^\omega \\ &\equiv 1 + v_\omega(b_{\text{inv}}) && \bmod 2^\omega \end{aligned}$$

Exemple : Avec $\omega = 4$, on a

$$7 = -0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$
$$-8 = -1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$$

On pose ensuite l'addition $7 + (-8)$:

$$\begin{array}{r} 0111 \\ +\ 1000 \\ \hline 1111 \end{array}$$

Or

$$-1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = -1 = 7 - 8$$

Complet
↪ Graphe complet

Complexité
↪ Classe de complexité

Complexité spatiale Quantité maximale de mémoire utilisée par un algorithme à un instant fixé au cours de son exécution.

Composante connexe

- Une composante connexe d'un graphe non orienté est un sous-graphe connexe ne faisant partie d'aucun sous-graphe connexe plus grand.
- Dans un graphe non orienté, la relation $\sim$ définie par

$$x \sim y \Leftrightarrow \text{il existe un chemin entre } x \text{ et } y$$

est une relation d'équivalence, et les composantes connexes sont les sous-graphes induit par les classes d'équivalence de la relation $\sim$.

Remarque : Intuitivement, une composante connexe d'un graphe G peut être représentée par un des sommets x de G, et regroupe alors tous les sommets y de G pour lesquels il existe un chemin allant de x vers y.
Un simple parcours du graphe permet d'obtenir ses composantes connexes.

Calcul des composantes connexes

Entrée : un graphe $G = (V, E)$
Sortie : une liste de composantes $C_1, ..., C_k$

```
vus ← ∅
composantes ← ()
fonction Ajouter(x, C)
    si x ∉ vus alors
        vus ← vus ∪ {x}
        pour y voisin de x faire
            Ajouter(y, C)
        C ← C, x
pour x ∈ V faire
    si x ∉ vus faire
        C ← ()
        Ajouter(x, C)
        composantes ← C, composantes
retourner composantes
```

Algorithme 6. – Calcul des composantes connexes

Composante fortement connexe La relation $\sim$ définie sur les sommets d'un graphe orienté G par

$$x \sim y \Leftrightarrow (x \text{ accessible depuis } y \text{ et } y \text{ accessible depuis } x)$$

est une relation d'équivalence. Les graphes induits par ses classes d'équivalence sont appelées les composantes fortement connexe de G.

Remarque : On utilise l'abréviation CFC pour *composante fortement connexe*, éventuellement SCC pour *strongly connected component*.
↪ Kosaraju (algorithme de)

Compression de chemin Lors d'une opération Find sur une structure UnionFind, on doit remonter la classe de l'élément jusqu'à sa racine pour pouvoir l'identifier et la renvoyer. On peut en profiter pour faire remonter tous les nœuds visités sans changer la complexité.

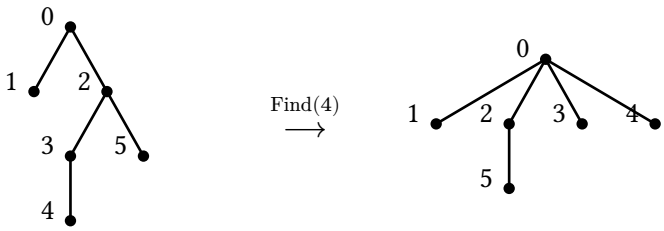


Figure 3. – Compression de chemin

↪ UnionFind

Compression sans perte Compression permettant une reconstruction exacte des données d'origine : aucune donnée n'est perdue ou dégradée.
C'est notamment utilisé pour la compression de texte. Le format PNG est par exemple un format de compression d'image sans perte, alors que JPG est un format de compression d'image avec perte.

Concaténation de langages Soient X, Y deux langages sur Σ. On note $X \cdot Y$ ou XY la concaténation de X et Y le langage

$$X \cdot Y = \{u \cdot v; u \in X \text{ et } v \in Y\}$$

Cette opération est associative.

Exemple :

$$\mathcal{L}(a^*) \cdot \mathcal{L}(b^*) = \mathcal{L}(a^*b^*)$$

Concaténation de mots La concaténation de u et v deux mots sur un même alphabet Σ correspond au fait de mettre bout à bout deux chaînes de caractères.

On note une telle opération $u \cdot v$, ou encore uv, définie par :
- $\varepsilon \cdot v = v$
- $u \cdot \varepsilon = u$
- $(u_1...u_n) \cdot (v_1...v_m) = u_1...u_nv_1...v_m$

Remarque : En OCaml, l'opérateur `^` permet de concaténer deux chaînes de caractères (`String.concat` est plus rapide).

Concurrence Un programme est concurrent s'il contient plusieurs fils (unités s'exécutant chacune séquentiellement) dont les exécutions peuvent être entrelacées ou simultanées.

Confusion
↪ Matrice de confusion

Conjonction *Remarque :* En C et OCaml, la conjonction logique est symbolisée par &&.

```c
if (a && b) {}
```

```ocaml
if a && b then ()
```

Connexe
↪ Graphe connexe, Graphe fortement connexe

Contrôle
↪ Graphe de flot de contrôle

Course critique On dit qu'il y a une course critique dans un programme dès que
- au moins deux fils peuvent accéder simultanément à un emplacement mémoire
- au moins l'un de ces accès est une écriture.

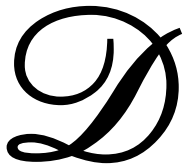
Remarque : En C, l'existence d'une course critique rend le programme UB (Undefined Behaviour, à éviter).

Couvrant
↪ Arbre couvrant

Creux
↪ Graphe creux

Cycle Chemin simple non réduit à un sommet, dont le premier et le dernier sommet sont confondus.

Cycle eulérien Cycle d'un graphe parcourant chaque arête du graphe exactement une fois.



DAG
↪ Graphe orienté acyclique

Décidable
↪ Problème décidable

Degré d'un nœud Dans un graphe non orienté, le degré d'un sommet x noté $\deg(x)$ est le nombre d'arêtes incidentes à x (de la forme $\{x, y\}$).
Dans un graphe orienté, on distingue
- le degré entrant noté $d_-(x)$ le nombre d'arcs où x est le successeur (de la forme (y, x))
- le degré sortant noté $d_+(x)$ le nombre d'arcs où x est le prédécesseur (de la forme (x, y))
- le degré total noté $\deg(x)$ la somme de son degré entrant et de son degré sortant

Remarque :
- Dans un graphe non orienté sans boucle, le degré d'un nœud est son nombre de voisins.
- Le degré sortant d'un sommet s'obtient en sommant la ligne correspondante de la matrice d'adjacence, le degré entrant en sommant la colonne.
- Dans un graphe orienté $G = (V, E)$, on a
$$|E| \leq |V| \cdot (|V| - 1)$$
- Dans un graphe non orienté,
$$|E| \leq \frac{|V| \cdot (|V| - 1)}{2}$$

Exemple : Dans un graphe cycle, le degré de tous les sommets vaut deux.
Dans un graphe complet K_n, le degré de tous les sommets vaut $n - 1$.

Dense
↪ Graphe dense

Dépassement arithmétique Situation lors de laquelle le résultat d'un calcul d'entiers dépasse l'entier maximal stockable. Le résultat peut alors devenir petit négatif.

Exemple :

```c
#include <stdio.h>
int main() {
  short x = 0;
  while (x >= 0) x += 1;
  printf("x = %hi\n", x);
}
```

Le code ci-dessus donne comme sortie `x = -32768`.

Déréférencement
↪ Opérateur de déréférencement

Dérivation Dans une grammaire sans contexte, pour $\alpha, \beta \in (\Sigma \cup V)^*$, on dit qu'il existe une dérivation de α en β, noté $\alpha \Rightarrow^* \beta$, s'il existe $n \geq 0$ tel que
$$\alpha \Rightarrow_n \beta$$

Dérivation de longueur n Pour $\alpha, \beta \in (\Sigma \cup V)^*$, on dit qu'il existe une dérivation de longueur $n \geq 0$ de α en β, noté $\alpha \Rightarrow_n \beta$, s'il existe des mots
$$\gamma_0 = \alpha, \gamma_1, ..., \gamma_n = \beta$$
tels que
$$\gamma_0 \Rightarrow \gamma_1 \Rightarrow ... \Rightarrow \gamma_n$$

Dérivation droite Dérivation constituée uniquement de dérivations immédiates droites.

Dérivation gauche Dérivation constituée uniquement de dérivations immédiates gauches.

Dérivation immédiate Pour $\alpha, \beta \in (\Sigma \cup V)^*$, on dit qu'il existe une dérivation immédiate de α en β, noté $\alpha \Rightarrow \beta$ s'il existe une règle $X \to \gamma$ telle que
$$\alpha = \alpha_1 X \alpha_2 \text{ et } \beta = \alpha_1 \gamma \alpha_2$$
où $\alpha_1, \alpha_2 \in (\Sigma \cup V)^*$.

Dérivation immédiate droite Dérivation de la forme $\alpha X u \Rightarrow \alpha \beta u$ où $u \in \Sigma^*$, $\alpha \in (\Sigma \cup V)^*$ et $X \to \beta$ est une règle de production de la grammaire.

Dérivation immédiate gauche Dérivation de la forme $uX\alpha \Rightarrow u\beta\alpha$ où $u \in \Sigma^*$, $\alpha \in (\Sigma \cup V)^*$ et $X \to \beta$ est une règle de production de la grammaire.

