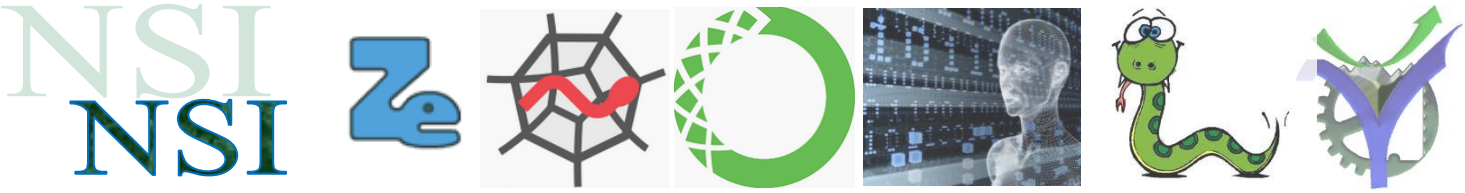
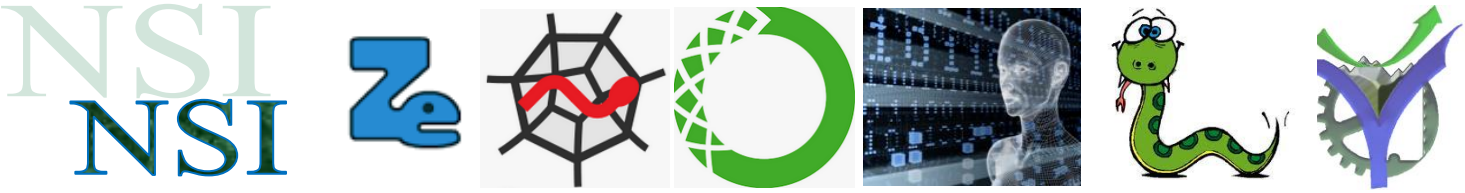


La spécialité NSI de terminale

Sommaire :



1. Les objectifs de la formation

L'enseignement de spécialité de numérique et sciences informatiques du cycle terminal de la voie générale vise l'appropriation des fondements de l'informatique pour préparer les élèves à une poursuite d'études en les formant à la pratique d'une démarche scientifique et en développant leur appétence pour des activités de recherche.

L'objectif de cet enseignement général est l'appropriation des concepts et des méthodes qui fondent l'informatique, dans ses dimensions scientifiques et techniques. Il s'appuie sur l'universalité de quatre concepts fondamentaux et la variété de leurs interactions :

- les **données**, qui représentent sous une forme numérique unifiée des informations très diverses : textes, images, sons, mesures physiques, sommes d'argent, etc. ;
- les **algorithmes**, qui spécifient de façon abstraite et précise des traitements à effectuer sur les données à partir d'opérations élémentaires ;
- les **langages**, qui permettent de traduire les algorithmes abstraits en **programmes** textuels ou graphiques de façon à ce qu'ils soient exécutables par les machines ;
- les **machines**, et leurs systèmes d'exploitation, qui permettent d'exécuter des programmes en enchaînant un grand nombre d'instructions simples, assurent la persistance des données par leur stockage et gèrent les communications. Y sont inclus les **objets connectés** et les **réseaux**.

À ces concepts s'ajoute un élément transversal : les **interfaces** qui permettent la communication, la collecte des données et la commande des systèmes.

