

Algorithme #2

Michael
X  NATIS



Compétence demandée :
Comprendre les concepts
algorithmiques intermédiaires

1. Récursivité
2. Tri
3. DFS
4. Compression
5. Décisionnel

1. Récursivité
2. Tri
3. DFS
4. Compression
5. Décisionnel

Récurtivité

Qu'est-ce que la récurrence ?



Un **algorithme récursif** est un algorithme qui résout un problème en calculant des **solutions d'instances plus petites** du même problème. L'approche récursive est un des concepts de base en informatique.

Un **algorithme récursif** peut être considéré comme **une fonction qui s'appelle elle-même** ! Cela crée donc une boucle (implicite).

La récursion est une alternative élégante à la boucle.



Pour qu'un algorithme puisse être appelé, il faut qu'il soit nommé et encapsulé dans une procédure ou une fonction.

```
@Fonction compter @Entrée tab, i, longueur
@DebutBloc
@Si i >= longueur
@DebutBloc
@Retourner 0
@FinBloc

@Si tab[i] >= 18
@DebutBloc
@Retourner 1 + compter @Avec tab, i + 1, longueur
@FinBloc
@Sinon
@DebutBloc
@Retourner compter @Avec tab, i + 1, longueur
@FinBloc

@FinBloc
```

```
@Fonction compter @Entrée tab, i, longueur
@DebutBloc
@Si i >= longueur
@DebutBloc
@Retourner 0
@FinBloc

@Si tab[i] >= 18
@DebutBloc
@Retourner 1 + compter @Avec tab, i + 1, longueur
@FinBloc
@Sinon
@DebutBloc
@Retourner compter @Avec tab, i + 1, longueur
@FinBloc

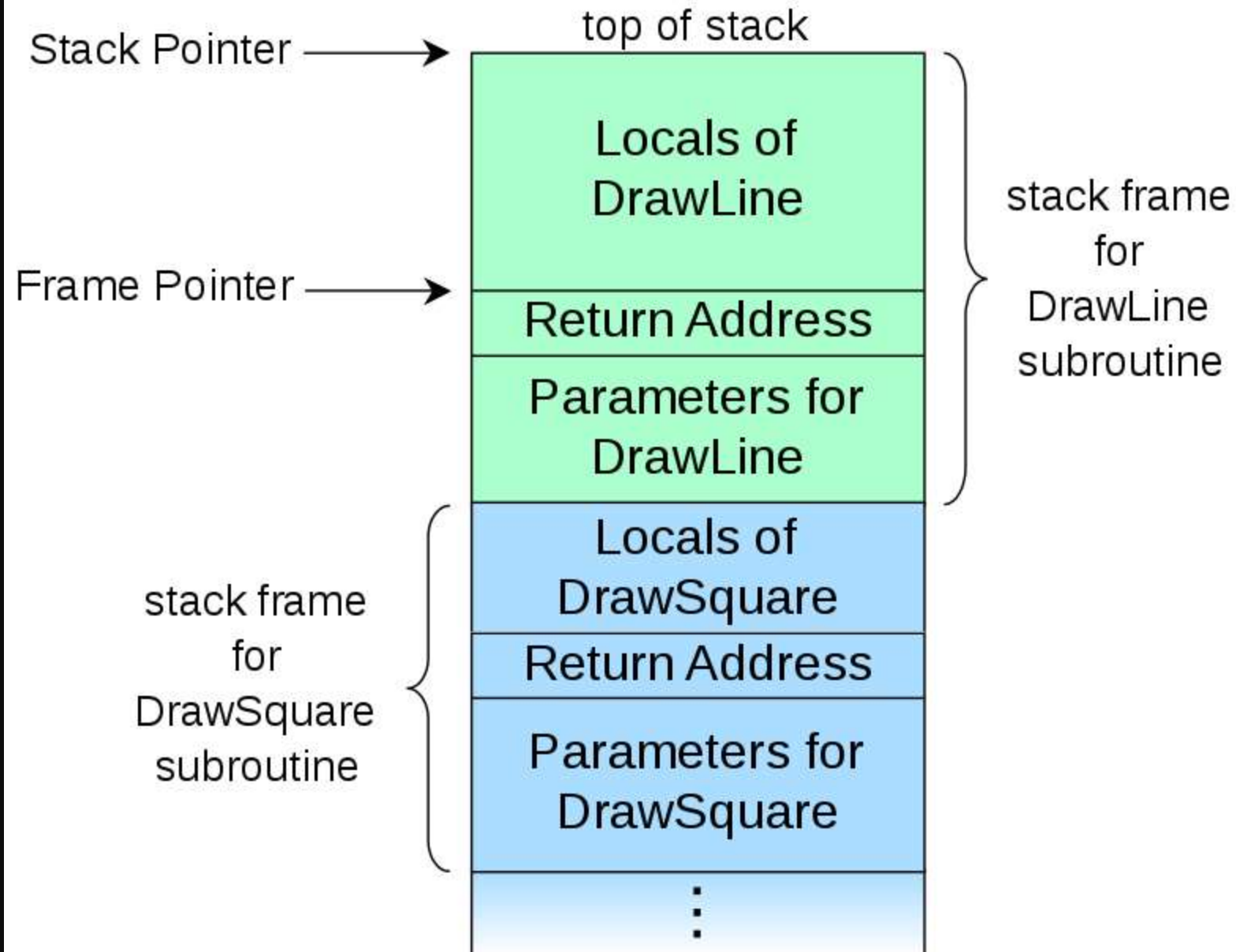
@FinBloc
```