DFS
↪ Parcours en profondeur

Différence (égalité) En C, on utilise le symbole `!=` pour tester la différence, tandis qu'en OCaml on utilisera `<>`.

```c
if (a != b)  {}
```

```ocaml
if a <> b then ()
```

Disjonction *Remarque :* En C et OCaml, la disjonction logique est symbolisée par `||`.

```c
if (a || b) {}
```

```ocaml
if a || b then ()
```

Distance (graphe) Dans un graphe, la distance entre deux sommets a et b est la plus petite des longueurs des chemins reliant a et b.
Par convention, cette distance vaut ∞ si a et b ne sont reliés par aucun chemin.

Remarque : La longueur d'un chemin est
- son nombre d'arêtes si le graphe est non pondéré
- la somme des poids des arêtes du chemin si le graphe est pondéré

Distance d'édition
↪ Levenshtein (distance de)

Durabilité (base de données) Garantie qu'une requête, une fois exécutée, soit stockée de manière permanente.

Dynamique
↪ Programmation dynamique, Tableau dynamique



Effet de bord Pour une fonction, c'est le fait de modifier un état en dehors de son environnement local.

Égalité En C, on utilise le symbole == pour tester une égalité. En OCaml, le symbole = permet de tester l'égalité, tandis que == teste l'égalité physique, c'est-à-dire au niveau de la mémoire.

```c
if (a == b) {}
```
C

```ocaml
if a = b then ()
```
OCaml

Élément minimal Élément tel qu'aucun élément ne lui est inférieur, $x \in E$ est un élément minimal de $(E, \leq)$ si et seulement si :

$$\forall y \in E, \quad y \leq x \implies x = y$$

Élementaire
↪ Chemin élémentaire

Endianness
↪ Boutisme

Ensemble préordonné Ensemble E muni d'un préordre $\leq$, on le note $(E, \leq)$.

Entier En C, on distingue plusieurs types d'entiers provenant de `<stdint.h>` :
- les entiers signés `int8_t`, `int16_t`, `int32_t`, `int64_t`
- les entiers non signés `uint8_t`, `uint16_t`, `uint32_t`, `uint64_t`

Équivalence faible Deux grammaires G et G' sont faiblement équivalentes si

$$\mathcal{L}(G) = \mathcal{L}(G')$$

État nul En théorie des jeux, état terminal gagnant pour aucun des joueurs. Avec les notations usuelles, il s'agit de

$$V \setminus (T_0 \cup T_1)$$

État terminal En théorie des jeux, puits de l'arène du jeu considéré. Une fois dans un état terminal, le jeu est terminé, qu'un des joueurs ait gagné ou non.

Étiquette Donnée associée au nœud d'un arbre.

Étoile (lemme de)
↪ Pompage (lemme de)

Étoile de Kleene Si X est un langage sur Σ, l'étoile de Kleene de X est

$$X^* = \bigcup_{n \geq 0} X^n$$

Euler-Hierholzer (théorème de) Un graphe connexe non orienté admet
- un chemin eulérien si et seulement si ses sommets sont tous de degré pair sauf éventuellement un ou deux d'entre eux.
- un cycle eulérien si et seulement si tous ses sommets sont de degré pair.

L'algorithme d'Hierholzer permet de trouver un tel chemin.

Eulérien
↪ Graphe eulérien, Chaîne eulérienne

Exclusion mutuelle Se dit d'un verrou qui est toujours tenu par au plus un fil.

Exponentiation rapide Manière de calculer la puissance d'un nombre de manière récursive, en tirant partie des propriétés des puissances.
Le calcul se fait en $O(\log(|n|))$, où n est la puissance.

Exponentiation rapide

Entrée : un nombre a, un entier n
Sortie : a^n
fonction $\exp(a, n)$
  si $n = 0$ **alors**
    retourner 1
  si $n < 0$ **alors**
    retourner $\exp\left(\frac{1}{a}, -n\right)$
  si n est pair **alors**
    $p \leftarrow \exp\left(a, \frac{n}{2}\right)$
    retourner p^2
  sinon
    $p \leftarrow \exp\left(a, \frac{n-1}{2}\right)$
    retourner $a \times p^2$

Algorithme 7. – Exponentiation rapide

Exemple :

OCaml
```
let rec exp a n =
  if n = 0 then 1
  else if n < 0 then exp (1. /. a) (-n)
  else if n mod 2 = 0 then (
    let p = exp a (n/2) in
    p*p
  ) else (
    let p = exp a (n/2) in
    a*p*p
  )
```

$\mathscr{F}$

Facteur Si u et v sont deux mots définis sur un même alphabet Σ, u est un facteur de v s'il existe $w, z \in \Sigma^*$ tel que

$$v = w \cdot u \cdot z$$

Feuille Dans un arbre, nœud d'arité nulle, c'est-à-dire n'ayant pas d'enfant.

Fibonacci (suite de) Suite définie par

$$F_{n+2} = F_{n+1} + F_n \text{ où } \begin{cases} F_0 = 0 \\ F_1 = 1 \end{cases}$$

Remarque : On peut la calculer de manière itérative.

```c
int fibonacci(int n) {
  int* v = malloc((n+1) * sizeof(int));
  v[0] = 0;
  v[1] = 1;

  for (int i = 2; i <= n; i++) {
    v[i] = v[i-1] + v[i-2];
  }

  return v[n];
}
```

FIFO First in first out : les éléments ajoutés en premier sont les premiers à être retirés.

File Structure de données FIFO, analogue à une file d'attente. Il s'agit d'une séquence $x_{n-1}, \ldots, x_1, x_0$ de valeurs du même type, munie de deux opérations :
- Retirer un élément à droite de la file (défiler ou pop)
- Ajouter un élément à gauche de la file (enfiler ou push).

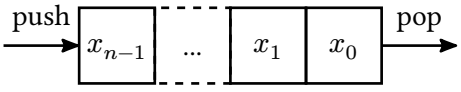
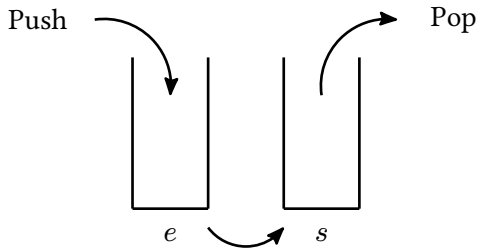


Figure 4. – Représentation d'une file

Implémentation :

- En OCaml, on utilise deux listes, une appelée e pour entrée et une appelée s pour sortie. On enfile dans e et défile dans s, si s est vide on renverse e dans s. La complexité est en $O(1)$ amortie et la capacité de la file est a priori non bornée.



Renversement de e dans s

Figure 5. – File en OCaml

- En C, on utilise un tableau circulaire, une variable pour la sortie des données dans le tableau et une pour la taille de la file. La complexité est en $O(1)$ et la capacité de la pile est fixée à la création.

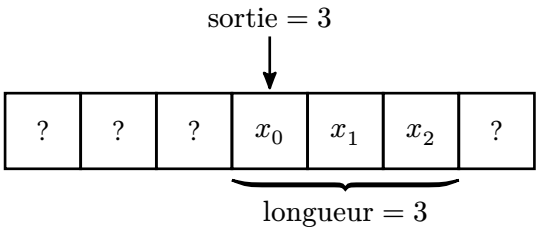


Figure 6. – File en C

Flot de contrôle
↪ Graphe de flot de contrôle

Flottant Norme d'écriture de nombres proche de la notation scientifique, représentant un nombre par :
- un signe, entre 1 et -1
- une mantisse, un entier codant le nombre sans virgule
- un exposant, codant la puissance de la base utilisée

Ce nombre est alors

$$\text{signe} \times \text{mantisse} \times \text{base}^{\text{exposant}}$$

Fonction de comparaison Fonction permettant de comparer deux éléments, tel que
- $\text{compare}(x, y) < 0$ si x doit être placé avant y
- $\text{compare}(x, y) > 0$ si x doit être placé après y

Ainsi, pour avoir un tri croissant on utilisera

$$(x, y) \mapsto x - y$$

Pour avoir un tri décroissant, on utilisera

$$(x, y) \mapsto y - x$$

Fonction récursive terminale Fonction récursive dont chaque appel fait au plus un appel récursif. Aucun calcul ne doit être fait à partir du résultat de cet appel.

Ces fonctions permettent d'économiser de l'espace mémoire par rapport à des fonctions récursives non terminales, car l'état de la fonction appelante n'a pas besoin d'être conservée sur la pile d'exécution.

Exemple :

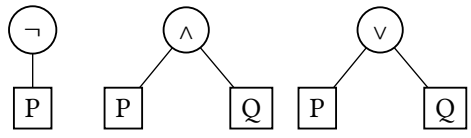
For
↪ Boucle for

Forêt Graphe acyclique. Un graphe est une forêt si et seulement si chacune de ses composantes connexes est un arbre.

Forme normale conjonctive Une forme normale conjonctive, abrégée FNC, est une conjonction de max-termes.
Toute formule propositionnelle P admet une unique forme normale conjonctive (à l'ordre près), et cette forme est dite canonique si chacun des max-termes contient toutes les variables de P.

