COURS PI

L'école sur-mesure

de la Maternelle au Bac, Établissement d'enseignement privé à distance, déclaré auprès du Rectorat de Paris

Terminale - Module 1 - Structures et bases de données

Numérique et Sciences Informatiques

v.5.1

- **Guide de méthodologie** *pour appréhender notre pédagogie*
- **Leçons détaillées** *pour apprendre les notions en jeu*
- **Exemples et illustrations** *pour comprendre par soi-même*
- **Prolongement numérique** *pour être acteur et aller + loin*
- **Exercices d'application** *pour s'entraîner encore et encore*
- **Corrigés des exercices** *pour vérifier ses acquis*

www.cours-pi.com

Paris & Montpellier

EN ROUTE VERS LE BACCALAURÉAT

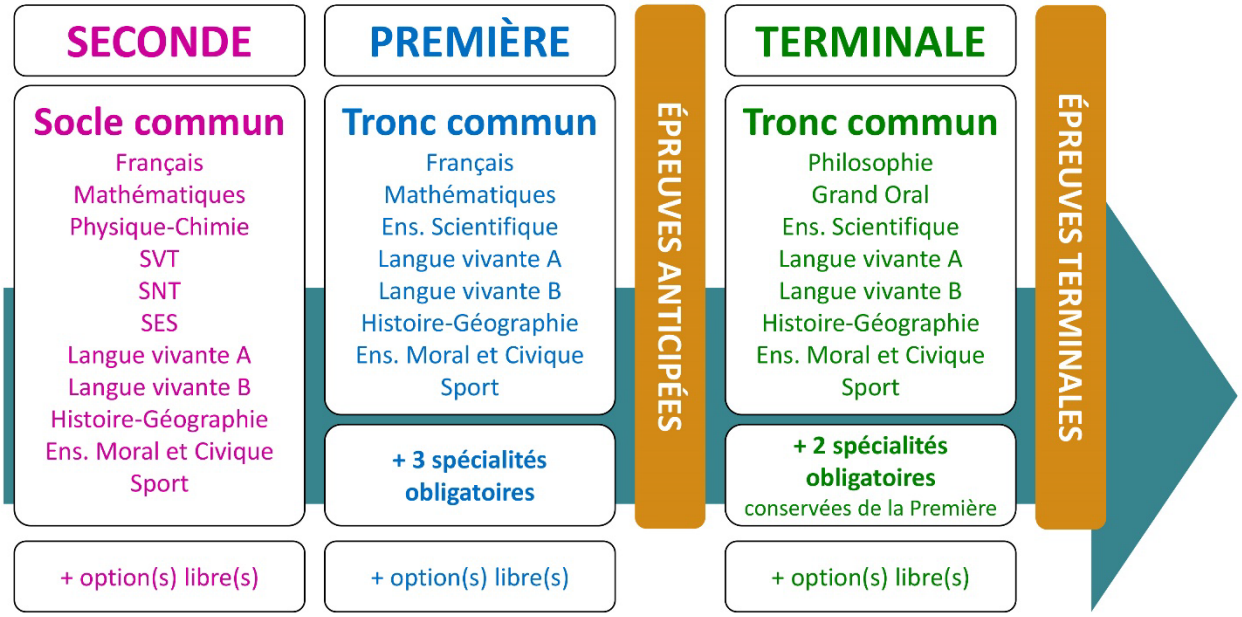
Comme vous le savez, **la réforme du Baccalauréat** est entrée en vigueur progressivement jusqu'à l'année 2021, date de délivrance des premiers diplômes de la nouvelle formule.

Dans le cadre de ce nouveau Baccalauréat, **notre Etablissement**, toujours attentif aux conséquences des réformes pour les élèves, s'est emparé de la question avec force **énergie** et **conviction** pendant plusieurs mois, animé par le souci constant de la réussite de nos lycéens dans leurs apprentissages d'une part, et par la **pérennité** de leur parcours d'autre part. Notre Etablissement a questionné la réforme, mobilisé l'ensemble de son atelier pédagogique, et déployé tout **son savoir-faire** afin de vous proposer un enseignement tourné continuellement vers l'**excellence**, ainsi qu'une scolarité tournée vers la **réussite**.

- Les **Cours Pi** s'engagent pour faire du parcours de chacun de ses élèves un **tremplin vers l'avenir**.
- Les **Cours Pi** s'engagent pour ne pas faire de ce nouveau Bac un diplôme au rabais.
- Les **Cours Pi** vous offrent **écoute** et **conseil** pour coconstruire une **scolarité sur-mesure**.

LE BAC DANS LES GRANDES LIGNES

Ce nouveau Lycée, c'est un enseignement à la carte organisé à partir d'un large tronc commun en classe de Seconde et évoluant vers un parcours des plus spécialisés année après année.



CE QUI A CHANGÉ

- Il n'y a plus de séries à proprement parler.
- Les élèves choisissent des spécialités : trois disciplines en classe de Première ; puis n'en conservent que deux en Terminale.
- Une nouvelle épreuve en fin de Terminale : le Grand Oral.
- Pour les lycéens en présentiel l'examen est un mix de contrôle continu et d'examen final laissant envisager un diplôme à plusieurs vitesses.
- Pour nos élèves, qui passeront les épreuves sur table, le Baccalauréat conserve sa valeur.

CE QUI N'A PAS CHANGÉ

- Le Bac reste un examen accessible aux candidats libres avec examen final.
- Le système actuel de mentions est maintenu.
- Les épreuves anticipées de français, écrit et oral, tout comme celle de spécialité abandonnée se dérouleront comme aujourd'hui en fin de Première.

A l'occasion de la réforme du Lycée, nos manuels ont été retravaillés dans notre atelier pédagogique pour un accompagnement optimal à la compréhension. Sur la base des programmes officiels, nous avons choisi de créer de nombreuses rubriques :

- **Observe, word bank et l'essentiel** pour souligner les points de cours à mémoriser au cours de l'année
- **À vous de jouer** pour mettre en pratique le raisonnement vu dans le cours et s'accaparer les ressorts de l'analyse, de la logique, de l'argumentation, et de la justification
- **Pour aller plus loin** pour visionner des sites ou des documentaires ludiques de qualité
- Et enfin ... la rubrique **Les Clés du Bac by Cours Pi** qui vise à vous donner, et ce dès la seconde, toutes les cartes pour réussir votre examen : notions essentielles, méthodologie pas à pas, exercices types et fiches étape de résolution !

NUMÉRIQUE ET SCIENCES INFORMATIQUES TERMINALE

Module 1 – Structures et bases de données

L'AUTEUR

Adrien SAURAT

« L'enseignement, c'est favoriser l'autonomie et l'enrichissement des élèves, avec en autres objectifs, apprendre un métier. » Professeur et formateur en informatique avec plus de douze ans d'expérience en développement web et dans l'animation du réseau Canopé, il se passionne aussi pour le théâtre et l'écriture de nouvelles.
Des passions qui l'ont déjà conduit sur les planches du Festival d'Avignon.

PRÉSENTATION

Ce **cours** est divisé en chapitres, chacun comprenant :

- Le **cours**, conforme aux programmes de l'Education Nationale
- Des **applications** dont les **corrigés** se trouvent en **fin de chapitre**
- Des **exercices d'entraînement** et leurs **corrigés** en **fin de fascicule**
- Des **devoirs** soumis à correction (et ***se trouvant hors manuel***). Votre professeur vous renverra le corrigé-type de chaque devoir après correction de ce dernier.

Pour une manipulation plus facile, les corrigés-types des exercices d'application et d'entraînement sont regroupés en fin de manuel.

CONSEILS A L'ÉLÈVE

Vous disposez d'un support decours complet : **prenez le temps** de bien le lire, de le comprendre mais surtout de l'**assimiler**. Vous disposez pour cela d'exemples donnés dans le cours et d'exercices types corrigés.
Vous pouvez rester un peu plus longtemps sur une unité mais travaillez régulièrement.

LES DEVOIRS

Les devoirs constituent le moyen d'évaluer l'acquisition de vos *savoirs* (« Ai-je assimilé les notions correspondantes ? ») et de vos *savoir-faire* (« Est-ce que je sais expliquer, justifier, conclure ? »).

Placés à des endroits clés des apprentissages, ils permettent la vérification de la bonne assimilation des enseignements.

Aux *Cours Pi*, vous serez accompagnés par un professeur selon chaque matière tout au long de votre année d'étude. Référez-vous à votre « Carnet de Route » pour l'identifier et découvrir son parcours.
Avant de vous lancer dans un devoir, assurez-vous d'avoir **bien compris les consignes**.
Si vous repérez des difficultés lors de sa réalisation, n'hésitez pas à le mettre de côté et à revenir sur les leçons posant problème. **Le devoir n'est pas un examen**, il a pour objectif de s'assurer que, même quelques jours ou semaines après son étude, une notion est toujours comprise.
Aux *Cours Pi*, chaque élève travaille à son rythme, parce que chaque élève est différent et que ce mode d'enseignement permet le « sur-mesure ».
Nous vous engageons à respecter le moment indiqué pour faire les devoirs. Vous les identifierez par le bandeau suivant :

Vous pouvez maintenant
faire et envoyer le devoir n°1

Il est **important de tenir compte des remarques, appréciations et conseils du professeur-correcteur**. Pour cela, il est **très important d'envoyer les devoirs au fur et à mesure** et non groupés. **C'est ainsi que vous progresserez !**

Donc, dès qu'un devoir est rédigé, envoyez-le aux *Cours Pi* par le biais que vous avez choisi :

1) Par **soumission en ligne** via votre espace personnel sur **PoulPi**, pour un envoi **gratuit**, **sécurisé** et plus **rapide**.
2) Par **voie postale** à *Cours Pi*, 9 rue Rebuffy, 34 000 Montpellier
   *Vous prendrez alors soin de joindre une **grande enveloppe libellée à vos nom et adresse**, et **affranchie au tarif en vigueur** pour qu'il vous soit retourné par votre professeur*

***N.B. :** quel que soit le mode d'envoi choisi, vous veillerez à **toujours joindre l'énoncé du devoir ;** plusieurs énoncés étant disponibles pour le même devoir.*
***N.B. :** si vous avez opté pour un envoi par voie postale et que vous avez à disposition un scanner, nous vous engageons à conserver une copie numérique du devoir envoyé. Les pertes de courrier par la Poste française sont très rares, mais sont toujours source de grand mécontentement pour l'élève voulant constater les fruits de son travail.*

SOUTIEN ET DISPONIBILITÉ

VOTRE RESPONSABLE PÉDAGOGIQUE

Professeur des écoles, professeur de français, professeur de maths, professeur de langues : notre Direction Pédagogique est constituée de spécialistes capables de dissiper toute incompréhension.