Les moyens utilisés

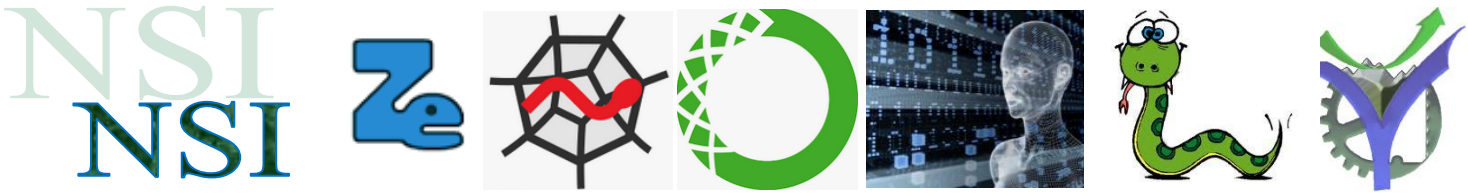
Cet enseignement se déploie en mettant en activité les élèves, **sous des formes variées** qui permettent de développer des compétences transversales :

- faire preuve d'autonomie, d'initiative et de créativité ;
- présenter un problème ou sa solution, développer une argumentation dans le cadre d'un débat ;
- coopérer au sein d'une équipe dans le cadre d'un projet ;
- rechercher de l'information, partager des ressources ;
- faire un usage responsable et critique de l'informatique.

Fournitures

- **Un cahier à spirale au format 24 x 32** de préférence à petits carreaux nombre de pages au choix
- Une calculatrice (pas de besoin particulier, voir le modèle conseillé en math)
- Une clé USB sera utile dans le cadre du lycée en seconde, elle pourra être partagée avec les autres enseignements.

  Il est impératif d'étiqueter cette clé avec le nom de famille de l'élève ainsi qu'avec le nom de la classe et du groupe.



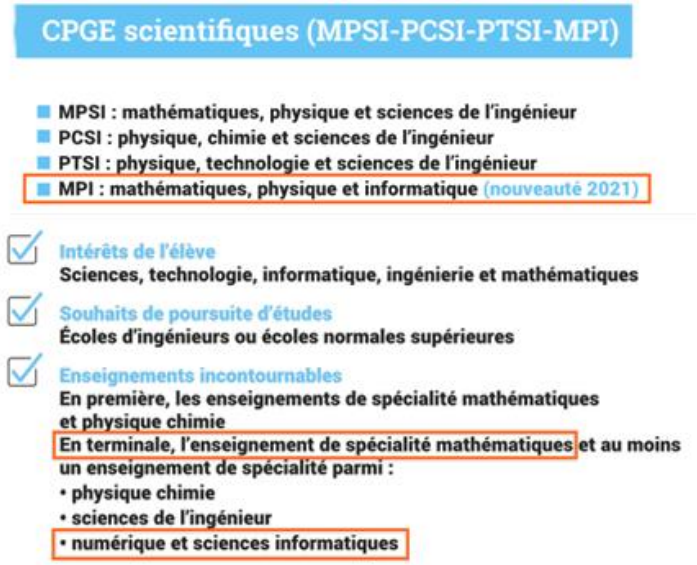
Après le Bac : poursuivre dans une voie centrée autour de l'informatique :

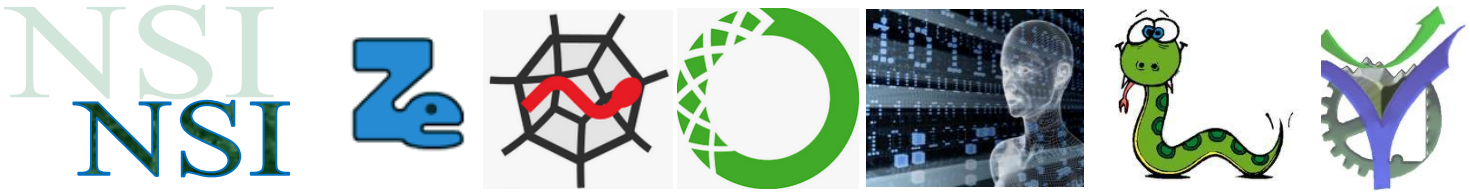
- au niveau BTS, IUT, Licence Pro

- licence et master de mathématique et informatique

- classes préparatoires aux grandes écoles CPGE scientifiques.