En informatique, la pile d'exécution (souvent abrégée en la pile ; en anglais, call stack) est une structure de données de type pile qui sert à enregistrer des informations au sujet des fonctions actives dans un programme informatique.

```
@Fonction DrawLine @Entrée x1, y1, x2, y2
    @DebutBloc
    // Dessiner une ligne de (x1, y1) à (x2, y2)
    @FinBloc

@Fonction DrawRectangle
    @DebutBloc

    DrawLine @Avec 10, 10, 100, 10
    DrawLine @Avec 100, 10, 100, 100
    DrawLine @Avec 100, 100, 10, 100
    DrawLine @Avec 10, 100, 10, 10
    @FinBloc
```



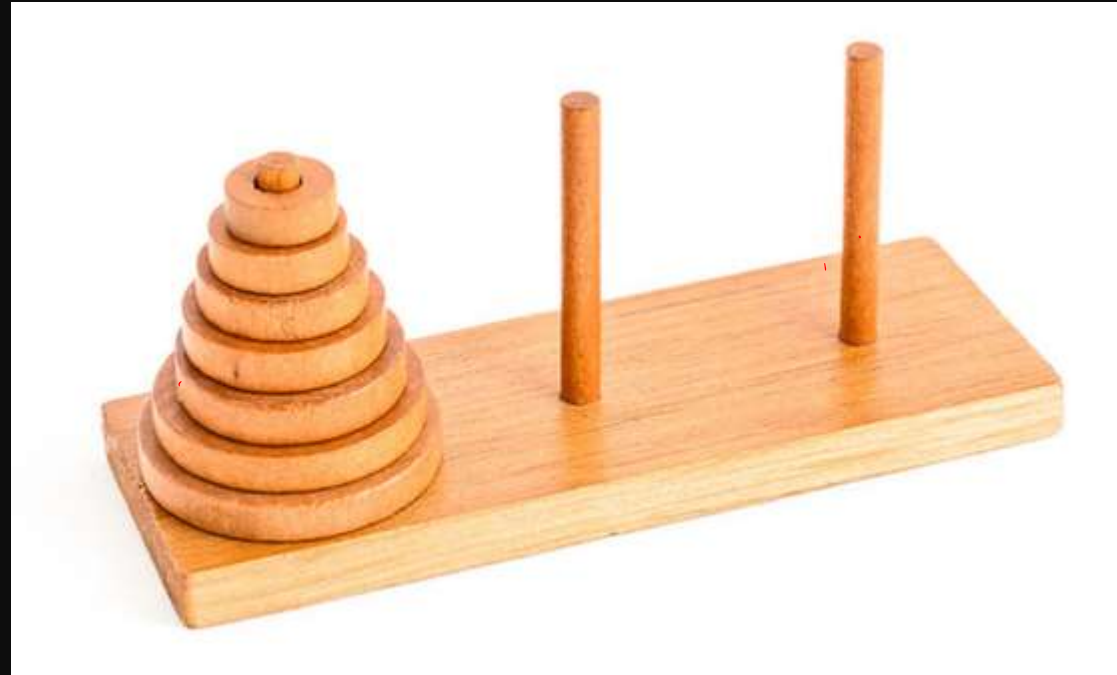
La **réursion** prend **plus d'espace en RAM** qu'un simple boucle. L'espace occupé est proportionnel au nombre d'itérations.