Au-delà de cet accompagnement ponctuel, notre Etablissement a positionné ses Responsables pédagogiques comme des « super profs » capables de co-construire avec vous une scolarité sur-mesure.
En somme, le Responsable pédagogique est votre premier point de contact identifié, à même de vous guider et de répondre à vos différents questionnements.

Votre Responsable pédagogique est la personne en charge du suivi de la scolarité des élèves.
Il est tout naturellement votre premier référent : une question, un doute, une incompréhension ? Votre Responsable pédagogique est là pour vous écouter et vous orienter. Autant que nécessaire et sans aucun surcoût.

QUAND PUIS-JE LE JOINDRE ?

Du lundi au vendredi : horaires disponibles sur votre carnet de route et sur PoulPi.

QUEL EST SON RÔLE ?

Orienter les parents et les élèves.
Proposer la mise en place d'un accompagnement individualisé de l'élève.
Faire évoluer les outils pédagogiques.
Encadrer et coordonner les différents professeurs.

VOS PROFESSEURS CORRECTEURS

Notre Etablissement a choisi de s'entourer de professeurs diplômés et expérimentés, parce qu'eux seuls ont une parfaite connaissance de ce qu'est un élève et parce qu'eux seuls maîtrisent les attendus de leur discipline. En lien direct avec votre Responsable pédagogique, ils prendront en compte les spécificités de l'élève dans leur correction. Volontairement bienveillants, leur correction sera néanmoins juste, pour mieux progresser.

QUAND PUIS-JE LE JOINDRE ?

Une question sur sa correction ?

- faites un mail ou téléphonez à votre correcteur et demandez-lui d'être recontacté en lui laissant un message avec votre nom, celui de votre enfant et votre numéro.
- autrement pour une réponse en temps réel, appelez votre Responsable pédagogique.

LE BUREAU DE LA SCOLARITÉ

Placé sous la direction d'Elena COZZANI, le Bureau de la Scolarité vous orientera et vous guidera dans vos démarches administratives. En connaissance parfaite du fonctionnement de l'Etablissement, ces référents administratifs sauront solutionner vos problématiques et, au besoin, vous rediriger vers le bon interlocuteur.

QUAND PUIS-JE LE JOINDRE ?

Du lundi au vendredi : horaires disponibles sur votre carnet de route et sur PoulPi.
04.67.34.03.00
scolarite@cours-pi.com

LE SOMMAIRE

Numérique et Sciences Informatiques - Module 1 - Structures et bases de données

OBJECTIFS

- Spécifier une structure de données par son interface.
- Distinguer interface et implémentation.
- Distinguer des structures par le jeu des méthodes qui les caractérisent.
- Choisir une structure de données adaptée à la situation à modéliser.
- Distinguer la recherche d'une valeur dans une liste et dans un dictionnaire.
- Identifier des situations nécessitant une structure de données arborescente.
- Évaluer quelques mesures des arbres binaires (taille, encadrement de la hauteur, etc.).
- Modéliser des situations sous forme de graphes.

COMPÉTENCES VISÉES

- Les principes de structure de donnée, d'interface et d'implémentation.
- Le vocabulaire de la programmation objet.
- L'identification des structures linéaires : listes, piles, files.
- L'identification des structures en arbres.
- La modélisation de situations sous forme de graphes et leur implémentation.

OBJECTIFS

- Identifier les concepts définissant le modèle relationnel.
- Savoir distinguer la structure d'une base de données de son contenu.
- Repérer des anomalies dans le schéma d'une base de données.
- Identifier les services rendus par un système de gestion de bases de données relationnelles : persistance des données, gestion des accès concurrents, efficacité de traitement des requêtes, sécurisation des accès.

COMPÉTENCES VISÉES

- Le concept de modèle relationnel.
- La distinction entre structure et contenu dans une base.
- La détection d'anomalies dans le schéma d'une base de données.
- L'identification des services rendus par un SGBDR.

OBJECTIFS

- Identifier les composants d'une requête.
- Construire des requêtes d'interrogation à l'aide des clauses du langage SQL : SELECT, FROM, WHERE, JOIN.
- Construire des requêtes d'insertion et de mise à jour à l'aide de : UPDATE, INSERT, DELETE.

COMPÉTENCES VISÉES

- Ce qu'est le langage SQL.
- L'identification des différents composants d'une requête SQL.
- L'utilisation des requêtes d'insertion, de mise à jour et de suppression.
- L'utilisation de la requête d'interrogation.

SUGGESTIONS CULTURELLES

OUVRAGES

- **Les bases de données NoSQL et le BigData: Comprendre et mettre en oeuvre,** *Rudi Bruchez*
- **MDM: Enjeux et méthodes de la gestion des données**, *Franck Regnier-Pecastaing, Michel Gabassi, Jacques Finet.*
- **Big Data, Smart Data, Stupid Data... : Comment (vraiment) valoriser vos données**, *Antoine Denoix*

FILMS ET SERIES

- **Office Space**, *Mike Judge*
- **Jobs**,*Joshua Michael Stern*
- **Snowden**, *Oliver Stone*
- **The Imitation Games**, *Morten Tyldum*

DOCUMENTAIRES AUDIOVISUELS

- **Citizenfour**, *Laura Poitras*
- **Les ordinateurs du passé**, *Le Vortex*
- **Big Data La révolution des mégadonnées,** *Sandy Smolan*
- **Democracy – La ruée vers les datas**, *Daniel Bernet*

SITES ET OUTILS EN LIGNE

- **www.dbdiagram.io/d**
- **www.db-fiddle.com**
- **www.code.org**
- **www.openclassrooms.com**

PRÉSENTATION DE L'ÉPREUVE

Le bac approche, voilà de quoi vous y préparer !

Dans le cadre d'une scolarisation à la maison, une partie de l'épreuve (dite « épreuve pratique ») est annulée. Vous n'aurez donc qu'à travailler l'épreuve dite « écrite ». En quoi consiste-t-elle ?

Le texte officiel annonce ceci :

La partie écrite (d'un durée de 3 heures 30) consiste en **la résolution de trois exercices permettant d'évaluer les connaissances et les capacités attendues conformément aux programmes de première et de terminale de la spécialité**. Chaque exercice est noté sur 4 points.

Le sujet propose cinq exercices, parmi lesquels le candidat choisit les trois qu'il traitera. Ces cinq exercices permettent d'aborder les différentes rubriques du programme, sans obligation d'exhaustivité. Le sujet comprend obligatoirement au moins un exercice relatif à chacune des trois rubriques suivantes :

- traitement de données en tables et bases de données ;
- architectures matérielles, systèmes d'exploitation et réseaux ;
- algorithmique, langages et programmation.

Note importante : les trois exercices mentionnés totalisent 12 points, car dans le cadre d'une épreuve complète, la partie pratique apporte les 8 points supplémentaires. **Dans notre cas, cette note de 12 points sera rapportée à 20 points pour représenter l'intégralité de votre résultat.**

Que peut-on conclure de ce texte ?

Trois exercices à choisir parmi cinq, qui couvrent tout le programme NSI (première et terminale), sans oublier le fait que le programme SNT de seconde pose des bases qu'il est bon de connaître (portes logiques, binaire, etc.). Ces cinq exercices proposés couvrent des domaines variés, ce qui vous permettra de trouver au moins un sujet ou deux sur lesquels vous serez peut-être plus à l'aise. **Mais bien sûr, il n'est pas prudent de négliger certaines parties du programme, car elles sont souvent liées !**


Le Cours

CHAPITRE 1
STRUCTURES DE DONNÉES

Ce module traitera en grande partie des bases de données, que nous verrons surtout en chapitre deux et chapitre trois, mais dans **ce premier chapitre nous commencerons par aborder plusieurs structures de données qui peuvent trouver leur place dans différents programmes**, y compris de petits scripts python comme ceux que nous avons déjà codés en seconde avant-première. Piles, files, arbres ou graphe : ces structures et d'autres encore seront explorées. Principalement ici au niveau théorique, puis grâce au langage python (et ce surtout dans le module 3 de cette année).

Nous ferons aussi un détour par la programmation objet, qui permet de programmer en mettant certaines données au centre de l'organisation du code.

Ce chapitre établira des liens avec le module trois consacré intégralement à la programmation. En effet, Nous avons évoqué des types abstraits permettant de théoriser des structures de données aux usages divers, mais leur implémentation technique (dans notre cas, leur écriture en Python) interviendra plus tard dans l'année.

OBJECTIFS

- Spécifier une structure de données par son interface.
- Distinguer interface et implémentation.
- Distinguer des structures par le jeu des méthodes qui les caractérisent.
- Choisir une structure de données adaptée à la situation à modéliser.
- Distinguer la recherche d'une valeur dans une liste et dans un dictionnaire.
- Identifier des situations nécessitant une structure de données arborescente.
- Évaluer quelques mesures des arbres binaires (taille, encadrement de la hauteur, etc.).
- Modéliser des situations sous forme de graphes.

COMPÉTENCES VISÉES

- Les principes de structure de donnée, d'interface et d'implémentation.
- Le vocabulaire de la programmation objet.
- L'identification des structures linéaires : listes, piles, files.
- L'identification des structures en arbres.
- La modélisation de situations sous forme de graphes et leur implémentation.