Forme normale disjonctive Une forme normale disjonctive, abrégée FND, est une disjonction de min-termes.
Toute formule propositionnelle P admet une unique forme normale disjonctive (à l'ordre près), et cette forme est dite canonique si chacun des min-termes contient toutes les variables de P.

Formule propositionnelle Une formule propositionnelle est un élément de l'ensemble $\mathcal{P}$ des formules propositionnelles défini de manière inductive par :
- $\bot, \top \in \mathcal{P}$
- $\mathcal{V} \subset \mathcal{P}$ où $\mathcal{V}$ est un ensemble infini dénombrable dont les éléments sont les variables propositionnelles
- si $P, Q \in \mathcal{P}$, alors les arbres suivants sont dans $\mathcal{P}$



Remarque : La hauteur et la taille d'une formule propositionnelle correspondent respectivement à la hauteur et la taille de l'arbre correspondant.

Exemple : Le type d'un formule propositionnelle en OCaml, avec $\mathcal{V} = \mathbb{Z}$ est

OCaml
```
type prop =
  | Const of bool
  | Var of int
  | Non of prop
  | Et of prop * prop
  | Ou of prop * prop
```

Frères Enfants d'un même père.

Fusion
↪ Tri fusion

$\mathscr{G}$

Garbage collector
↪ Ramasse-miette

Glouton
↪ Algorithme glouton

Grammaire hors contexte Une grammaire hors contexte (aussi appelé grammaire sans contexte) est un quadruplet (Σ, V, R, S) où :
- Σ est un alphabet fini dont les éléments sont appelés symboles terminaux.
- V est un alphabet fini disjoint de Σ, ses éléments sont appelées symboles non terminaux ou variables.
- R est un ensemble fini de règles de production de la forme $X \to \alpha$ où $X \in V$ et $\alpha \in (V \cup \Sigma)^*$.
- S est un élément de V appelé symbole initial.

Grand O Si u et v sont deux suites à valeurs réelles positives, on dit que $u = O(v)$ si

$$\exists A > 0, n_0 \in \mathbb{N}, \forall n \geq n_0, A \cdot u_n \leq v_n$$

Autrement dit, u est majoré asymptotiquement par v, à un facteur près.

Graphe Un graphe G est un couple (V, E) où
- V est un ensemble non vide dont les éléments sont appelés sommets du graphe
- E est un ensemble de paires $\{x, y\}$ où x et y sont des éléments de V avec $x \neq y$. Ces paires sont les arêtes du graphe.

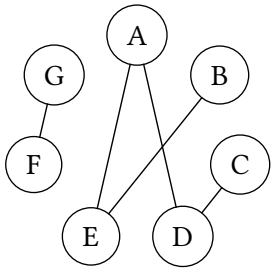


Figure 7. – Exemple de graphe

Remarque :
- Intuitivement, une arête $\{x, y\}$ permet de passer de x à y, et de y à x.
- On peut autoriser l'existence d'arête n'étant pas une paire mais un singleton, qu'on appelle alors boucle.
- On notera souvent xy au lieu de $\{x, y\}$ (on a alors $xy = yx$).
- Deux sommets x et y reliés pas une arête $e = \{x, y\}$ sont dits adjacents. On dit alors que e est incidente à x et y.
- Ceci est la définition d'un graphe non orienté. Pour un graphe orienté, on ne considère plus des paires mais des 2-uplets, où l'ordre a une importance et l'on parle alors d'arcs. Quand on parle d'un graphe tout court, on sous-entend souvent qu'il est non-orienté.

Exemple : Le graphe d'une carte ferroviaire a un sommet pour chaque gare, et une arête pour chaque voie ferrée reliant deux gares. Si toutes ces voies peuvent s'emprunter dans les deux sens, alors ce graphe est non-orienté. Dans le cas contraire, il est orienté.

Graphe biparti complet Le graphe biparti complet $K_{n,p}$ possède $n + p$ sommets, dont n noirs et p blancs, et une arête entre chacun des sommets noirs et chacun des sommets blancs.

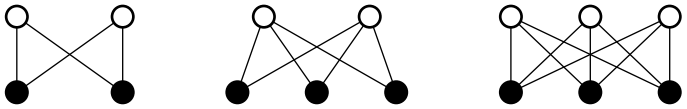


Figure 8. – Quelques graphes bipartis complets

Graphe chemin Le graphe chemin P_n possède n sommets numérotés $0, ..., n-1$ et une arête entre les sommets i et j si et seulement si $i = j + 1$.

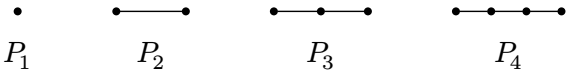


Figure 9. – Quelques graphes chemins

Graphe complet Le graphe complet K_n possède n sommets avec une arête entre chaque paire de sommets.

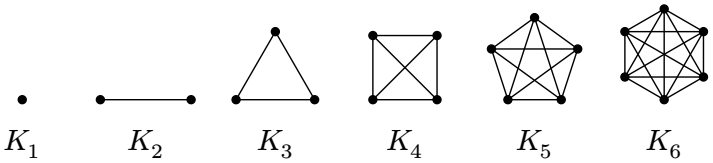


Figure 10. – Quelques graphes complets

Remarque : Pour $n \geq 5$, K_nn'est pas planaire.

Graphe connexe Un graphe non orienté G est dit connexe si pour tous sommets u et v de G, il existe un chemin entre u et v.

Graphe creux Graphe dans lequel le nombre d'arêtes est faible, c'est-à-dire vérifie

$$\frac{|E|}{|V|^2} \ll 1$$

↪ Degré d'un nœud

Graphe cycle Le graphe cycle C_n, avec $n \geq 3$, possède n sommets numérotés $0, ..., n-1$ et une arête entre i et j si et seulement si $i = j + 1 \bmod n$.

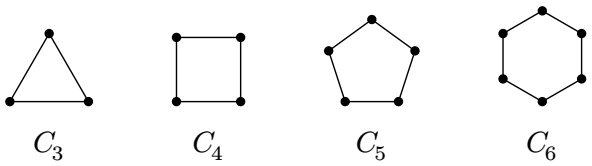


Figure 11. – Quelques graphes cycles

Graphe de flot de contrôle Graphe représentant les chemins pouvant être suivi par un programme durant son exécution.
On l'abrège parfois par CFG pour *control flow graph*.

Exemple :

```c
int i = 0;
while (i <= 10) {
  printf("Étape %d\n", i);
  if (i <= 5) {
    i += 2;
  } else {
    i++;
  }
}
printf("Enfin terminé \n");
```

Le programme écrit ci-dessus en C donne le graphe de flot de contrôle ci-desous.

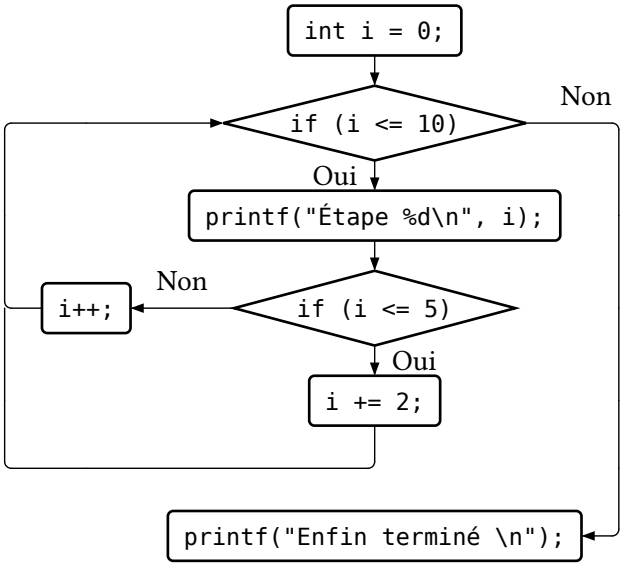


Figure 12. – Exemple de graphe de flot de contrôle

Graphe dense Graphe dans lequel le nombre d'arêtes est grand, avec par exemple $\frac{|E|}{|V|^2}$ proche de $\frac{1}{2}$ ou 1 selon que le graphe soit orienté ou non orienté respectivement.
C'est-à-dire que le nombre d'arêtes s'approche du nombre maximal d'arêtes (voir Degré d'un nœud).
↪ Degré d'un nœud

Graphe eulérien Graphe dans lequel il existe un cycle eulérien.

Graphe fortement connexe Un graphe orienté G est dit fortement connexe si, pour tout couple de sommets (u, v) de G, il existe un chemin de u vers v.
↪ Composante fortement connexe

Graphe orienté Un graphe orienté G est un couple (V, E) où

- V est un ensemble non vide dont les éléments sont appelés sommets du graphe
- E est un ensemble de couples (x, y) où x et y sont des éléments de V avec éventuellement $x \neq y$. Ces couples sont les arcs (parfois appelées arêtes) du graphe.

Remarque :

- Intuitivement, un arc (x, y) permet de passer de x à y, et non de y à x. On écrira souvent $x \to y$.
- On peut considérer les graphes non orientés comme des cas particuliers de graphes orientés, vérifiant
$$\forall (x, y) \in E, (y, x) \in E$$
- S'il y a un arc $x \to y$, on dira que x est un prédécesseur de y, et que y est un successeur de x.