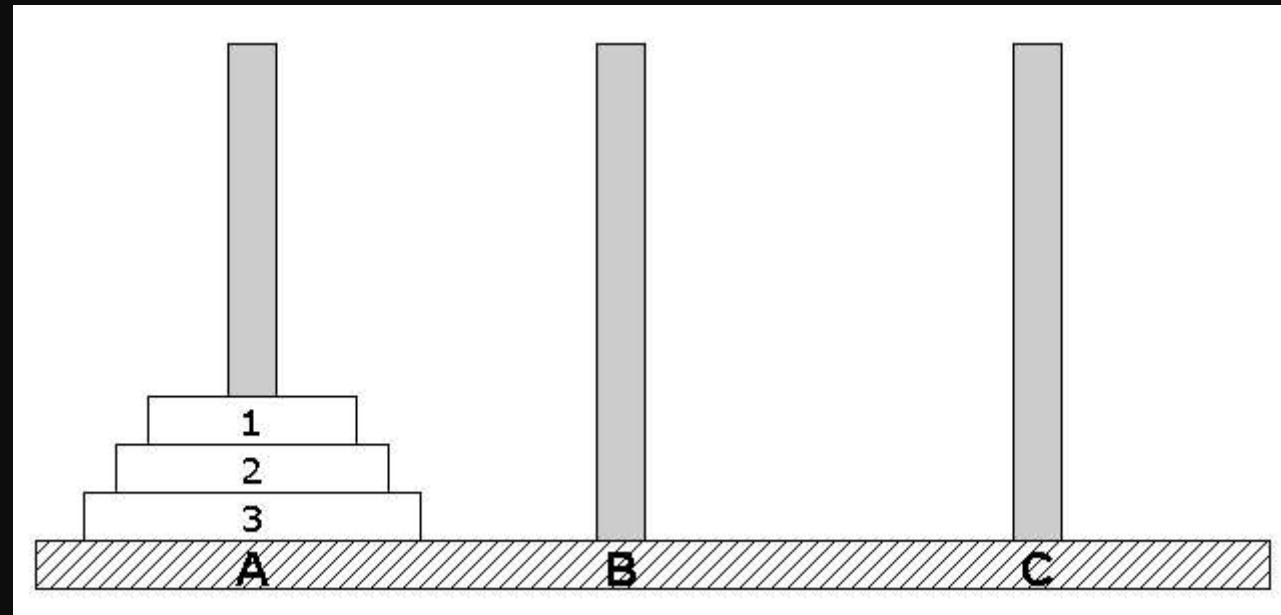
Comment supprimer la récursivité dans un algorithme ?



Tours de Hanoi



- On ne peut déplacer plus d'un disque à la fois
- on ne peut placer un disque que sur un autre disque plus grand que lui ou sur un emplacement vide




```
@Fonction deplacer @Entrée disques, pileDepart, pileDestination, pileBuffer
@DebutBloc

@Si disques vide
    @DebutBloc
    @Retourner
    @FinBloc

deplacer @Avec disques[0..n-2], pileDepart, pileBuffer, pileDestination
Push disques[n-1] sur pileDestination
déplacer @Avec disques[0..n-2], pileBuffer, pileDestination, pileDepart

@FinBloc
```

Théorie des graphes

La théorie des graphes est la discipline mathématique et informatique qui étudie les graphes, lesquels sont des modèles abstraits de dessins de réseaux reliant des objets.

Ces modèles sont constitués par la donnée de **sommets** (aussi appelés nœuds ou points, en référence aux polyèdres), et **d'arêtes** (aussi appelées liens ou lignes) entre ces sommets ; ces arêtes sont parfois non-symétriques (les graphes sont alors dits **orientés**) et sont appelées des flèches ou des **arcs**.

En théorie des graphes, l'algorithme de Dijkstra sert à résoudre le problème du **plus court chemin**. Il permet, par exemple, de déterminer un plus court chemin pour se rendre d'une ville à une autre connaissant le réseau routier d'une région.

L'algorithme de Bellman-Ford, aussi appelé algorithme de Bellman–Ford–Moore, est un algorithme qui calcule des plus courts chemins depuis un sommet source donné dans un graphe orienté pondéré.

Contrairement à l'algorithme de Dijkstra, l'algorithme de Bellman-Ford **autorise la présence de certains arcs de poids négatif** et permet de détecter l'existence d'un circuit absorbant, c'est-à-dire de poids total strictement négatif, accessible depuis le sommet source.