MATÉRIEL NÉCESSAIRE

- Un ordinateur avec système d'exploitation Windows récent (Windows 8 et supérieur).

Première approche

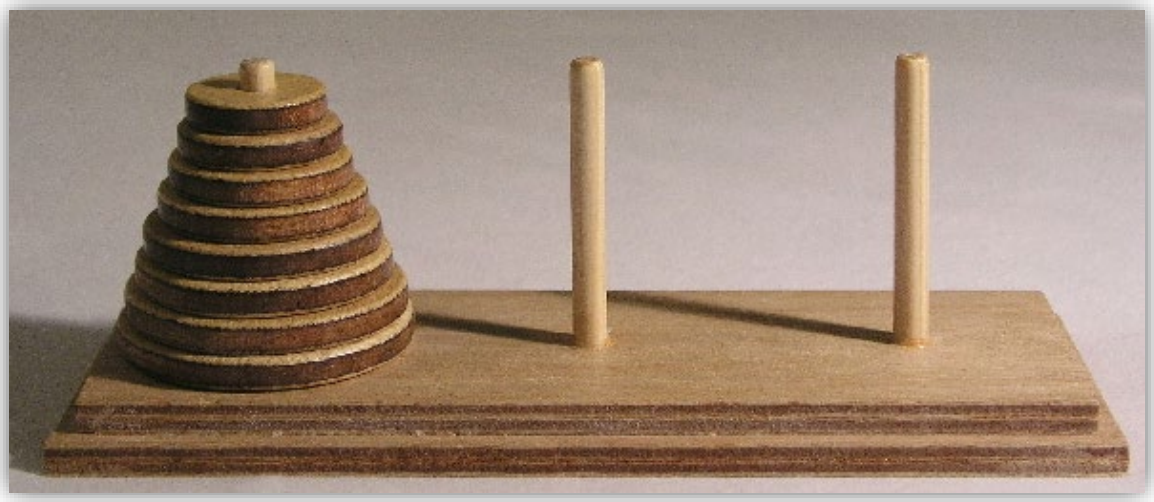
Jouons ! Connaissez-vous les tours de Hanoï ? Ce terme désigne un jeu de réflexion, créé par le mathématicien français Edouard Lucas à la fin du XIX^{ème} siècle. La création de ce jeu est entourée de différentes légendes plus ou moins crédibles, l'une d'elle indiquant que de tels tours existent dans un temple indien à Kashi Vishwanath, et que le jour où ce puzzle serait résolu, la fin du monde arriverait ! Heureusement, dans cette version de l'histoire, il y a 64 disques à déplacer, ce qui, à raison d'un disque bougé par seconde, prendrait des milliards d'années.

Pourquoi est-ce que déplacer 64 disques prendrait tant de temps ?
Parce qu'on doit respecter certaines règles ! Allons-y, expliquons le principe du jeu.

Pour ce qui est de la mise en place. Nous avons :

- Trois tours (des bâtons de bois si on veut se faire une version personnelle) ;
- Un certain nombre de disques (au nombre de huit dans la version la plus courante, mais cela peut varier), tous de rayon différent, et placés au départ sur la même tour (le plus large étant en bas, le plus étroit en haut, et entre les deux on reste sur une décroissance régulière).

Cela peut donner ceci avec huit disques :



Nous voyons tous les disques empilés d'une certaine façon sur la tour de gauche (qu'on peut aussi appeler tour numéro 1).

Le but du jeu sera de déplacer cette pile sur la tour de droite (numéro 3), sachant que les pistes devront être empilés de la même façon.

Dernier point important, on doit effectuer les mouvements en respectant les règles suivantes :

- On ne peut déplacer qu'un disque à la fois ;
- On ne peut placer un disque que sur un autre disque de rayon supérieur, ou sur un emplacement vide (tour ne disposant d'aucun disque).

EN PASSANT, SUR LE WEB…

Jouer aux Tours de Hanoï en ligne

Rendez-vous dans votre moteur de recherche préféré, et saisissez « Tours de Hanoï en ligne » ou « Hanoi towers online » et vous devriez trouver plusieurs versions du jeu, gratuitement accessible.

Toutes ne sont pas aussi pratiques à utiliser, n'hésitez pas à en tester deux ou trois avant de faire votre choix. Une fois la bonne interface trouvée, essayez de résoudre l'énigme (avec trois ou quatre disques ce sera déjà très bien… en ajoutant plus de disques la durée de la partie peut s'allonger considérablement.

Question : avez-vous réussi à déplacer la pile initiale de la gauche vers la droite ? Combien de coups ont été nécessaires ?

.....

.....

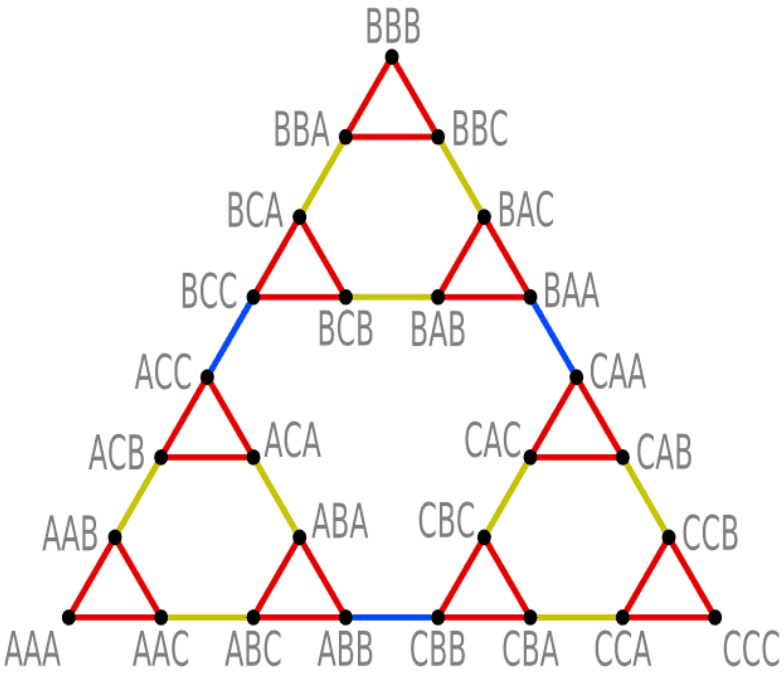
.....

Peut-être avez-vous joué sur trois disques en premier lieu ? Dans ce cas, si vous avez joué moins de dix coups, vous êtes dans le bon ordre de grandeur, mais cette version peut être résolue en sept coups ! Pour quatre disques, il en faut quinze, et pour cinq disques, trente et un coups. Pour huit disques ? Deux cent cinquante-cinq coups ! Est-ce que vous voyez là une progression qui vous rappelle quelque chose ?

Et bien, cela fait écho aux progressions exponentielles que l'on a pu voir les années précédentes dans l'écriture binaire, basée sur les puissances de deux. Néanmoins, si ce détail est intéressant à relever, nous voyons aujourd'hui les tours de Hanoï en ce qu'elles permettent d'illustrer plusieurs structures de données que nous allons étudier dans ce chapitre. Voyez-vous comment chaque pile de disques ne peut être modifiée qu'en agissant sur son sommet, là où le dernier disque a été ajouté ou enlevé ? Nous verrons une structure (appelée tout simplement « pile ») qui fonctionne exactement comme cela !

Quant aux solutions du problème, elles peuvent être représentées sous forme d'arbres ou de graphes qui nous serviront aussi par la suite. Examinons d'un peu plus près ces représentations de l'énigme des tours.

Voici une représentation de la résolution pour 3 disques :



Nous décrirons son interprétation un peu plus loin, mais en attendant, est-ce que cela vous rappelle certains concepts ? D'après vous, à quoi correspond chaque « arête », c'est à dire chaque segment séparant deux points de ce schéma ?

↳ Et bien, chaque arête correspond à un « mouvement » que l'on peut effectuer.

Si « A » correspond à la pile de gauche, « B » à celle du milieu et « C » à celle de droite, les sommets AAA, BBB et CCC représentent les situations où les 3 disques sont sur un même poteau.

Si on l'on veut passer de AAA à BBB ou CCC, on commence par déplacer un disque vers B ou C (on arrive alors à AAB ou AAC), etc.

Par exemple : le chemin le plus rapide pour aller de AAA à CCC passe par la ligne droite à la base du triangle. La couleur des arêtes sert à savoir quel disque on déplace : rouge pour le plus petit, bleu pour le grand disque et jaune pour le disque intermédiaire.

Cela donne :

AAA *(trois disques sur la première pile A)*
AAC *(on a déplacé le petit disque vers C)*
ABC *(on a déplacé le disque moyen vers B)*
ABB *(on a déplacé le petit disque sur B, il se trouve donc sur le disque moyen)*
CBB *(on a déplacé le grand disque vers C)*
CBA *(on a déplacé le petit disque vers A)*
CCA *(on a déplacé le disque moyen vers C)*
CCC *(on a déplacé le petit disque vers C, la pile y est complète et dans le bon ordre)*

1. Utilisez un simulateur de Tours de Hanoï en ligne ! Pouvez-vous résoudre le puzzle en le jouant avec quatre disques ?

2. Revenant sur le triangle proposant la solution pour 3 disques, vous rappelle-t-il des schémas de décision ou de structures de données que vous connaissez ?

CORRECTION :

1. Vous devriez pouvoir y arriver en persévérant un peu ! Si besoin, certains sites permettant de jouer à ce jeu proposent des solutions visualisables automatiquement. Sinon, des vidéos YouTube permettent de voir comment y arriver, comme *« Tower of Hanoi - 4 Disks » ou « Tower of Hanoi: Four Rings Solution 4. »* !

2. Cela peut rappeler les choix d'ouvertures dans le jeu d'échecs, ou tout simplement l'arborescence d'un système de fichiers ! Il s'agit en effet d'un cas particulier d'arbre, que l'on pourrait aussi rapprocher des graphes. Ces différentes structures de données seront examinées plus en détail dans ce chapitre !