Graphe pondéré Un graphe pondéré est un couple $G = (V, E, \rho)$ tel que

- (V, E) soit un graphe, orienté ou non
- $\rho : E \to \mathbb{R}$ associe à chaque arête ou arc un poids

ρ représente souvent une distance ou un temps, auquel cas ρ est à valeurs positives.

Graphe semi-eulérien Graphe dans lequel il existe un chemin eulérien.

Greedy algorithms
↪ Algorithme glouton

Gros boutisme
↪ Boutisme



Hachage
↪ Table de hachage

Hamiltonien
↪ Graphe hamiltonien

Hamming (distance de) Distance entre deux suites de même longueur n, donnée par

$$d_H(u,v) = \text{Card}(\{\ k\ ;\ u_k \neq v_k\ \})$$

Il s'agit du nombre d'indices en lesquels ces suites diffèrent.

Implémentation : Voici un programme calculant la distance de Hamming de deux listes en OCaml.

```ocaml
let rec dist u v =
  match u, v with
  | [], [] -> 0
  | x::u', y::v' ->
    (if x = y then 0 else 1) + dist u' v'
  | _, _ -> failwith "Longueurs différentes"
```

Hamming (nombre de) Un nombre de Hamming est un entier de la forme

$$n = 2^a \cdot 3^b \cdot 5^c \text{ où } a,b,c \in \mathbb{N}$$

Hauteur d'un arbre Profondeur maximale d'une feuille de l'arbre.

$\mathscr{I}$

Incident Une arête est incidente à deux sommets si elle relie ces deux sommets.

Insertion
↪ Tri par insertion, Insertion (arbre rouge-noir)

Invariant de boucle Utilisé pour prouver la correction d'un programme comportant une boucle, un invariant de boucle doit :
- être vraie avant la première itération
- rester vraie à la fin d'une itération si elle est vrai au début de celle-ci

Isolation (base de données) Garantie dans le cas où le système de gestion interagit avec plusieurs clients, que lorsque plusieurs requêtes sont traitées simultanément tout doit se passer comme si elles avaient été traitées successivement.

$\mathscr{J}$

Jeu J'ai perdu.

Jeu à deux joueurs Un jeu à deux joueurs est un quadruplet (G, s_0, T_0, T_1) où

- $G = (V_0 \sqcup V_1, E)$ est un graphe orienté biparti appelé arène. Les sommets de V_0 sont ceux contrôlés par le joueur 0, ceux de V_1 ceux contrôlés par le joueur 1.
- $s_0 \in V$ est le sommet initial
- T_0 et T_1 sont des parties disjointes de V ne contenant que des puits. Les sommets de T_0 sont ceux gagnants pour le joueur 0 et ceux de T_1 ceux gagnant pour le joueur 1.

On appelle les puits de V états terminaux, et les sommets ni dans T_0 ni dans T_1 sont appelés les états nuls.

Exemple : Le jeu du tic-tac-toe est un exemple de jeu à deux joueurs, comme les échecs, les dames, le puissance 4, Othello…

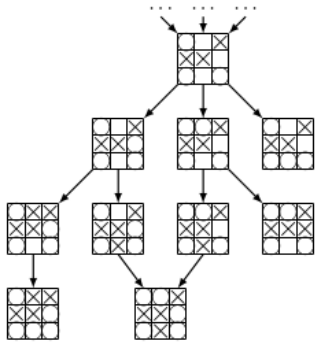


Figure 13. – Partie de l'arène du tic-tac-toe

↪ Chomp (jeu de)

$\mathscr{K}$

Karatsuba (algorithme de) Algorithme utilisé pour multiplier rapidement deux nombres, il tire parti de l'observation suivante : pour tout $a, b, c, d \in \mathbb{N}$ et pour tout $k \in \mathbb{N}$,

$$\big(a \cdot 2^k + b\big)\big(c \cdot 2^k + d\big) = ac \cdot 2^{2k} + (ad + bc) \cdot 2^k + bd$$

C'est un algorithme de type diviser pour régner, dont la complexité temporelle est en $O\big(n^{\log_2(3)}\big)$ (obtenue grâce au théorème maître), où n est la longueur des deux nombres. L'intérêt repose sur le fait qu'une multiplication naïve se fait en $O(n^2)$ tandis qu'une addition se fait en $O(n)$.

Implémentation :

```c
int karatsuba(int x, int y, int n) {
  // Cas de base
  if (n == 1) {
    if (x == 1 && y == 1) return 1;
    return 0;
  }

  int k = n >> 1;

  int a = x >> k;
  int b = x - (a << k);

  int c = y >> k;
  int d = y - (c << k);

  return
    (karatsuba(a, c, k) << (k + k))
    + ((karatsuba(a, d, k) + karatsuba(b, c, k)) << k)
    + karatsuba(b, d, k);
}
```

↪ Master theorem

Kleene
↪ Étoile de Kleene

Kosaraju (algorithme de) Algorithme calculant les composantes fortement connexes d'un graphe orienté.

Kruskal (algorithme de) Algorithme calculant un arbre couvrant minimal d'un graphe connexe.

Algorithme de Kruskal

Entrées : un graphe connexe $G = (V, E)$
Sorties : un arbre couvrant T de G

$T \leftarrow (V, \varnothing)$
pour $xy \in E$, dans l'ordre **faire**
  si $C_{T(x)} \neq C_{T(y)}$ **alors**
    $T \leftarrow T + xy$
renvoyer T

Algorithme 8. – Algorithme de Kruskal

Remarque : Pour prouver la correction de cet algorithme, montrer que peu importe les choix faits lors des i premiers tours de boucle, le graphe acyclique obtenu peut être complété de manière à obtenir un arbre couvrant minimal.
↪ Composante connexe

$\mathscr{L}$

Langage Un langage est une partie de Σ^*.

Remarque : On note Σ^n le langage des mots de longueur n.

Exemple : $\mathcal{L} = \{a^n \mid n \geq 0\}$ pour $\Sigma = \{a, b\}$.

Langage algébrique Langage engendré par une grammaire sans contexte.

Remarque : Les langages rationnels sont stables par concaténation, union et étoile de Kleene.

Exemple : $\{a^n b^n \mid n \geq 0\}$ est algébrique, dont la grammaire est

$$V = \{S\}$$
$$\Sigma = \{a, b\}$$
$$R = \{S \to \varepsilon, S \to aSb\}$$

Langage engendré par une grammaire Le langage engendré par la grammaire $G = (\Sigma, V, R, S)$ est

$$\mathcal{L}(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$

↪ Langage algébrique

Langage rationnel Langage décrit par une expression régulière : il existe une expression régulière e telle que

$$X = \mathcal{L}(e)$$

Remarque : Les langages rationnels sont stables par concaténation, union, passage au complémentaire et étoile de Kleene.

Exemple : $\{a^n \mid n \geq 0\} = \mathcal{L}(a^*)$ est rationnel.

Langage régulier
↪ Langage rationnel

Lexicographique
↪ Ordre lexicographique

LIFO Last in first out : les éléments ajoutés en dernier sont les permiers à être retirés.

Liste doublement chaînée Structure de données composées de nœuds, chacun muni d'une étiquette et d'un accès à son successeur et à son prédécesseur s'ils existent.



Figure 14. – Liste doublement chaînée correspondant à [1; 2; 3]

Liste simplement chaînée Structure de données composées de nœuds, chacun muni d'une étiquette et d'un accès à son successeur s'il existe.

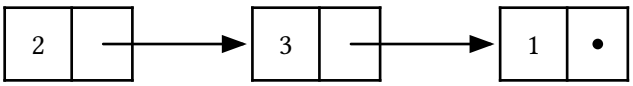


Figure 15. – Liste simplement chaînée correspondant à [1; 2; 3]

Littéral Formule constituée uniquement d'une variable ou de la négation d'une variable :

$$x \text{ ou } \neg x$$

Longueur
↪ Longueur d'un mot

Longueur d'un mot Nombre de lettres dans ce mot :

$$\begin{cases} |\varepsilon| = 0 \\ |m_1 ... m_n| = n \end{cases}$$

$\mathscr{M}$

Machine de Turing Il existe plusieurs définitions similaires, nous en proposons une ici.
Intuitivement, une machine de Turing est un automate particulier travaillant sur une bande au moyen d'une tête de lecture et écriture se déplaçant au plus d'une case vers la droite ou la gauche à chaque étape de calcul.

Formellement, une machine de Turing M est un tuple $(Q, q_0, \Sigma, \delta, \$, B)$ où
- Q est un nombre fini d'états
- $q_0 \in Q$ est l'état initial
- Σ est un alphabet fini de cardinal supérieur à 3 (car contenant $\$ \neq B$ ainsi qu'au moins un autre symbole distinct)
- $\delta : Q \times \Sigma \to Q \cup \{\text{accept}, \text{reject}\} \times \Sigma \times \{\leftarrow, \rightarrow, \downarrow\}$ est la fonction de transition (où accept $\neq$ reject). δ doit vérifier
  - $\forall q \in Q, \delta(q, \$) = (q', \$, \rightarrow)$ on n'enlève jamais le \$, et quand on se trouve à sa position on repart.
  - $\forall q \in Q, \forall a \in \Sigma - \{\$\}, \delta(q, a) = (q', b, d)$ où $b \neq \$$, on n'écrit jamais le symbole \$ si on n'est pas en début de la bande
- $\$, B \in \Sigma$ sont des symboles spéciaux distincts, \$ marquant le début de la bande et B le symbole blanc.