1. Récursivité
2. Tri
3. DFS
4. Compression
5. Décisionnel

Tri

Pourquoi trier des données/valeurs ?



Un **algorithme de tri** est, en informatique ou en mathématiques, un algorithme qui permet d'organiser une collection d'objets selon **une relation d'ordre** déterminée.



Nom	Cas optimal	Cas moyen	Pire des cas	Complexité spatiale	Stable
Tri rapide	$n \log n$	$n \log n$	n^2	$\log n$ en moyenne, n dans le pire des cas ; variante de Sedgewick : $\log n$ dans le pire des cas	Non
Tri fusion	$n \log n$	$n \log n$	$n \log n$	n	Oui
Tri par tas	$n \log n$	$n \log n$	$n \log n$	1	Non
Tri par insertion	n	n^2	n^2	1	Oui
Introsort	$n \log n$	$n \log n$	$n \log n$	$\log n$	Non
Tri par sélection	n^2	n^2	n^2	1	Non
Timsort	n	$n \log n$	$n \log n$	n	Oui
Tri de Shell	n	$n \log^2 n$ ou $n^{3/2}$	$n \log^2 n$ pour la meilleure suite d'espacements connue	1	Non
Tri à bulles	n	n^2	n^2	1	Oui
Tri arborescent	$n \log n$	$n \log n$	$n \log n$ (arbre équilibré)	n	Oui
Smoothsort	n	$n \log n$	$n \log n$	1	Non
Tri cocktail	n	n^2	n^2	1	Oui
Tri à peigne	n	$n \log n$	n^2	1	Non
Tri pair-impair	n	n^2	n^2	1	Oui

Quelle complexité est meilleure ?



Comportement asymptotique

$$O(1) < O(\log n) < O(n) < O(n \log(n)) < O(n*n) < O(\exp(n))$$

Plus **rapide**

Plus **lent**

Problème NP-complet

1. Tri à bulles
2. Tri casier
3. Tri fusion

Tri à bulles

Tri à bulles

Il consiste à faire remonter le maximum en bout de tableau par permutations.

Il doit son nom au fait qu'il déplace rapidement les plus grands éléments en fin de tableau, comme des bulles d'air qui remonteraient rapidement à la surface d'un liquide.

Tri casier

Tri casier

Le tri casier est un tri sans comparaison. Il consiste à distribuer les valeurs dans des casiers et les empiler par après.

Il est aussi appelé tri comptage.

Tri fusion

Le tri fusion, ou merge sort, est un algorithme de tri par comparaison stable.

Sa complexité temporelle pour une entrée de taille n est de l'ordre de $n \log n$, ce qui est asymptotiquement optimal.

Comment est fait le tri des langages interprétés ?