STRUCTURES DE DONNÉES
Structures linéaires

Listes

Vous avez sans doute rencontré en seconde et en première des variables de type liste ou tableau. Vous vous en servirez à nouveau, **mais attention : dans ce chapitre le terme de liste revêt un sens différent**. **En effet, une liste désignera ici une organisation de données que l'on peut implémenter de différentes façons et dans différents langages.**

Il existe bien sûr des points communs entre les listes déjà étudiées et celle que nous allons observer aujourd'hui, gardez simplement à l'esprit que suivant le contexte, ce mot peut désigner différentes choses en informatique.

Mais alors ? De quoi allons-nous parler ?

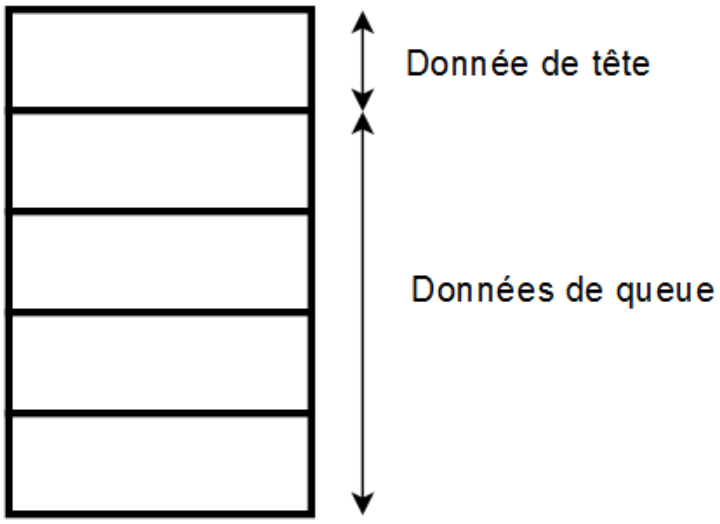
D'une structure de données qui fut principalement mise en application à l'apparition du langage Lisp, créé par John McCarthy en 1958. **Une liste permet de regrouper des données en les organisant suivant deux catégories :**

- **Donnée de tête**
- **Données de queue**

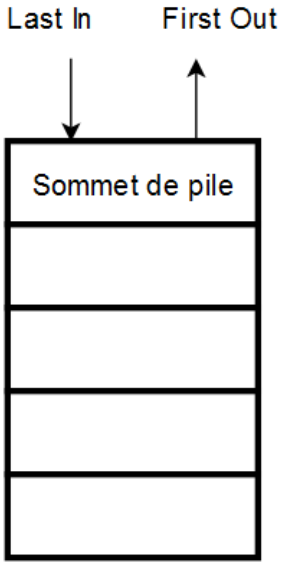
Ainsi, à la manière d'une chenille (et surtout d'une chenille numérique à la manière de celle du jeu Centipede), la tête est composée d'un seul élément tandis que la queue contient tout le reste.

Pour ce qui est de manipuler une telle liste, on dispose des opérations suivantes :

- Création d'une liste vide ;
- Vérification de l'état d'une liste (est-elle vide ?) ;
- Ajout d'un élément en tête de liste ;
- Suppression de la tête de liste ;
- Comptage du nombre d'éléments présents dans la liste.



Piles



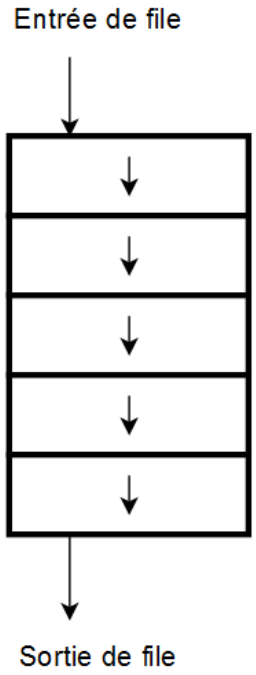
Les piles constituent une autre organisation de données, et s'inspire largement des listes que nous venons de voir. Dans ce cas précis, **on ne peut manipuler que le dernier élément ajouté à la pile**. C'est comme lorsqu'un tout jeune enfant empile des cubes : il peut en ajouter un au-dessus ou retirer le dernier, mais s'il touche aux cubes situés en dessous…

Lorsque seul le dernier élément est manipulable, on parle de structure LIFO, ce qui signifie « Last In First Out » *(dernier entré, premier sorti).*

Les opérations courantes à propos d'une pile sont les suivantes :

- Vérification de l'état de la pile (vide ou pas) ;
- Ajout d'un élément sur la pile ;
- Lecture de l'élément accessible (au sommet de la pile) ;
- Récupération de l'élément accessible (au sommet de la pile), il est alors lu mais aussi supprimé ;
- Comptage du nombre d'éléments de la pile.

Files



Nous repartons encore une fois de la liste pour partir vers une autre façon d'organiser nos données. **C'est ici un peu l'inverse d'une pile.** En effet, **dans une file, ce n'est pas le dernier élément ajouté qui ressort en premier, mais celui qui est le plus ancien.** Dans la même logique que précédemment, on parle de structure FIFO pour « First In First Out » (premier entré, premier sorti).

Les opérations courantes à propos d'une file sont les mêmes que pour la pile, mais appliquées au premier élément ajouté au lieu du dernier :

- Vérification de l'état de la file (vide ou pas) ;
- Ajout d'un élément sur la file ;
- Lecture de l'élément accessible (au début de la file) ;
- Récupération de l'élément accessible (au début de la file), il est alors lu mais aussi supprimé ;
- Comptage du nombre d'éléments de la file.

Dictionnaires

Tout comme pour les listes, vous avez peut-être déjà étudié en python les dictionnaires. Nous allons généraliser ce concept évoquer le type abstrait qu'on appelle dictionnaires, mais aussi tableau associatif.

Ici, la position d'un élément de notre structure n'a pas d'importance. Ce qui nous permet de retrouver nos données, **c'est l'utilisation d'identifiants, ou clés. À chaque clé est associée une valeur, on parle donc de couple clé valeur.**

- Pour ce qui est des opérations courantes nous retrouvons donc :
- L'ajout d'un nouveau couple clé valeur ;
- La modification d'un couple existant ;
- La suppression d'un couple existant ;
- La recherche d'une valeur à partir d'une clé.

L'implémentation Python de ce concept a déjà été abordée dans le chapitre 2 du module 2 du programme de NSI Première. Elle sera rapidement revue dans le module 3 du programme de cette année.

Listes ou dictionnaires ?

Organiser les données en liste, pile ou file permet de traiter plus facilement des problématiques particulières. Si l'on veut par exemple simuler un trafic automobile on pourrait ainsi recourir à des fils pour simuler les séries de véhicules arrêtés devant un feu rouge.

Nous avions vu dans python d'autres formes d'organisation de données, et parmi celles-ci le dictionnaire. Rappelez-vous que dans un dictionnaire, on accède à une valeur en invoquant la clé qui lui correspond. Par exemple, dans un répertoire téléphonique, si chaque personne a pour identifiant son numéro de téléphone et pour valeur son nom, on peut accéder directement au nom d'une personne en appelant le dictionnaire avec le numéro de téléphone correspondant. **Ainsi, dans un dictionnaire, le concept de début ou de fin n'a pas d'importance**. Il en va de même pour l'ordre des données, on peut éventuellement afficher les numéros dans l'ordre croissant en les triant, mais c'est rarement pertinent et python ne stocke pas le dictionnaire d'une manière ordonnée.

Il est clair que pour tout programme, la réflexion sur les deux variables à utiliser est primordiale. Chaque type peut se révéler très utile dans un cas et inutile dans un autre.

Abstraction des structures de données

Nous avons déjà observé quelques structures de données. Avant d'aller plus loin, revenons sur ce que peut regrouper ce terme ? **Il désigne une structure permettant de gérer un ensemble de données à l'aide d'un certain nombre de fonctions que l'on appellera méthode.**

La somme de ces méthodes est appelée interface, dans le sens où c'est par ce biais que les utilisateurs / utilisatrices peuvent interagir avec l'ensemble de données.

Vient ensuite le concept d'implémentation. En effet, si une interface prévoit des actions que l'on peut faire sur une structure, la façon de coder ses actions peut varier suivant le langage informatique employé mais aussi suivant nos habitudes de programmation ou les contraintes à respecter (économiser de la mémoire, de l'espace, etc.).

En conséquence, pour une personne qui utilise une structure de données, l'interface est visible (on s'en sert via des noms de fonctions comme par exemple « ajouterValeur »), mais pas l'implémentation (il s'agit là du code interne). Lorsque vous utilisez un module python avec la commande import, vous avez ainsi accès à de nouvelles fonctions, mais sans savoir nécessairement ce qui se cache derrière.