Machine universelle Une machine universelle, aussi appelée machine de Turing universelle, est une machine de Turing capable de simuler n'importe quelle machine de Turing sur n'importe quelle entrée.
Cette machine M calcule la fonction

$$f : (\text{code de } M', \text{ruban d'entrée}) \mapsto \text{résultat de } M'$$

Où le résultat de M' est le résultat de l'exécution de cette machine de Turing M' sur le ruban d'entrée donné en argument.

Malloc En C, la fonction `malloc` permet de réserver un zone mémoire. Elle est déclarée dans l'en-tête `<stdlib.h>`.

```c
void *malloc(size_t nb_octets);
```

Manhattan (distance de) Distance consistant à additionner les différences d'abscisse et d'ordonnée de deux points : c'est la distance associée à la norme un.

$$d_M(a, b) = |a_x - b_x| + |a_y - b_y|$$

On peut la voir, sur une grille, comme le nombre de déplacements élémentaires verticaux et horizontaux à réaliser pour aller du point a au point b.

Implémentation :

```ocaml
let distance a b =
  let ax, ay = a in
  let bx, by = b in
  abs (ax - bx) + abs (ay - by)
```

Mantisse Aussi appelé significande, partie d'un flottant stockant la valeur absolue du nombre multiplié par un facteur (la base à une certaine puissance)
↪ Flottant

Master theorem Soit $a \geq 1$ et $b > 1$ fixés, soit $f : \mathbb{N} \to \mathbb{N}$ une fonction et $T(n)$ définie sur les entier positifs par la formule de récurrence

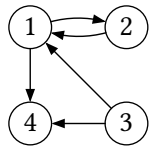
$$T(n) = f(n) + aT\left(\frac{n}{b}\right)$$

Alors $T(n)$ peut être borné asymptotiquement par
- si $f(n) = O\left(n^{\log_b(a-\varepsilon)}\right)$ où $\varepsilon > 0$, alors
$$T(n) = O\left(n^{\log_b(a)}\right)$$
- si $f(n) = \Theta\left(n^{\log_b(a)}\right)$ alors
$$T(n) = \Theta\left(n^{\log_b(a)}\right)\log(n)$$
- si $f(n) = \Omega\left(n^{\log_b(a+\varepsilon)}\right)$ où $\varepsilon > 0$ et si $af\left(\frac{n}{b}\right) \leq cf(n)$ où $c < 1$ et n grand, alors
$$T(n) = \Theta(f(n))$$

Ce résultat est particulièrement utile lorsqu'il s'agit de déterminer la complexité d'un algorithme de type « diviser pour régner », comme par exemple l'algorithme de Karatsuba.
↪ Karatsuba (algorithme de)

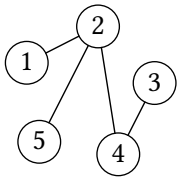
Matrice d'adjacence La matrice d'adjacence d'un graphe $G = (V, E)$ où $V = \{x_1, ..., x_n\}$ est la matrice $M = (m_{i,j}) \in \mathcal{M}_n$ définie par

$$m_{i,j} = \begin{cases} 1 \text{ si il y a une arête de } x_i \text{ vers } x_j \\ 0 \text{ sinon} \end{cases}$$



$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 16. – Matrice d'adjacence d'un graphe orienté



$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Figure 17. – Matrice d'adjacence d'un graphe non orienté

Remarque :
- M est symétrique si et seulement si le graphe est non orienté.
- Si on interdit les boucles, il n'y a que des zéros sur la diagonale.

Matrice de confusion Matrice utilisée pour déterminer la justesse d'un algorithme de classification.
Si on note $c(x)$ la classe d'un échantillon x et $f(x)$ sa classe inférée par un algorithme de classification f, alors la matrice de confusion de f pour un ensemble d'échantillons X est M définie par

$$M_{i,j} = \text{Card}(\{x \in X \mid c(x) = i \text{ et } f(x) = j\})$$

On remarque que
- M est diagonale si et seulement si tous les échantillons $x \in X$ ont bien été classés par f
- le pourcentage d'échantillons mal classifiés (le taux d'erreur global) est égal à la somme des coefficients hors diagonale divisé par la somme de tous les coefficients.

Max-terme Disjonction de littéraux dans laquelle chacune des variables apparaît une fois.

Exemple :

$$x \vee y \vee \neg z$$

est un max-terme.

$$a \vee (b \wedge c)$$

n'est pas un max-terme.

Merge sort
↪ Tri fusion

Min-terme Conjonctions de littéraux dans laquelle chacune des variables apparaît une fois.

Exemple : $x \wedge (\neg y) \wedge z$ est un min-terme, mais $a \wedge (b \vee c)$ n'est pas un min-terme.

Minimum Plus petit élément d'un ensemble préordonné, $x \in E$ est un minimum de $(E, \leq)$ si et seulement si :

$$\forall y \in E, \quad x \leq y$$

Mot Le mot vide ε ou une suite finie $m_1 ... m_n$ d'éléments de Σ pour $n \geq 1$.

On note Σ^* l'ensemble des mots sur Σ.

Exemple : abba est un mot de $\Sigma = \{a, b\}$.

Mot généré Une grammaire engendre un mot $w \in \Sigma^*$ si $S \Rightarrow^* w$.

Multi-ensemble Collection d'objet dans lequel l'ordre des éléments n'a pas d'importance mais le nombre d'occurrences compte.
On le note

$$\text{ms}(x_1, ..., x_n)$$

Mutex Un mutex m, ou verrou, est une primitive de synchronisation dotée au moins de l'interface suivante :
- une fonction CreateLock() renvoyant un nouveau mutex initialement libre
- une fonction Lock(m) ne renvoyant rien, qui permet d'acquérir le verrou. Un appel à cette fonction termine au bout d'un temps indéterminé, éventuellement infini, mais quand l'appel termine le fil ayant effectué l'appel possède le verrou.
- une fonction Unlock(m) ne renvoyant rien, qui permet de libérer le verrou.

À tout moment, le verrou est tenu par au plus un fil.

Remarque :
- Le terme mutex vient de *MUTual EXclusion.*

Implémentation : On peut implémenter un Mutex avec à l'algorithme de Peterson.
↪ Peterson (algorithme de)



Négation *Remarque :* En C, la disjonction logique est symbolisée par `!`, en OCaml on utilise le mot-clé `not`.

```c
if (!a) {}
```

```ocaml
if not a then ()
```

Nombre chromatique Le nombre chromatique d'un graphe G est l'entier k le plus petit tel que G soit k-coloriable.

Nœud Dans un arbre, un nœud est un sommet.

Nœud interne Nœud n'étant pas une feuille, d'arité non nulle.



OCaml Le langage OCaml est un langage de programmation fonctionnel maintenu par l'Inria.

Octet Suite de 8 bits codant une information.

Exemple : Sur un octet, on peut stocker un nombre dans
- $[\![0, 255]\!]$ dans le cas d'un entier non signé.
- $[\![-128, 127]\!]$ dans le cas d'un entier signé.

Omega Si u et v sont deux suites à valeurs réelles positives, on dit que $u = \Omega(v)$ si

$$\exists A > 0, n_0 \in \mathbb{N}, \forall n \geq n_0, A \cdot v_n \leq u_n$$

Autrement dit u majore asymptotiquement v, à un facteur près.

Option
↪ Type option

Ordre
↪ Relation d'ordre

Ordre bien fondé Ordre sur un ensemble E pour lequel toute partie non vide admet un élément minimal.

Cela équivaut à l'absence de suites infinis strictement décroissantes.

Exemple : La relation d'inclusion sur les parties finies de $\mathbb{N}$ est bien fondée, ce n'est pas le cas de la relation de supériorité sur $\mathbb{N}$.

Ordre d'un graphe Nombre de sommets de ce graphe.

Ordre lexicographique Ordre sur des suites finies d'un produit cartésien d'ensembles ordonnés. Il s'agit en quelque sorte de l'ordre du dictionnaire : avec $(E_1, <), ..., (E_p, <)$ des ensembles ordonnés, l'ordre lexicographique sur $E_1 \times \cdots \times E_p$ est défini par

$$(a_1, ..., a_p) < (b_1, ..., b_p)$$
$$\Leftrightarrow$$
$$\exists k \in [\![1, p]\!], \forall i < k, a_i = b_i \text{ et } a_k < b_k$$

L'ordre lexicographique sur $E_1 \times \cdots \times E_p$ est bien fondé.

Exemple : L'ordre lexicographique sur $\mathbb{N}^p$, ou bien sur $\{a, b, ..., z\}^{10}$.

Remarque : On peut l'utiliser pour prouver la terminaison de boucles imbriquées, avec des variants de boucle sur $\mathbb{N}^p$.

Ordre partiel Un ordre $\leq$ sur E est partiel lorsqu'il existe au moins une paire d'éléments non comparables :

$$\exists (x, y) \in E \quad (x \not\leq y) \wedge (y \not\leq x)$$

Ordre topologique Un ordre topologique sur un graphe orienté G est un ordre total sur les sommets de G tel que, s'il y a un arc de u vers v, alors $u \prec v$ (au sens strict).