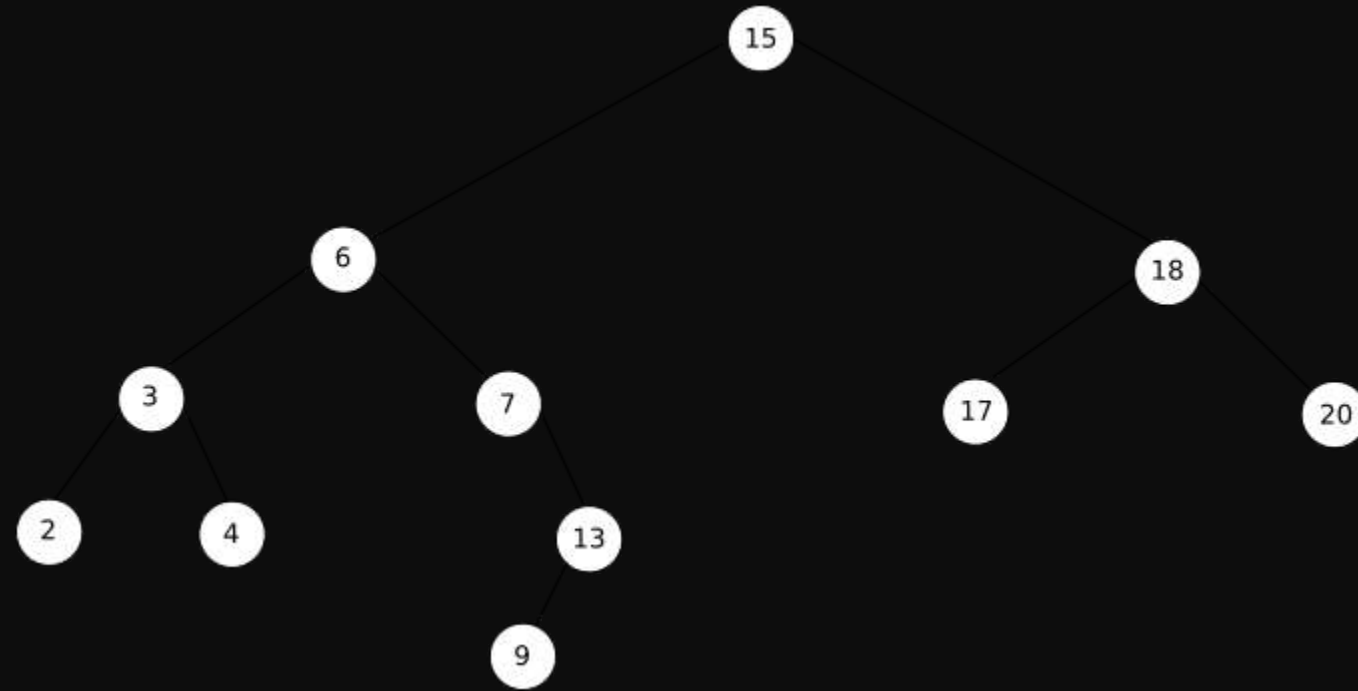
Nom	Cas optimal	Cas moyen	Pire des cas	Complexité spatiale	Stable
Tri rapide	$n \log n$	$n \log n$	n^2	$\log n$ en moyenne, n dans le pire des cas ; variante de Sedgewick : $\log n$ dans le pire des cas	Non
Tri fusion	$n \log n$	$n \log n$	$n \log n$	n	Oui
Tri par tas	$n \log n$	$n \log n$	$n \log n$	1	Non
Tri par insertion	n	n^2	n^2	1	Oui
Introsort	$n \log n$	$n \log n$	$n \log n$	$\log n$	Non
Tri par sélection	n^2	n^2	n^2	1	Non
Timsort	n	$n \log n$	$n \log n$	n	Oui
Tri de Shell	n	$n \log^2 n$ ou $n^{3/2}$	$n \log^2 n$ pour la meilleure suite d'espacements connue	1	Non
Tri à bulles	n	n^2	n^2	1	Oui
Tri arborescent	$n \log n$	$n \log n$	$n \log n$ (arbre équilibré)	n	Oui
Smoothsort	n	$n \log n$	$n \log n$	1	Non
Tri cocktail	n	n^2	n^2	1	Oui
Tri à peigne	n	$n \log n$	n^2	1	Non
Tri pair-impair	n	n^2	n^2	1	Oui

Quelle est la meilleure complexité en pire des cas pour les algorithmes de tri basés sur des comparaisons ?



1. Récursivité
2. Tri
3. DFS
4. Compression
5. Décisionnel

DFS



Le DFS mené sur un BST produit une liste triée

DFS non récursif ?



DFS

Nous pouvons supprimer la récursivité et obtenir un algorithme non récursif en utilisant une pile (une structure LIFO) !

Implémentation
récursive



Implémentation en
pile (LIFO)