STRUCTURES DE DONNÉES

Structures hiérarchiques

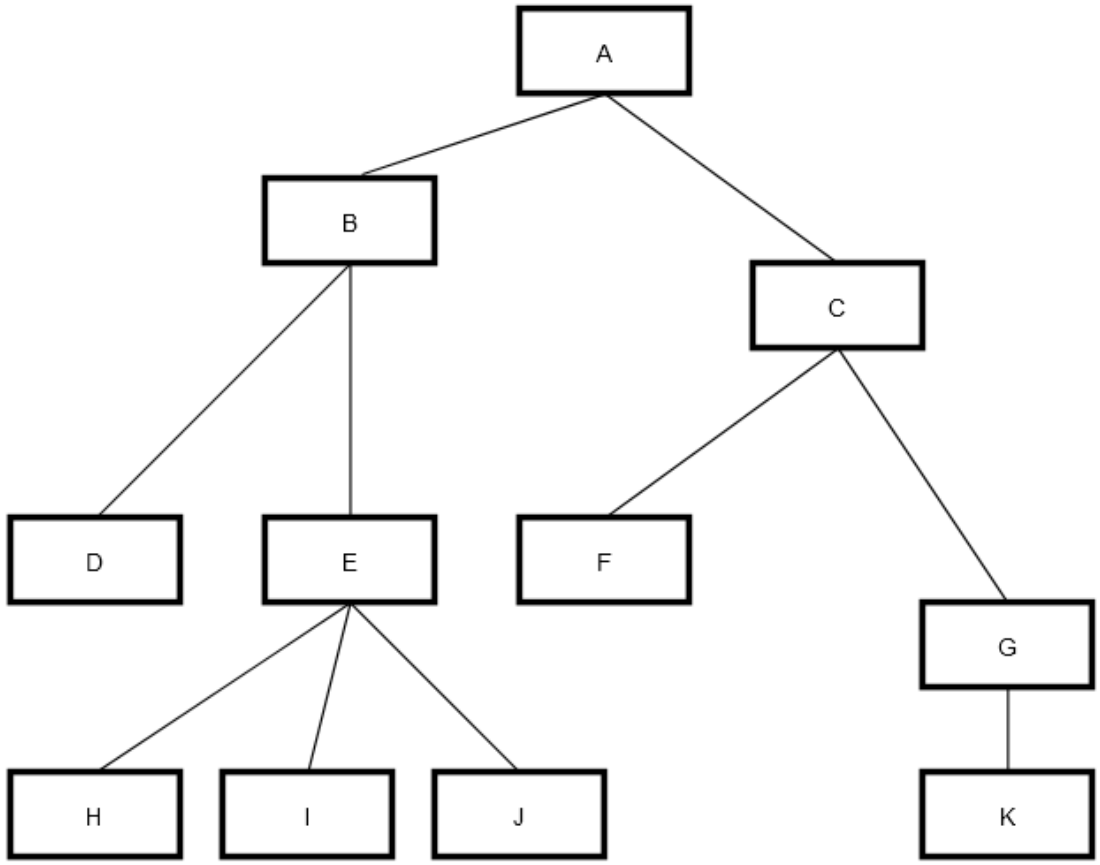
Arbres

Parmi les types abstraits qui transcendent les systèmes et peuvent être implémentés aussi bien en python que dans d'autres langages, nous avons les structures hiérarchiques et notamment l'arbre. En fait, vous avez nécessairement rencontré des données organisées de cette façon, ne serait-ce qu'en naviguant dans les dossiers de votre ordinateur. **En effet, un disque dur est organisé d'une telle manière : il contient un répertoire racine, ainsi que des sous-répertoires.**

Cette capacité de stocker des données de manière hiérarchique permet d'établir des notions de :

- Proximité (distance d'un nœud à l'autre)
- Temporalité (un niveau donné peut correspondre à un moment précis, et les niveaux inférieurs à d'autres moments… comme les différentes étapes d'un tournoi sportif : quarts de finale, demi-finales, finale)
- Contenance (un sous-arbre est « contenu » dans son nœud racine)

Mais nous utilisons déjà quelques termes qu'il est bon de définir !



Chaque élément de l'arbre (contenant ici des lettres) est appelé **nœud**.

Le nœud initial (ici A) en est la **racine**.

B et C sont les **fils** du nœud A, D et E sont les fils du nœud B, etc.

Un nœud n'ayant aucun fil (ici D ou F par exemple) est appelé **feuille.**

On dit aussi des arbres qu'il s'agit de structures **récursives**. Cela est dû au fait que si l'on prend un un nœud quelconque dans un arbre (qui a au moins un fil), il constitue lui-même la racine d'un **sous-arbre** si on prend en compte ses fils. **Ainsi, chaque sous-arbre peut être aussi considéré comme un arbre.**

Exemple des ouvertures d'échecs

Le jeu d'échecs est étudié en détail depuis des siècles ! De nombreuses bases de mouvements permettent de retrouver les coups les plus fameux, et il existe par exemple des bibliothèques de coups dits **« d'ouverture »** qui permettent de commencer une partie de manière optimale.

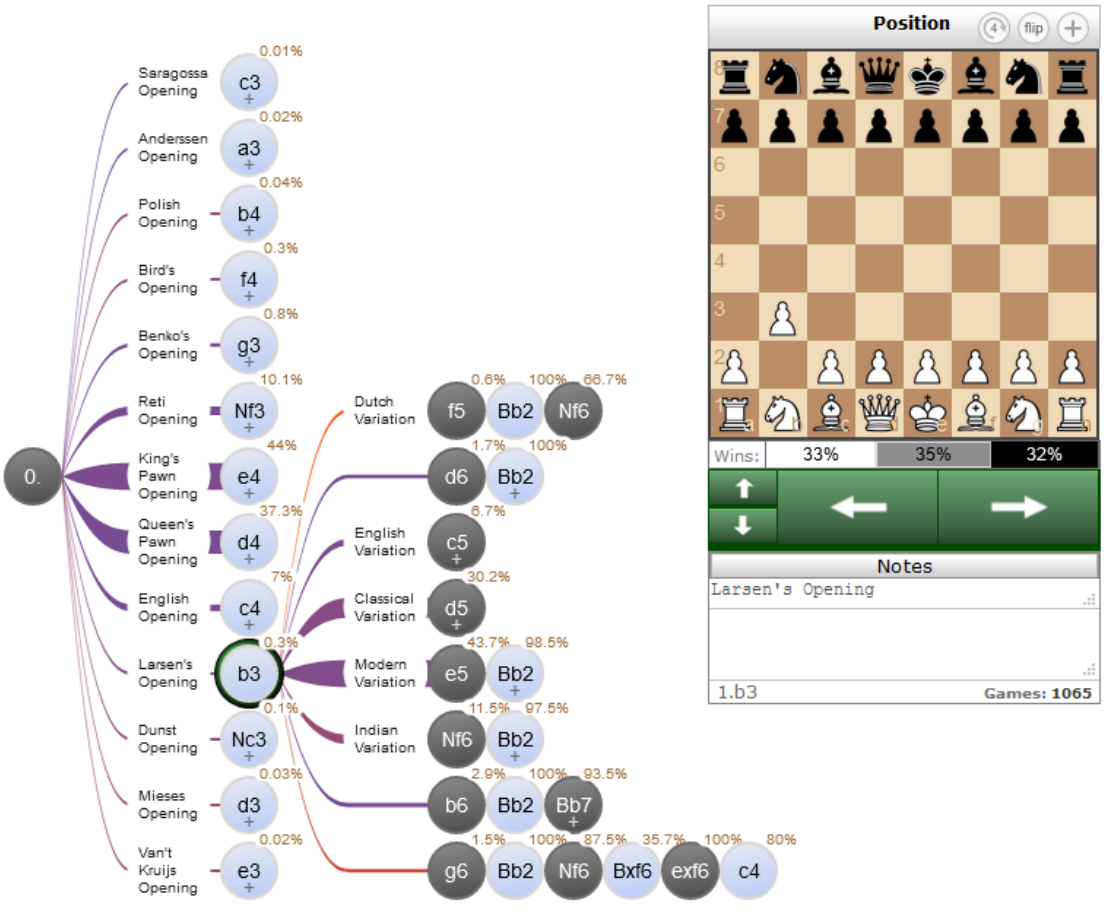
Or, ce type d'information peut être stocké sous forme d'arbre ! En effet, le nombre de coups possibles à chaque tour est limité, et d'un coup donné découlent les coups possibles à la suite.

Par exemple, sur le site chesstree.net, on peut visualiser facilement les sous-arbres découlant de certaines ouvertures connues (qui ont toutes un nom).

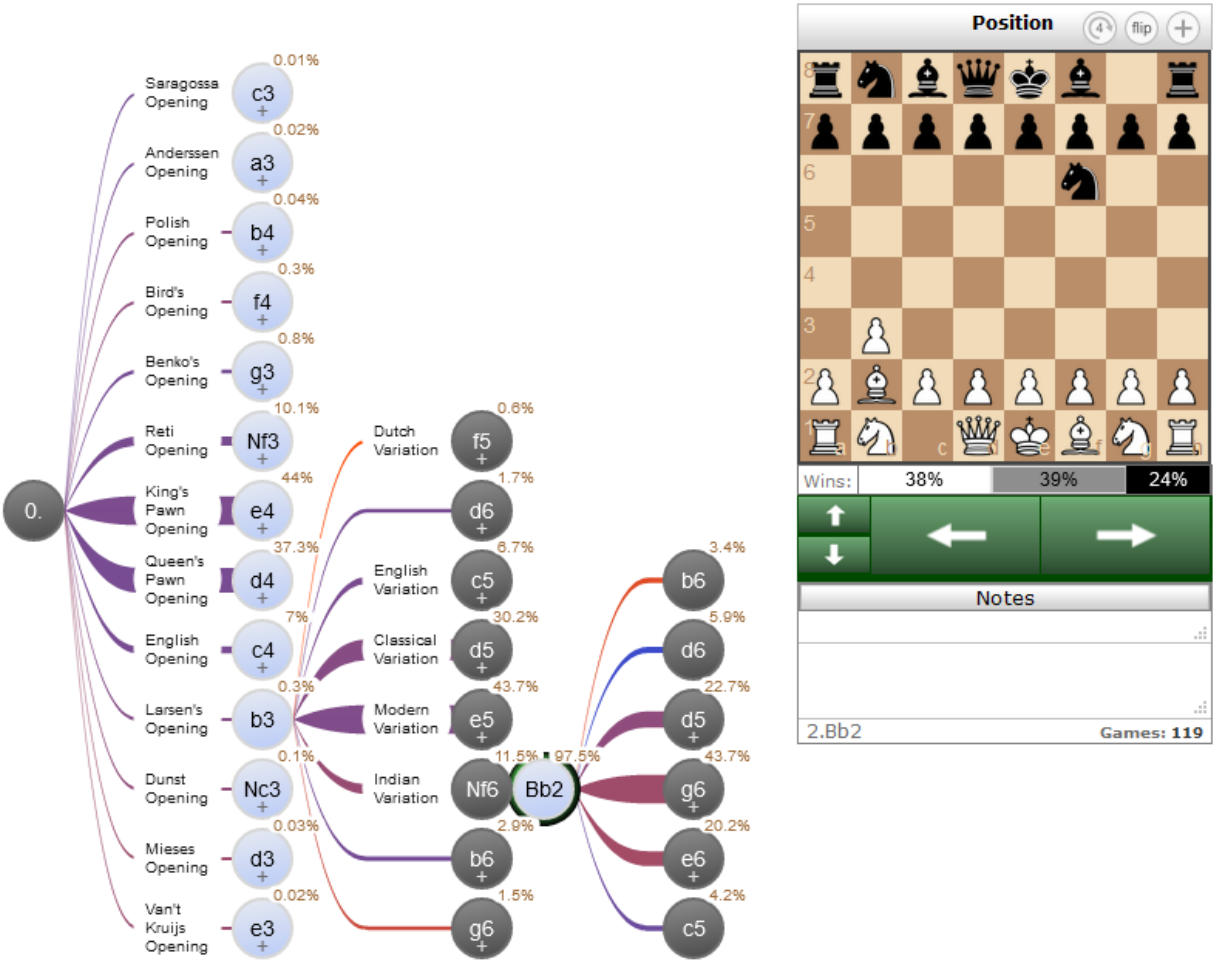
Par exemple :