Ordre total Un ordre $\leq$ sur E est total lorsque tous les éléments sont comparables :

$$\forall x, y \in E \quad (x \leq y) \vee (y \leq x)$$

Orienté
↪ Graphe orienté

Overflow
↪ Dépassement arithmétique, Soupassement arithmétique



Paire Ensemble de cardinal égal à deux.

Exemple : $\{8, 6\}$ et $\{6, 1\}$ sont des paires. $\{8, 6, 1\}$ n'en est pas une.

Palindrome Mot étant égal à son miroir, si $u = u_1 ... u_n$ est un palindrome :

$$\forall i \in [\![1; n]\!] \quad u_i = u_{n+1-i}$$

Exemple : *bob* et *kayak* sont des palindromes, contrairement à *bloubiboulga*.

Remarque : Vérifier qu'un mot est un palindrome se fait aisément en temps linéaire.

```c
bool est_palindrome(char* str) {
  int n = strlen(str);

  for (int i = 0; i < n / 2; i++) {
    if (str[i] != str[n - 1 - i]) {
      return false;
    }
  }

  return true;
}
```

Parallélisme Fait d'exploiter la concurrence et la présence de plusieurs processeurs (unités d'exécution) pour accélérer un calcul ou programme.
↪ Concurrence

Parcours eulérien
↪ Chaîne eulérienne

Partie Une partie d'un jeu est un chemin de l'état initial vers un état terminal dans le graphe du jeu.
On parle de partie infinie si le chemin est infini (présence d'un cycle par exemple).

Permutation Soit E un ensemble, une permutation de E est une bijection de E dans E. L'ensemble des permutations de E muni de la composition $\circ$ est un groupe, noté S_E ou $\mathfrak{S}(E)$.

Lorsque $E = [\![1, n]\!]$, on l'appelle *groupe symétrique*, noté S_n ou $\mathfrak{S}_n$.

Peterson (algorithme de) Implémentation d'un mutex garantissant l'exclusion mutuelle, l'absence d'interblocage et l'absence de famine.

Algorithme de Peterson

fonction CreateLock()
  renvoyer $\{\text{turn} = 0; \text{want} = [\text{false}, \text{false}]\}$
fonction Unlock(m, t)
  m.want $\leftarrow$ false
fonction Lock(m, t)
  m.want $\leftarrow$ true
  m.turn $\leftarrow 1 - t$
  tant que m.turn $= 1 - t$ et m.want$[1 - t]$ **faire**
    Rien

Algorithme 9. – Algorithme de Peterson

Remarque :
- Suivre ce pseudo-code pour implémenter cet algorithme en C ne fonctionne pas car il peut y avoir un interblocage : cela est dû au fait que le processeur peut inverser l'ordre de deux instructions si on ne prend pas de précautions.
- Dans la pratique n'écrivez pas votre propre mutex, utilisez celui de la bibliothèque standard.

Petit boutisme
↪ Boutisme

Pile (Booléen) Structure de données LIFO, analogue à une pile d'assiettes. Il s'agit d'une séquence $x_0, x_1, ..., x_{n-1}$ de valeurs du même type, munie de deux opérations :
- Retirer un élément au sommet de la pile (dépiler ou pop)
- Ajouter un élément au sommet de la pile (empiler ou push).

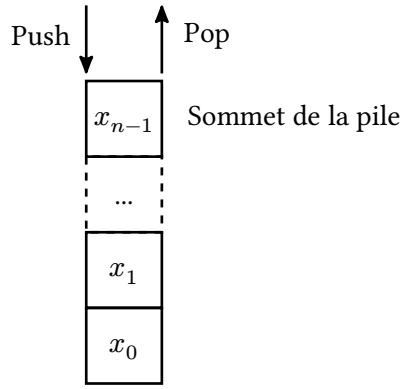


Figure 18. – Représentation d'une pile

Implémentation :

- En OCaml, on utilise simplement une liste chaînée. La complexité est en $O(1)$ et la capacité de la pile est a priori non bornée.
- En C, on utilise un tableau ainsi qu'une variable pour la taille de la pile. La complexité est en $O(1)$ et la capacité de la pile est fixée à la création.

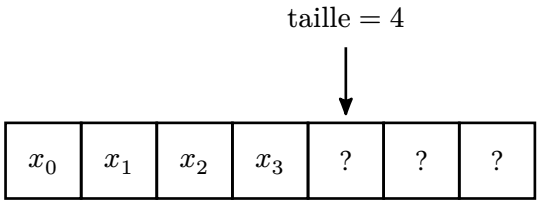


Figure 19. – Pile en C

Pile d'exécution Pile servant à stocker les informations des appels de fonction.

Poids d'un sous-graphe Si H est un sous-graphe de G, le poids de H, noté $\rho(H)$, est la somme des poids des arêtes de H.

Pondéré
↪ Graphe pondéré

Préfixe Si u et v sont deux mots définis sur un même alphabet Σ, u est un préfixe de v s'il existe $w \in \Sigma^*$ tel que

$$v = u \cdot w$$

Préordre Relation réflexive et transitive.

Principe d'optimalité de Bellman Principe stipulant qu'une solution globale d'un problème décomposé en plusieurs sous-problèmes est optimale si et seulement si chacun de ces sous-problème est résolu de manière optimale.

Exemple : Supposons qu'on cherche le chemin de A vers C passant par B le plus court.

$$A \xrightarrow{\gamma} B \xrightarrow{\delta} C$$

On décompose ce problème en la recherche d'un chemin γ de A vers B et d'un chemin δ de B vers C, tous deux les plus courts possibles.
Le chemin global est optimal si et seulement si γ et δ sont optimaux.
↪ Bellman-Ford (algorithme de)

Problème de l'arrêt Problème de décision suivant.

- **Instance :** le code source d'un programme f et la description d'un argument x pour ce programme
- **Question :** est-ce que le programme f s'arrête en temps fini sur l'argument x ?

Remarque : Ce problème est indécidable. On peut considérer la terminaison de ce programme `paradoxe.ml` pour le montrer.

```OCaml
let arret f x = ... (* code source de la fonction arret *)
let lire_fichier x = ... (* code lisant un fichier *)

let () =
  let code_source = lire_fichier "paradoxe.ml" in
  if arret code_source "" then (while true do () done)
```

Problème décidable Un problème de décision est décidable s'il existe un algorithme qui

- prend en entrée une instant x du problème
- répond « Oui » en temps fini si x est une instance positive
- répond « Non » en temps fini si x est une instance négative

Sinon, le problème est dit indécidable.

Exemple :

- Le problème SAT est décidable.
- Le problème de l'arrêt est indécidable.

↪ Problème de l'arrêt

Profondeur d'un nœud Longueur du chemin reliant la racine à ce nœud. Cela revient à compter le nombre d'ancêtres de ce nœud.

Programmation dynamique Fait d'échanger de la complexité spatiale contre de la complexité temporelle.

Puissance d'un langage Un langage X à la puissance n est le langage contenant exactement les mots formés de la concaténation de n mots de X.

$$X^n = X \cdot X \cdots X$$

Puissance d'un mot Pour $u \in \Sigma^*$ et $n \geq 0$, on définit u^n par :

$$\begin{cases} u^0 = \varepsilon \\ u^{n+1} = u \cdot u^n \end{cases}$$

Exemple : $(ab)^3 = ababab$.



Queue
↪ File

Quicksort
↪ Tri rapide

$\mathscr{R}$

Racine Dans un arbre, nœud initial n'ayant pas de père.

Radix
↪ Tri radix

Ramasse-miette Système de gestion automatique de la mémoire, recyclant la mémoire allouée mais inutilisée.

Rapide
↪ Tri rapide

Recherche dichotomique Algorithme de recherche permettant de déterminer la présence d'un élément dans un tableau trié.
Il consiste à comparer l'élément recherché au milieu du tableau, puis à poursuivre la recherche dans la moitié droite ou gauche du tableau selon si l'élément recherché est plus petit ou plus grand.

Implémentation : Dans le cas d'un tableau trié par ordre croissant, voici un exemple d'implémentation retournant l'indice de la position de la valeur recherchée.

```c
int dichotomie(int* arr, int len, int x)
{
  int a = 0;// Borne inférieure inclue
  int b = len - 1;// Borne supérieure inclue

  while (a <= b) {
    int milieu = (a + b)/2;
    int valeur = arr[milieu];

    if (x == valeur) {
      return milieu;
    } else if (x < valeur) {
      b = milieu - 1;
    } else {
      a = milieu + 1;
    }
  }

  return -1;
}
```

Redondant Qui se répète.

Régner
↪ Diviser pour régner

Relation Une relation entre deux ensembles E et F est une partie Γ de $E \times F$. Deux éléments $(x, y) \in E \times F$ sont en relation si (x, y) appartient à Γ.

Dans le cas $E = F$, on parle simplement de relation sur E.

Exemple : L'égalité modulo 861 est une relation sur $\mathbb{Z}$, avec

$$\Gamma = \{(a, a + 861k) \mid a, k \in \mathbb{Z}\}$$

Relation antisymétrique Relation sur E où deux éléments en relation l'un avec l'autre sont égaux :

$$\forall x, y \in E \quad (x \mathcal{R} y) \wedge (y \mathcal{R} x) \implies x = y$$

C'est-à-dire :

$$\forall x, y \in E \quad (x, y) \in \Gamma \wedge (y, x) \in \Gamma \implies x = y$$

Exemple : La relation de supériorité sur $\mathbb{N}$ est antisymétrique.