1. Récursivité
2. Tri
3. DFS
4. Compression
5. Décisionnel

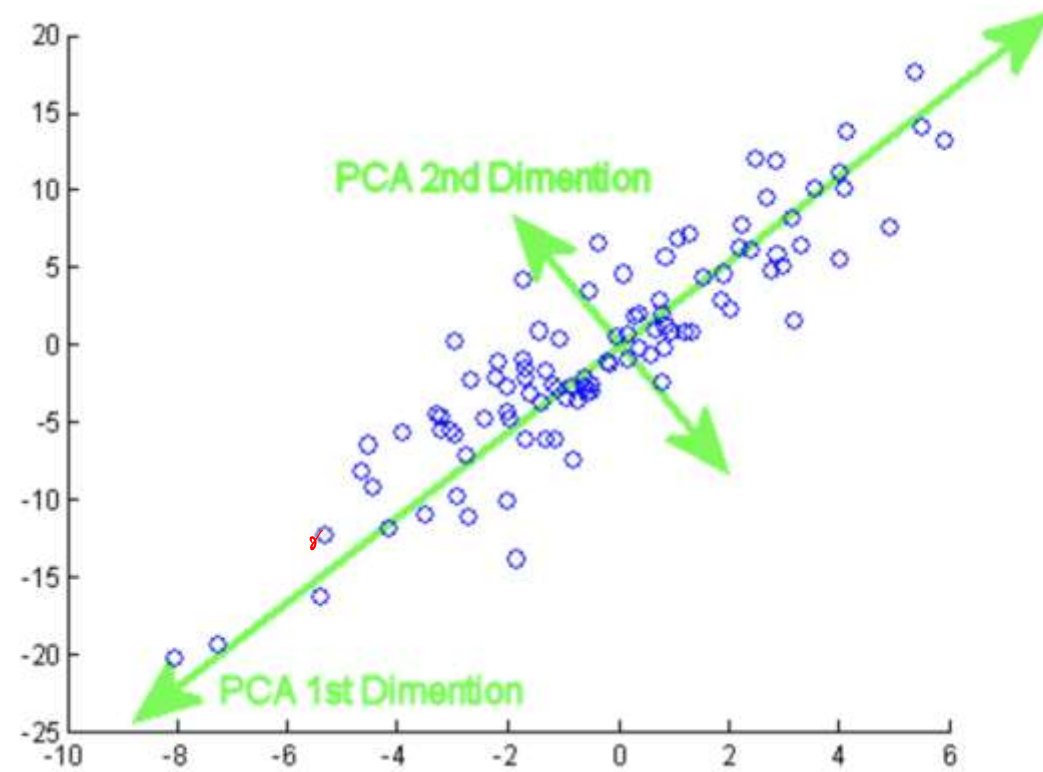
COMPRESSION

Algorithme de Huffman

Le **codage de Huffman** est un algorithme de compression de données **sans perte**. Le codage de Huffman utilise un code à longueur variable pour représenter un symbole de la source (par exemple un caractère dans un fichier).

Le code est déterminé à partir d'une estimation des probabilités d'apparition des symboles de source, un code court étant associé aux symboles de source les plus fréquents.

Analyse en composantes principales (PCA)



La PCA peut être considéré comme un algorithme de **compression avec perte**, qu'il peut être facilement compréhensible.

La PCA vise à **décrire les directions des données** représentées dans un espace vectoriel et non plus les données elles-mêmes.

1. Récursivité
2. Tri
3. DFS
4. Compression
5. Décisionnel

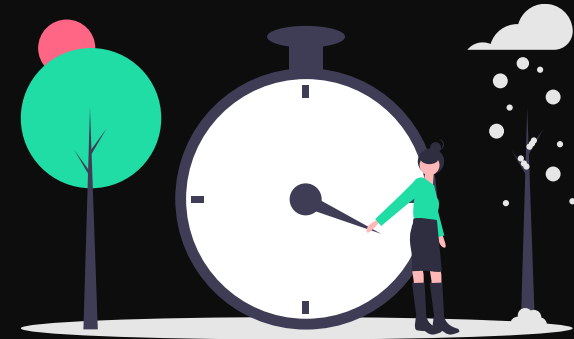
DECISIONNEL

Un système expert est un outil capable de reproduire les mécanismes cognitifs d'un expert, dans un domaine particulier. Il s'agit de l'une des voies tentant d'aboutir à l'intelligence artificielle.

Un système expert a pour ambition de résoudre un problème décisionnel, c'est-à-dire dont la réponse est soit vrai soit faux.



Qu'elle est la formalisation d'un
prédicat ?



Qu'elle est la formalisation d'une proposition ?



Qu'est-ce qu'un syllogisme ?



Tous les chiens sont des mammifères
Il n'existe pas de mammifères qui soient des oiseaux

Quelle conclusion ?

Raisonnement qui se formalise par :

$[\forall x (estChien(x) \Rightarrow estMammifere(x)) \wedge \neg \exists y (estMammifere(y) \wedge estOiseau(y))] \rightarrow \neg \exists z (estChien(z) \wedge estOiseau(z))$ qui se lit

("pour tout x, si x est un chien alors x est un mammifère" et "il est faux qu'il existe y qui est à la fois un mammifère et un oiseau") implique qu'il est faux qu'il existe z qui est à la fois un chien et un oiseau".

Un système part d'une **prémisse** et combiné avec des **règles**, déduit une **conclusion** qui est elle-même une **nouvelle prémisse**.

Ces systèmes peuvent s'appuyer sur la logique d'ordre 0 (**propositions**) ou la logique d'ordre 1 (**prédicats**).