On voit ici l'arbre initial à gauche (nous allons pouvoir choisir d'explorer une ouverture plutôt qu'une autre). Notez que des informations statistiques sont incluses. On voit que la *« King's Pawn Opening »* est utilisée dans 44 % des parties enregistrées. Puis vient la *« Queen's Pawn Opening »* avec 37,3 % d'occurrences, et la *« Reti Opening »* qui apparaît dans 10,1 % des parties.

Explorons l'ouverture de Larsen, qui commence par un mouvement de pion en B3 :



On peut alors passer la souris sur les réponses Noir probables, puis sur les poursuites de Blanc qui peuvent en découler, etc. Si nous développons la variation indienne, nous arrivons à cet arbre :

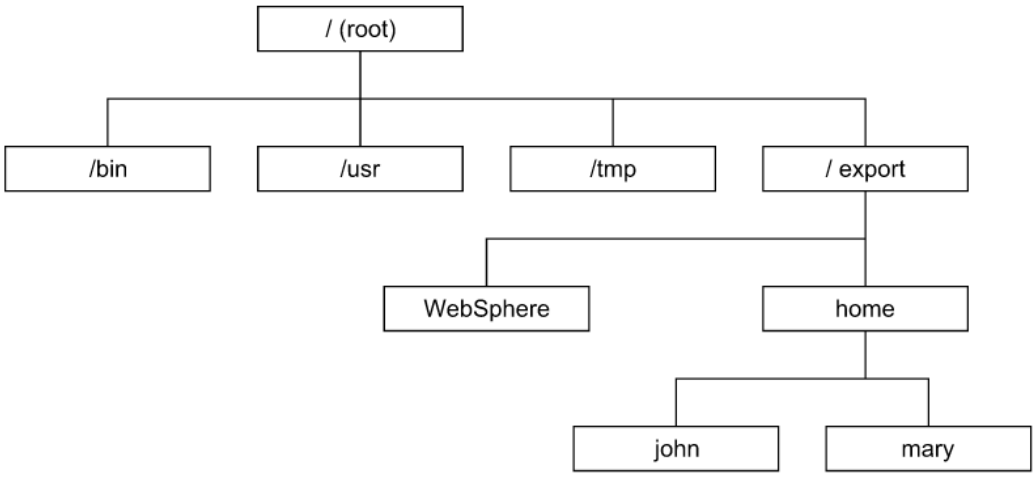


On constate à quel point l'arbre de décisions du jeu d'échec peut contenir un vaste nombre de branches ! D'autant que les possibilités de coups augmentent une fois que les pions ont pris le large, laissant ainsi plus de marge de manœuvre aux pièces majeures (qui ont un grand potentiel de déplacement) !

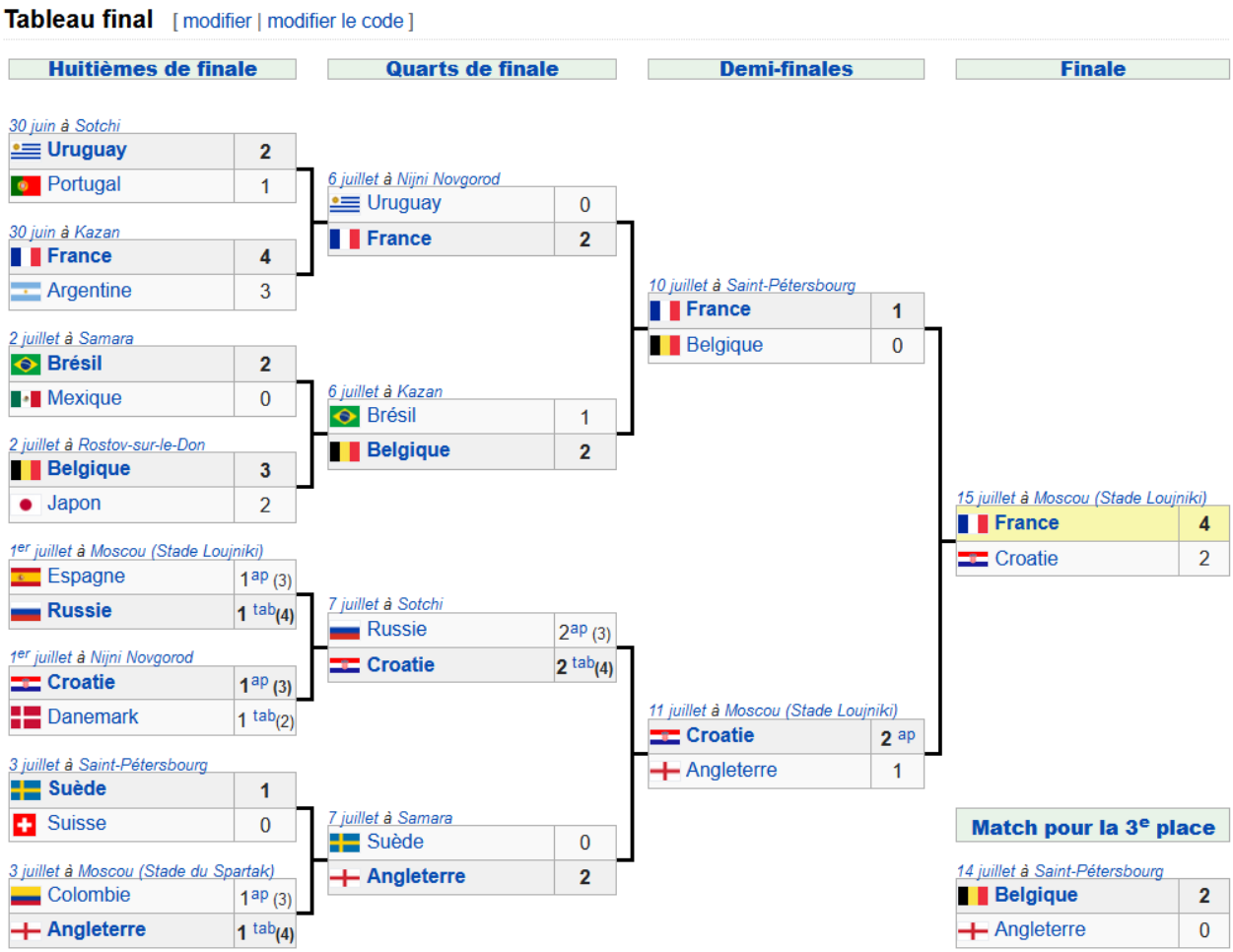
Autres exemples d'arbres

Le système de fichiers d'un ordinateur est organisé sous forme d'arbre : avec un nœud racine (qu'on appelle d'ailleurs aussi racine dans le système de fichiers), **puis des répertoires et sous-répertoires** (autant de nœuds et de sous-arbres). Un fichier, ou un dossier ne contenant rien, constituent une « feuille » (comme tout nœud sans fils dans un arbre).

Exemple de système de fichier UNIX :



Certains événements sportifs sont aussi représentables sous forme d'arbres. C'est le cas notamment des phases éliminatoires de la coupe du monde de football. *Exemple ci-dessous pour 2018.*



On retrouve bien ici **la racine (la finale)** et les nœuds qui en découlent (on remonte ici le temps à rebours !). Notons qu'il s'agit ici d'un type particulier d'arbre. **On parle d'arbre binaire**, et nous allons en examiner les caractéristiques !

Arbres binaires

Un arbre binaire est une structure hiérarchique qui (si elle n'est pas vide) **est constituée d'un nœud racine** (tout élément d'un arbre est appelé nœud) **et de deux sous-arbres** : un sous-arbre gauche et un sous-arbre droit. Les sous-arbres ne sont pas nécessairement identiques, cet arbre n'est pas symétrique. Notez aussi que dans un arbre binaire, chaque nœud possède tout aux plus deux fils (**il peut n'en avoir aucun, ou un seul**).