Relation asymétrique Relation sur E où deux éléments ne peuvent pas être mutuellement en relation :

$$\forall x, y \in E \quad x \mathcal{R} y \implies \neg(y \mathcal{R} x)$$

Ou encore :

$$\forall x, y \in E \quad (x, y) \in \Gamma \implies (y, x) \notin \Gamma$$

Exemple : La relation de supériorité stricte sur $\mathbb{N}$ est antisymétrique.

Relation d'équivalence Relation réflexive, transitive et symétrique.

Exemple : L'égalité est une relation d'équivalence.

Relation d'ordre Relation réflexive, transitive et antisymétrique.

Relation réflexive Relation sur E où tout élément est en relation avec lui-même :

$$\forall x \in E \quad x \mathcal{R} x$$

Ou encore si :

$$\forall x \in E \quad (x, x) \in \Gamma$$

Exemple : L'égalité est une relation réflexive.

Relation symétrique Relation sur E où deux éléments sont soit tous deux en relation avec l'autre, soit tout deux non en relation avec l'autre :

$$\forall x, y \in E \quad x\mathcal{R}y \iff y\mathcal{R}x$$

Cela revient à :

$$\forall x \in E \quad (x, y) \in \Gamma \iff (y, x) \in \Gamma$$

Exemple : L'égalité et la différence sont des relations symétriques.

Relation transitive Relation sur E où une suite de relations entre plusieurs éléments aboutie à une relation entre le premier et le dernier élément :

$$\forall x, y, z \in E \quad (x\mathcal{R}y) \land (y\mathcal{R}z) \implies x\mathcal{R}z$$

C'est-à-dire :

$$\forall x, y, z \in E \quad (x, y) \in E \land (y, z) \in \Gamma \implies (x, z) \in \Gamma$$

Exemple : L'inclusion, et l'inégalité stricte sur $\mathbb{N}$ sont des relations transitives.

Requête (base de données) Réaliser une requête sur une base de données désigne le fait de l'interroger (souvent au travers du langage SQL) pour utiliser ensuite son résultat.

Exemple : Voici un exemple de requête SQL.

```SQL
1 SELECT nom, code FROM departements;
```

Dont le résultat serait de la forme

nom	code
Finistère	29
Paris	75
Rhône	69

↪ SQL



Sac à dos (problème du) Problème classique NP-complet, consistant à optimiser le remplissage d'un sac à dos.

On dispose d'un sac à dos dont la charge utile est limitée à $p_{\max}$, ainsi que de n objets $a_1, ..., a_n$. Chaque objet est caractérisé par un poids p_i et une valeur v_i positive, le but est de maximiser la valeur contenue dans le sac.

On peut le résoudre avec de la programmation dynamique ou un algorithme glouton.

Exemple : Calculons la valeur maximale $v_{\max}$ par programmation dynamique.
Si on pose $f(k,p)$ la valeur optimale obtenue pour les objets $a_1, ..., a_k$ avec un poids p, on a:

- si $k = 0$ on ne peut prendre aucun objet et
$$f(0,p) = 0$$
- si $k > 0$, alors on considère le k-ième objet et
  - si $p_k > p$ on ne peut pas le prendre donc
  $$f(k,p) = f(k-1,p)$$
  - si $p_k \leq p$, on a le choix de le prendre donc
  $$f(k,p) = \max(f(k-1,p), v_k + f(k-1,p-p_k))$$

On peut donc calculer de cette manière

$$v_{\max} = f(n, p_{\max})$$

OCaml
```
let rec sac_a_dos p v pmax =
  match p, v with
  | [], [] -> 0
  | p_k::p', v_k::v' ->
    if p_k > pmax then sac_a_dos p' v' pmax
    else max
      (sac_a_dos p' v' pmax)
      (v_k + sac_a_dos p' v' (pmax - p_k))
  | _, _ -> failwith "Listes de taille différente"
```

Sanitizer Outil de programmation détectant les comportements indéfinis (`undefined behaviour`) et erreurs de mémoire notamment.

En C, on peut le configurer avec le flag `-fsanitize`, dont les arguments classiques sont

- `address` pour détecter les erreurs de mémoire avec `AddressSanitizer`)
- `undefined` pour modifier le programme à la compilation pour détecter les `undefined behaviour` à l'exécution

Satisfiable Une formule propositionnelle P est dite satisfiable s'il existe une valuation v telle que

$$\tilde{v}(P) = V$$

On note $\text{Mod}(P)$ l'ensemble des valuations qui satisfont P.

Sémantique La sémantique d'un algorithme A est la fonction partielle $f : \text{string} \to \text{string}$ définie sur les x pour lesquels $A(x)$ termine et alors

$$f(x) = A(x)$$

Sémaphore Primitive de synchronisation. Un sémaphore est une variable entière positive ou nulle et possèdant deux opérations atomiques :

- POST incrémente la variable
- WAIT tente de décrémenter la variable. Si cela la rendrait strictement négative, le fil ayant effectué l'appel est mis en attente

Implémentation : On peut implémenter un sémaphore avec l'algorithme de la boulangerie de Lamport.

Remarque :

- Un sémaphore représente donc typiquement une quantité de ressources disponibles.
- Il sert à garantir un comportement « premier arrivé, premier servi ».
- L'attente qu'une ressource soit libérée n'est pas une attente active : le fil est réellement mis en attente pour être réveillé le moment venu.

Exemple : En OCaml, on peut utiliser le module `Semaphore`, plus particulièrement `Semaphore.Counting`.
↪ Lamport (algorithme de la boulangerie de)

Séquence Collection d'éléments dans laquelle l'ordre et le nombre d'occurrences des éléments compte.
On notera

- la séquence $s = (s_0, ..., s_{n-1})$
- $|s| = n$ la longueur de la séquence
- $|s|_x$ le nombre d'occurrences de x dans s
- $s[d:f] = (s_d, ..., s_{f-1})$ les éléments de la séquence de l'indice de début d inclus à l'indice de fin f exclus, avec $0 \leq d \leq f \leq n$.
- $s[:f] = s[0:f]$ les éléments de la séquence strictement avant l'indice f

- $s[d:] = s[d:n]$ les éléments de la séquence après l'indice d

Séquent Couple (Γ, A) où Γ est un ensemble de formules propositionnelles et A une formule propositionnelle, notée

$$\Gamma \vdash A$$

On l'interprète comme « à partir des hypothèse Γ on en déduit A ».

Simple
↪ Chemin simple

Sommet Dans un graphe $G = (V, E)$, un sommet est un élément de V.

Soupassement arithmétique Situation lors de laquelle le résultat d'un calcul d'entiers devient plus petit que l'entier minimum stockable.

Exemple :

```c
#include <stdio.h>
int main() {
  short x = 0;
  while (x <= 0) x -= 1;
  printf("x = %hi\n", x);
}
```

Le code ci-dessus donne comme sortie `x = 32767`.
↪ Dépassement arithmétique

Sous-graphe Un sous-graphe d'un graphe $G = (V, E)$ est le graphe $G' = (V', E')$ défini par
- $V' \subset V$
- $E' \subset E$, et également $E' \subset V'^2$ (définition d'un graphe)

Remarque : Intuitivement, un sous-graphe de G est un sous-ensemble des sommets de G et une partie des arêtes reliant ces sommets entre eux.
↪ Graphe

Sous-graphe induit Un sous-graphe induit de $G = (V, E)$ est un sous-graphe $G' = (V', E')$ de G tel que E' est exactement l'ensemble des arêtes de G reliant deux sommets de V'.

Remarque : Intuitivement, un sous-graphe induit de G est un sous-ensemble des sommets de G et toutes les arêtes reliant ces sommets entre eux.

Sous-mot Soit $u, v \in \Sigma^*$, avec

$$u = u_1 ... u_n \text{ où } n = |u|$$

On dit que u est un sous-mot de v s'il existe des mots $t_0, ..., t_n$ tels que

$$v = t_0 \cdot u_1 \cdot t_1 \cdot u_2 \cdots t_{n-1} \cdot u_n \cdot t_n$$

Autrement dit u est composé des lettres de v en conservant leur ordre.

Spatial
↪ Complexité spatiale

SQL Langage informatique permettant d'exploiter des bases de données relationnelles. Ce sigle correspond à *Structured Query Language*.

Stable
↪ Tri stable

Stack
↪ Pile (Booléen)

Stratégie Une stratégie pour un joueur i est une application des sommets non terminaux contrôlés par i vers les sommets contrôlés par l'autre joueur (noté) tel que tous les coups joués soient valides.
Dans l'arène $G = (V, E)$, la stratégie s est

$$s : V_i' \to V_{\text{autre joueur}}$$

Et s vérifie

$$\forall x \in V_i',\ (x, s(x)) \in E$$

Suffixe Si u et v sont deux mots définis sur un même alphabet Σ, u est un suffixe de v s'il existe $w \in \Sigma^*$ tel que $v = w \cdot u$.