Exemple de recommandation pour l'intégration d'une classe de prépa scientifique (juillet 2019)





2. Les épreuves de baccalauréat en NSI :

Pour la spécialité de terminale les modalités de l'examen final sont :

- un écrit de 3H30 comptant pour 12 / 20 de la note finale
  L'écrit consiste à résoudre trois exercices choisis parmi les cinq proposés dans le sujet.

  La partie écrite 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.

- une partie pratique de programmation d'1H comptant pour 8 / 20
  Cette partie consiste à résoudre deux exercices sur ordinateurs, chacun comptant pour 4 points.

  La partie pratique consiste en la résolution de deux exercices sur ordinateur, chacun étant noté sur 4 points.
  Le candidat est évalué sur la base d'un dialogue avec un professeur-examinateur. Un examinateur évalue au maximum quatre élèves.
  L'examinateur ne peut pas évaluer un élève qu'il a eu en classe durant l'année en cours.
  L'évaluation de cette partie se déroule au cours du deuxième trimestre pendant la période de l'épreuve écrite de spécialité.

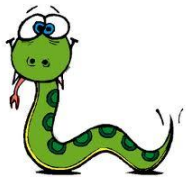
  - **Premier exercice**

  Le premier exercice consiste à programmer un algorithme figurant explicitement au programme, ne présentant pas de difficulté particulière, dont on fournit une spécification. Il s'agit donc de restituer un algorithme rencontré et travaillé à plusieurs reprises en cours de formation. Le sujet peut proposer un jeu de test avec les réponses attendues pour permettre au candidat de vérifier son travail.

  - **Deuxième exercice**

  Pour le second exercice, un programme est fourni au candidat. Cet exercice ne demande pas l'écriture complète d'un programme, mais permet de valider des compétences de programmation suivant des modalités variées : le candidat doit, par exemple, compléter un programme « à trous » afin de répondre à une spécification donnée, ou encore compléter un programme pour le documenter, ou encore compléter un programme en ajoutant des assertions, etc.





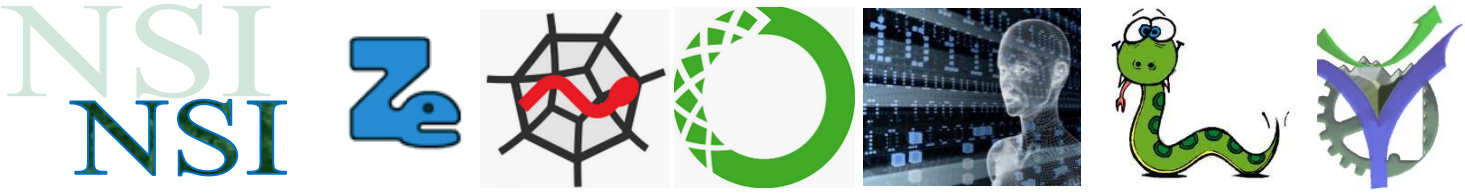

3. Le programme

3.1 Histoire de l'informatique

Contenus	Capacités attendues	Commentaires
Événements clés de l'histoire de l'informatique.	Situer dans le temps les principaux événements de l'histoire de l'informatique et leurs protagonistes. Identifier l'évolution des rôles relatifs des logiciels et des matériels.	Ces repères viennent compléter ceux qui ont été introduits en première. Ces repères historiques sont construits au fur et à mesure de la présentation des concepts et techniques.

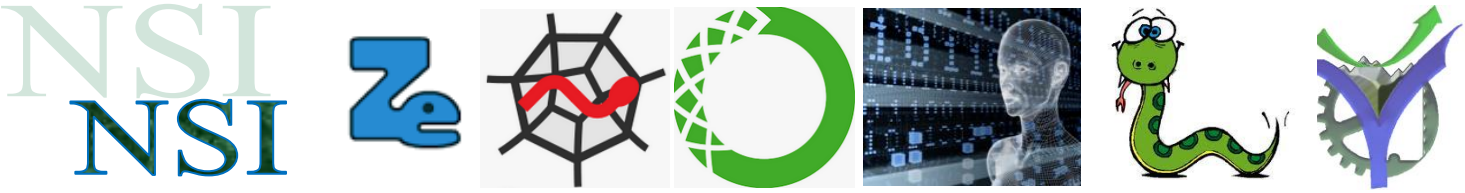
Cette rubrique transversale se décline dans chacune des cinq autres.