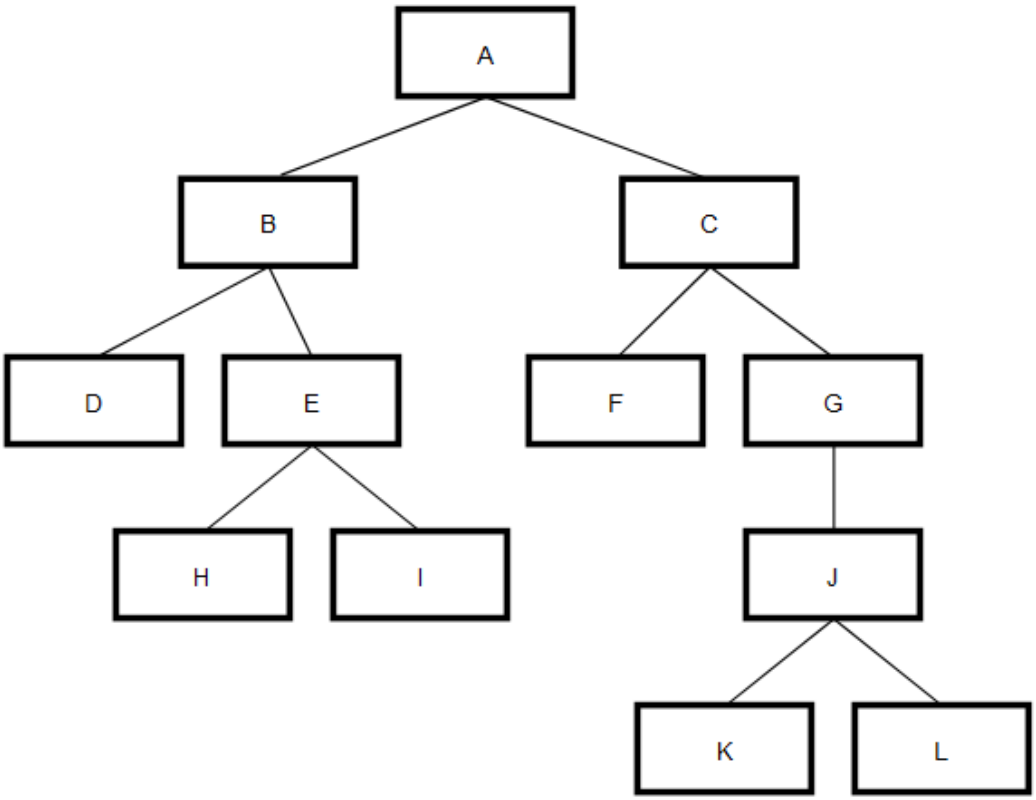
Il existe bien sûr un vocabulaire associé à ce type d'arbre. Par exemple, la taille d'un arbre correspond au nombre de ses nœuds, tandis qu'on désigne par hauteur profondeur ou niveau d'un nœud sa distance par rapport à la racine.

En effet, dans un arbre binaire, la profondeur d'un nœud ou d'une feuille X est le nombre de nœuds du chemin qui va de la racine à ce nœud X. Suivant la convention employée, la racine peut avoir une profondeur de 0 ou de 1. La profondeur des nœuds placés à sa suite en découle forcément !

La hauteur d'un arbre, quant à elle est la profondeur maximale des nœuds de cet arbre.

EXERCICE 01 **Caractéristiques d'un arbre binaire**

Soit l'arbre binaire suivant (on considère que sa racine A a une profondeur de 0) :



1. Quelle est la taille de cet arbre ?

2. Quelle est la profondeur du nœud E ?

3. Quelle est la hauteur de l'arbre ?

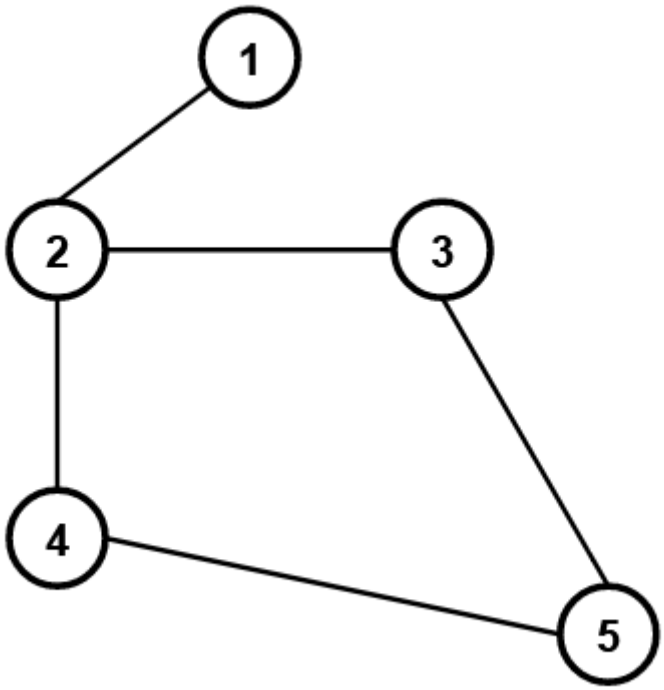
4. Que peut-on dire du nœud F ?

5. Que peut-on dire du nœud G ?

03 STRUCTURES DE DONNÉES
Structures relationnelles

Structures relationnelles : graphes

Voyons à présent le cas des graphes. Examinez cet exemple :



Cela peut représenter plusieurs choses : un réseau routier ou électrique, un groupe de personnes, des associations d'idées ou de concepts, etc.

L'important est de retenir que **chaque sommet du graphe est potentiellement relié aux autres par des arêtes.** **On parle ainsi de structure relationnelle.**

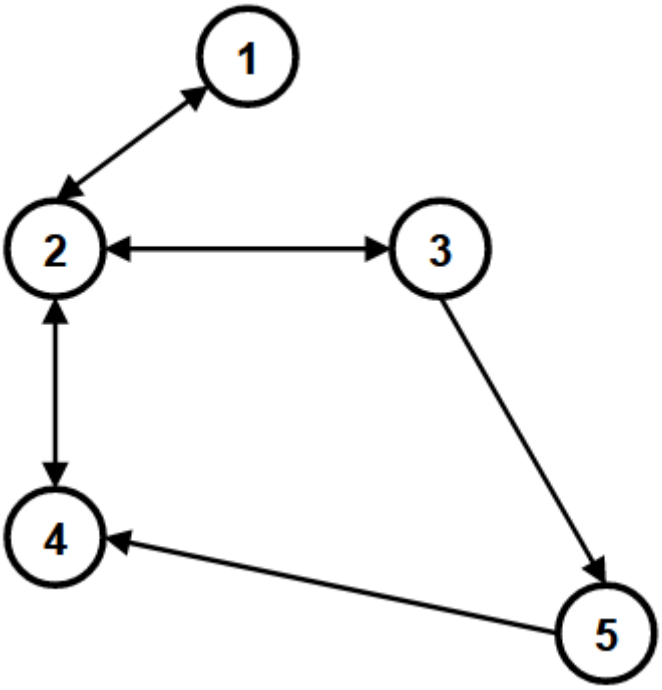
EXERCICE 02 **Constituer un graphe**

Dessinez le graphe de réseau social correspondant aux affirmations suivantes :

- Ida est amie de Eva, Luc et Bob.
- Max est ami de Luc et Léo.
- Léo est ami de Max et Eva.
- Eva est amie de Léo et Ida.
- Bob est ami d'Ida.
- Luc est ami de Max et Ida.

Graphe orienté

Souvent, les arêtes établissent des relations symétriques, mais il existe aussi des graphes dans lesquels ces arêtes sont associées à un sens de circulation. Pour reprendre l'exemple du réseau routier cela permet notamment de représenter les voies à sens unique (sur lesquels, si on les prend à contresens, on est confronté à un panneau « sens interdit »). Ce type particulier de graphe est désigné sous le terme de graphe orienté.



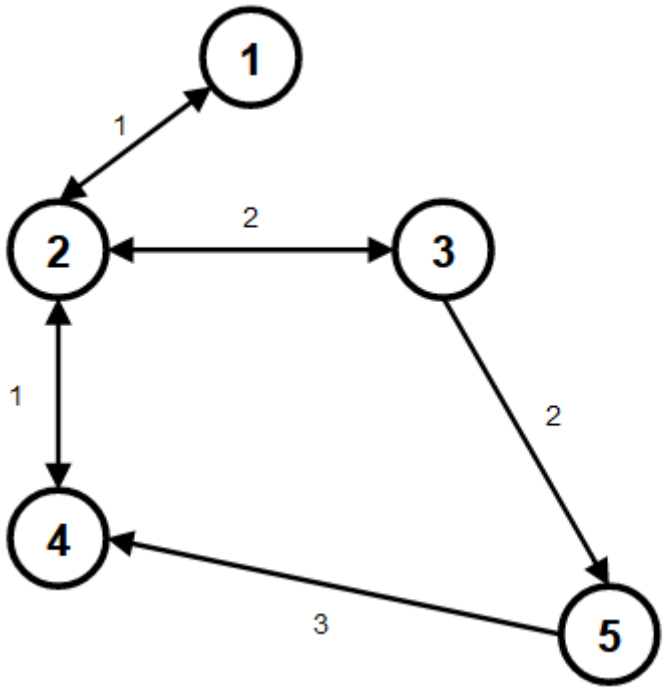
EXERCICE 03 **Parcours d'un graphe orienté**

1. Combien de parcours d'arêtes sont nécessaires pour atteindre le sommet 5 depuis le sommet 2 ?

2. Combien de parcours d'arêtes sont nécessaires pour atteindre le sommet 3 depuis le sommet 5 ?

Graphe pondéré

Les arêtes peuvent disposer d'une autre information supplémentaire : une valeur chiffrée qui peut représenter une quantité d'énergie ou une durée associée au parcours de cette arrête. Là encore, sur une représentation de réseau routier, ses valeurs numériques peuvent représenter des kilomètres à parcourir. Ce type de graphe est désigné sous le terme de graphe pondéré. Si vous avez étudié le chapitre de SNT consacré à la géolocalisation et au GPS, vous avez pu en rencontrer.



EXERCICE 04 **Parcours d'un graphe pondéré**

1. Quel est le coût minimum d'un déplacement du sommet 1 au sommet 5 ?

2. Quel est le coût minimum d'un déplacement du sommet 5 au sommet 3 ?

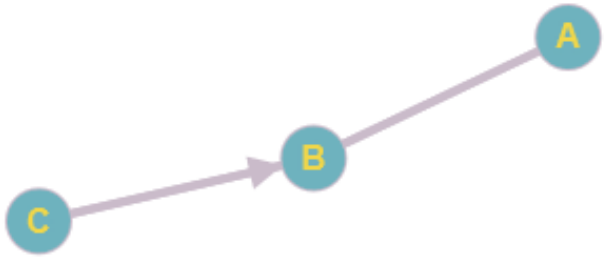
Matrices d'adjacence

Un tableau de tableaux est appelé matrice. On parle aussi de tableau à deux dimensions. Cette structure de données est très utile lorsqu'il s'agit d'implémenter en code le concept de graphe. Dans ces cas précis, on utilise ce qu'on appelle des matrices d'adjacences. Ce terme évoque le fait que l'on est intéressé par le fait de stocker, pour chaque sommet, les sommets qui lui sont adjacents, c'est-à-dire voisins.