Symétrique
↪ Relation symétrique

Système de Gestion de Bases de Données Un Système de Gestion de Bases de Données, ou SGDB, est un logiciel utilisé pour manipuler des bases de données. Les opérations effectuées sont notamment l'inscription, la recherche, la modification, le tri et la transformation d'informations.

Remarque :
- En anglais, SGDB est DBMS pour *Database Management System*.

↪ Base de données

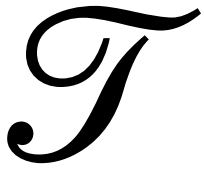


Table Dans une base de données relationnelles, il s'agit d'un tableau constitué de colonnes et lignes. Une colonne rassemble des valeurs du même type, tandis qu'une ligne est une suite de valeurs caractérisant un enregistrement.

Table de vérité La table de vérité d'une proposition P est un tableau représentant les valeurs prises par P selon les valuations des variables propositionnelles intervenant dans P.

Exemple : La table de vérité de la proposition $P = (x \wedge \neg y) \vee (\neg x \wedge y)$ est

x	y	$(x \wedge \neg y) \vee (\neg x \wedge y)$
0	0	0
0	1	1
1	0	1
1	1	0

Tableau Zone contiguë de mémoire permettant de stocker plusieurs valeurs du même type, chacune repérée par son indice.

Tableau statique Tableau de taille fixée à la création.

- en C, on écrit

```c
int t[6];
```

On peut initialiser chaque cellule avec

```c
t[1] = 4;
```

Ou à la création avec la syntaxe

```c
double t[3] = {0.1, 0.2, 0.3}
```

On peut omettre la taille avec cette syntaxe. Si elle s'avère plus grande que le nombre d'éléments renseignés, C complète automatiquement avec la valeur nulle.

- en Ocaml, on utilise le module `Array` en écrivant

```ocaml
let t = Array.make n x
```

Où `n` est la taille du tableau et `x` la valeur initiale de toutes les cases.
On accède ou modifie la valeur d'un élément en écrivant

```ocaml
let t = Array.make 3 0
t.(0) (* 0 *)
t.(1) <- 2
t.(1) (* 2 *)
```

Taille d'un arbre Nombre total de nœuds de cet arbre.

Tas
↪ Tas binaire

Tautologie Formule propositionnelle P telle que toute valuation la satisfait :

$$\mathrm{Mod}(P) = \mathcal{B}^{\mathcal{V}}$$

Terminaison Une fonction termine si elle renvoie un résultat en un nombre fini d'état, et ceux quel que soit ses paramètres.

Theta Si u et v sont deux suites à valeurs réelles positives, on dit que $u = \Theta(v)$ si

$$u = O(v) \text{ et } u = \Omega(v)$$

Top-down Approche descendante pour résoudre un problème, consistant à utiliser une relation de récurrence pour résoudre le problème sur des instances plus simples. C'est une solution récursive, permettant d'éviter les calculs inutiles. On peut mémoïser les résultats des instances plus simple pour éviter de les recalculer.
↪ Sac à dos (problème du)

Transitif Se dit d'une relation qui s'appliquant d'un premier élément à un second et du second à un troisième, s'applique également du premier au troisième.
↪ Relation transitive

Transtypage Conversion d'une variable d'un type vers un autre.

Exemple :

```c
float a = 5.5;
int b = (int)a;// b vaut 5
```

Tri fusion Aussi appelé *merge sort*, il consiste à séparer la séquence initiale en deux séquences triées récursivement pour ensuite les fusionner.
On utilise souvent un tri plus efficace pour les séquences de petite taille.

Tri fusion

Entrée : une séquence u
Sortie : la séquence u triée
fonction $\mathrm{tri}(u)$
  si $|u|$ suffisamment petit **alors**
    renvoyer $\mathrm{tri_bis}(u)$
  sinon
    $v, w \leftarrow u$
    renvoyer $\mathrm{fusion}(\mathrm{tri}(v), \mathrm{tri}(w))$

Algorithme 10. – Tri fusion

Sa complexité est $O(n \log(n))$.

Tri stable Tri conservant l'ordre initial entre des éléments égaux.
Pour tout x, y de le séquence initiale tel que $\mathrm{compare}(x, y) = 0$, on a

$$x \text{ avant } y \text{ dans la séquence triée}$$
$$\Leftrightarrow$$
$$x \text{ avant } y \text{ dans la séquence initiale}$$

Remarque :
- Le résultat d'un tri stable est défini de manière unique.
- Les éléments de la séquence ne sont échangés que lorsque c'est nécessaire.

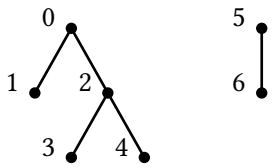


Underflow

↪ Soupassement arithmétique

Union par hauteur Lors d'une opération Merge, on doit choisir quel arbre greffer sur lequel. L'union par hauteur revient à greffer l'ordre le moins haut sur l'arbre le plus haut afin de garder nos arbres les plus « plats » possible.

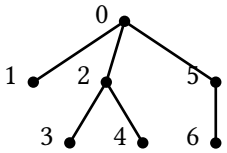
Afin d'accomplir cela, on stocke également un second tableau pour les hauteurs, que l'on garde à jour :



0	1	2	3	4	5	6
2	1	1	0	0	1	0

Figure 20. – UnionFind et le tableau hauteur associé

On greffe donc l'arbre de racine 5 sur celui de racine 0 :



0	1	2	3	4	5	6
2	1	1	0	0	1	0

Figure 21. – Exemple d'union par hauteur

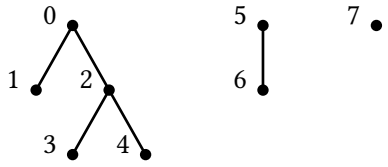
↪ UnionFind

UnionFind Structure de données représentant une partition finie $X = X_1 \sqcup ... \sqcup X_k$ où l'on peut :

- Créer une nouvelle partition en singleton ($[\![0; n-1]\!] = \{0\} \sqcup ... \sqcup \{n-1\}$).
- Trouver à quelle classe appartient un élément x de X.
- Fusionner deux classes en une seule au sein de la partition.

On note $X(x)$ la classe contenant x.

Implémentation : On utilise une forêt pour représenter la partition, chaque élément est un nœud et l'on crée un arbre par classe. On représente cela par un tableau des parents de chaque nœud.



0	1	2	3	4	5	6	7
0	0	0	2	4	5	5	7

Figure 22. – UnionFind et le tableau parent associé

Pour connaître la classe d'un élément x, on accède à l'indice x du tableau et on remonte jusqu'à la racine, que l'on identifie puisqu'elle est son propre parent.

Pour fusionner deux classes, la racine de l'un des arbres devient un fils de la racine de l'autre.



Valuation Avec $\mathcal{V}$ l'ensemble des variables propositionnelles, une valuation est une application

$$v : \mathcal{V} \to \mathcal{B}$$

Où

$$\mathcal{B} = \{0, 1\} \text{ ou } \mathcal{B} = \{F, V\}$$

Implémentation : Souvent, les variables seront notées x_i où i est un entier positif. Une valuation sera alors un tableau v où $v[i]$ vaut la valuation associée à la variable x_i.

Variant de boucle Utilisé pour prouver la terminaison d'un algorithme, un variant de boucle doit être une quantité

- entière
- minorée (respectivement majorée)
- strictement décroissante à chaque itération (respectivement strictement croissante)

Verrou

↪ Mutex



While

↪ Boucle while



Yamada
↪ McNaughton et Yamada

Table des matières

Liste des figures

Figure 1 Arbre ... 4
Figure 2 Arbre dans la nature ... 5
Figure 3 Compression de chemin ... 12
Figure 4 Représentation d'une file ... 18
Figure 5 File en OCaml ... 18
Figure 6 File en C ... 18
Figure 7 Exemple de graphe ... 20
Figure 8 Quelques graphes bipartis complets ... 20
Figure 9 Quelques graphes chemins ... 20
Figure 10 Quelques graphes complets ... 20
Figure 11 Quelques graphes cycles ... 21
Figure 12 Exemple de graphe de flot de contrôle ... 21
Figure 13 Partie de l'arène du tic-tac-toe ... 24
Figure 14 Liste doublement chaînée correspondant à [1; 2; 3] ... 26
Figure 15 Liste simplement chaînée correspondant à [1; 2; 3] ... 26
Figure 16 Matrice d'adjacence d'un graphe orienté ... 28
Figure 17 Matrice d'adjacence d'un graphe non orienté ... 28
Figure 18 Représentation d'une pile ... 32
Figure 19 Pile en C ... 32
Figure 20 UnionFind et le tableau hauteur associé ... 41
Figure 21 Exemple d'union par hauteur ... 41
Figure 22 UnionFind et le tableau parent associé ... 41

Liste des algorithmes

Algorithme 1 Algorithme long ... 4
Algorithme 2 Construction d'un arbre couvrant ... 5
Algorithme 3 Calcul des attracteurs ... 6
Algorithme 4 Algorithme de Bellman-Ford ... 7
Algorithme 5 Clusterin hiérarchique ascendant ... 10
Algorithme 6 Calcul des composantes connexes ... 12
Algorithme 7 Exponentiation rapide ... 17
Algorithme 8 Algorithme de Kruskal ... 25
Algorithme 9 Algorithme de Peterson ... 31
Algorithme 10 Tri fusion ... 40