Dans le cadre d'une matrice d'adjacence, le nombre de lignes est égal au nombre de colonnes : il s'agit de matrices carrées.

Pour ne pas encombrer notre tableau avec trop de paramètres, nous allons ici envisager **le cas des matrices d'adjacence appliquées aux graphes non pondérés** (par contre une telle matrice peut facilement gérer les graphes orientés).

Prenons donc un premier exemple de graphe :



Qu'avons-nous ici ? Un sommet A et un sommet B qui sont reliés à **double sens**, puis un sommet C qui est relié au sommet B en **sens unique.**

Comment pourrions-nous représenter ceci de manière numérique ?
Voici :

	A	**B**	**C**	**Explication**
A	0	1	0	Lien de A vers B
B	1	0	0	Lien de B vers A
C	0	1	0	Lien de C vers B

Ici nous avons un tableau très détaillé permettant de mieux comprendre le principe, mais en général une telle **matrice est simplement représentée ainsi :**

0, 1, 0
1, 0, 0
0, 1, 0

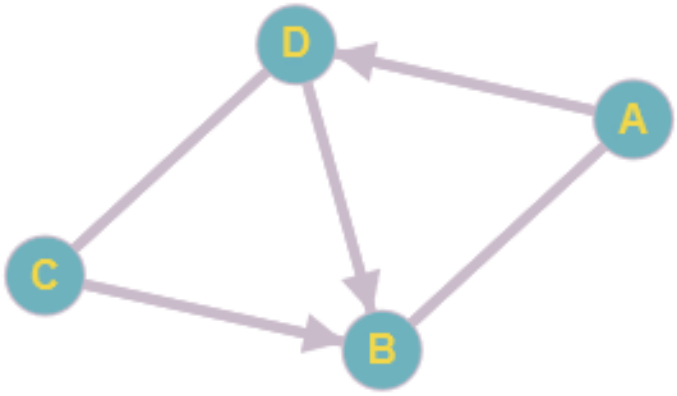
EXERCICE 05 D’une matrice vers un graphe

De la matrice d’adjacence suivante, dessinez le graphe correspondant :

0, 1, 0, 0
1, 0, 1, 0
0, 1, 0, 0
1, 1, 1, 0

EXERCICE 06 D’un graphe vers une matrice

Du graphe suivant, déduisez une matrice d’adjacence :



04 STRUCTURES DE DONNÉES
Bases de la programmation objet

Il existe plusieurs paradigmes de programmation. L'un d'eux, théorisé à partir des années 60 et très populaire durant ces dernières décennies, est désigné sous le terme de **programmation orientée objet** *(ou POO en français, OOP en anglais).* Pour en expliquer les fondements généraux, on prend souvent l'exemple d'une voiture. C'est en effet un bon moyen de mettre en évidence les briques de base de la POO : **objets, attributs, méthodes et instances.**

Pour commencer, attachons-nous à l'idée (presque platonicienne) de la voiture ! Sans penser marques ou modèles, qu'avons-nous dans une voiture standard ? Une structure générale, avec sa carrosserie, ses portières, ses vitres, etc. Puis nous avons des roues, un moteur, et à l'intérieur un tableau de bord ainsi qu'un volant. Bien sûr, des sièges, et des dizaines de petits éléments sur lesquels nous ne nous attarderons pas.

Ici, notre objet est la voiture. Nous l'écrirons sous forme de classe (un concept de base en POO) avec une majuscule : **Voiture.**
Ensuite, nous pouvons définir des attributs. Ces derniers contiennent les informations permettant de connaître les spécificités de notre voiture.

Prenons par exemple :
- ✓ Le nombre de portes ;
- ✓ La puissance du moteur ;
- ✓ Le type de moteur ;
- ✓ La présence d'un GPS ;

Ensuite, quelques méthodes permettront de définir les actions envisageables pour cet objet, par exemple :
- ✓ Allumer ou éteindre le moteur ;
- ✓ Avancer, reculer, etc. ;
- ✓ Tourner le volant à gauche ou à droite ;
- ✓ Allumer ou éteindre les phares ou la radio, etc.

Certaines méthodes peuvent aussi servir à lire des informations relatives à l'état de la voiture :
- ✓ Consulter le niveau d'essence ou le niveau d'huile ;
- ✓ Consulter la vitesse actuelle ou le nombre de tours par minute ;
- ✓ Etc.

Avec une telle classe, nous avons donc le modèle d'une Voiture type. On peut ensuite déclarer l'initialisation **d'une variable « maVoiture » de type Voiture** ! On vient de créer une instance de notre objet Voiture. Cette instance disposera de ses propres valeurs d'attributs, qui vont nécessairement varier avec le temps (vitesse, carburant, position, etc.)

EN PASSANT, SUR LE WEB…

Exemple d'implémentation en PHP

Nous procéderons à des implémentations d'un exemple POO en Python au cours du module 3 du programme NSI de terminale.

Néanmoins, les exemples abondent sur internet, bien sûr, si vous voulez déjà jeter un coup d'œil à une implémentation à PHP, vous pouvez en consulter une à cette adresse :

https://www.vulgarisation-informatique.com/php-poo.php

Question : voyez-vous l'utilité de la méthode « __construct » présentée dans l'exemple ?

.....

.....

Il s'agit d'un classique de la POO ! Une méthode appelée « constructeur » et qui est automatiquement exécutée lorsqu'une instance de l'objet est créée. Elle sert donc à initialiser les attributs de la classe. On retrouve l'équivalent « __destruct » (destructeur) qui est appelé automatiquement lorsque l'on supprime une instance.

LE TEMPS DU BILAN

NOTIONS ET CONCEPTS A RETENIR

Parmi les structures de données que nous avons explorées dans ce chapitre, nous avons vu **la liste** qui comporte une tête et une queue ainsi que différentes opérations permettant de créer une liste ou d'y ajouter et supprimer des éléments, etc.

Nous nous sommes aussi attardés sur **les piles**, qui partagent des caractéristiques avec les listes mais sont remarquables dans le sens où elle fonctionne sur le principe **LIFO** : Last In First Out (le dernier élément entré sera le premier à sortir). Sur le même modèle, nous avons vu les **files**, reposant quant à elles sur le principe **FIFO** : First In First Out (le premier élément entré sera le premier à sortir).

Ces **types abstraits** peuvent être implémentés de plusieurs manières suivant le langage de programmation utilisée, ou simplement suivant les besoins algorithmiques du moment (le module 3 approfondira cet aspect).

Nous avons vu d'autres types abstraits ! Ainsi, **un arbre** est composé de **nœuds** (avec un nœud initial appelé **racine**). Un segment qui relie deux nœuds est appelé **arête**. Les nœuds situés en-deçà d'un aux parents sont appelés **fils**, et un nœud qui n'a pas de fils est une **feuille** ! Il existe un type spécifique d'arbre nommé **arbre binaire** qui dispose de caractéristiques plus précises : chacune dispose au maximum que de deux fils.

Notons aussi l'existence de sous-arbres : les fils d'un d'entre eux sont des arbres (on peut parler de structure récursive) !

Nous avons aussi évoqué les graphes. Ces derniers permettent de représenter des **réseaux**, qu'ils soient sociaux, routiers, etc. On y parle de **sommets** et d'**arêtes**. Une **chaîne** est une suite d'arêtes consécutives, on la désigne par les lettres des sommets qui la constituent (exemple : ABDG). On appelle **cycle** une chaîne spécifique qui commence et se termine avec le même sommet. Ces graphes sont dits **pondérés** si chaque arête dispose d'un poids spécifique, et **orientés** si les arêtes comprennent un sens de circulation.

On peut en outre implémenter un graphe à l'aide d'une matrice d'adjacence. Il s'agit d'un tableau à double entrée, constitué d'autant de lignes que de colonnes. Chacune correspond à un sommet du graphe, Et c'est la même chose pour chaque colonne ! Une ligne permet donc de savoir, pour chaque sommet, s'il existe un chemin entre les deux sommets considérés.

Ce chapitre a aussi passé en revue la notion **de dictionnaire** (aussi appelés **tableaux associatifs**). On y associe chaque élément à une clé. On parle alors de **couple clé valeur** ! Enfin, nous avons effectué une première approche de la **programmation orientée objet**, ou **POO** (qui sera quelque peu approfondie en module 3). Nous avons appris les concepts **d'objet** (comprenant **attributs** et **méthodes**) et d'**instance** (occurrence précise d'un objet).

Concluons:

Nous avons passé en revue de nombreuses notions ! Et encore, il ne s'agissait que de les aborder d'un point de vue théorique avant de les expérimenter sous un aspect plus technique en fin d'année. Tout ceci vous sera néanmoins très utile pour ce qui est d'intégrer des principes fondamentaux en programmation et en gestion de bases de données.

N'hésitez pas vous entraîner à manipuler des graphes (avec leurs matrices), des arbres, des dictionnaires ou des listes de différents types. Si le vocabulaire est important à connaître, il en va bien sûr de même de votre capacité à appliquer ces principes dans des exercices de natures diverses. La section « clés du bac » située à la fin de chaque chapitre vous permettra tout au long de l'année de vérifier votre pratique de ces différentes notions.

Vous pouvez maintenant faire et envoyer le devoir n°1

Maintenant, en route pour le second thème majeur de ce module : les bases de données ! Nous commencerons ce sujet en découvrant le concept de modèle relationnel avec le chapitre deux.