Comme tous les concepts scientifiques et techniques, ceux de l'informatique ont une histoire et ont été forgés par des personnes. Les algorithmes sont présents dès l'Antiquité, les machines à calculer apparaissent progressivement au XVIIe siècle, les sciences de l'information sont fondées au XIXe siècle, mais c'est en 1936 qu'apparaît le concept de machine universelle, capable d'exécuter tous les algorithmes, et que les notions de machine, algorithme, langage et information sont pensées comme un tout cohérent. Les premiers ordinateurs ont été construits en 1948 et leur puissance a ensuite évolué exponentiellement. Parallèlement, les ordinateurs se sont diversifiés dans leurs tailles, leurs formes et leurs emplois : téléphones, tablettes, montres connectées, ordinateurs personnels, serveurs, fermes de calcul, méga-ordinateurs. Le réseau Internet, développé depuis 1969, relie aujourd'hui ordinateurs et objets connectés.



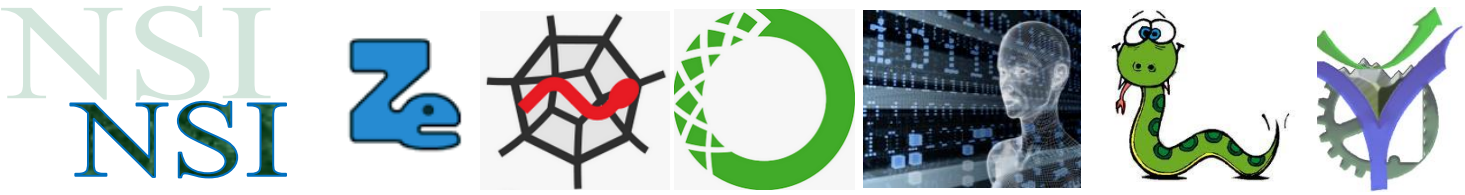
3.2 Structure de données

Contenus	Capacités attendues	Commentaires
Structures de données, interface et implémentation.	Spécifier une structure de données par son interface. Distinguer interface et implémentation. Écrire plusieurs implémentations d'une même structure de données.	L'abstraction des structures de données est introduite après plusieurs implémentations d'une structure simple comme la file (avec un tableau ou avec deux piles).
Vocabulaire de la programmation objet : classes, attributs, méthodes, objets.	Écrire la définition d'une classe. Accéder aux attributs et méthodes d'une classe.	On n'aborde pas ici tous les aspects de la programmation objet comme le polymorphisme et l'héritage.
Listes, piles, files : structures linéaires. Dictionnaires, index et clé.	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.	On distingue les modes FIFO *(first in first out)* et LIFO *(last in first out)* des piles et des files.
Arbres : structures hiérarchiques. Arbres binaires : nœuds, racines, feuilles, sous-arbres gauches, sous-arbres droits.	Identifier des situations nécessitant une structure de données arborescente. Évaluer quelques mesures des arbres binaires (taille, encadrement de la hauteur, etc.).	On fait le lien avec la rubrique « algorithmique ».
Graphes : structures relationnelles. Sommets, arcs, arêtes, graphes orientés ou non orientés.	Modéliser des situations sous forme de graphes. Écrire les implémentations correspondantes d'un graphe : matrice d'adjacence, liste de successeurs/de prédécesseurs. Passer d'une représentation à une autre.	On s'appuie sur des exemples comme le réseau routier, le réseau électrique, Internet, les réseaux sociaux. Le choix de la représentation dépend du traitement qu'on veut mettre en place : on fait le lien avec la rubrique « algorithmique ».



3.3 Base de données

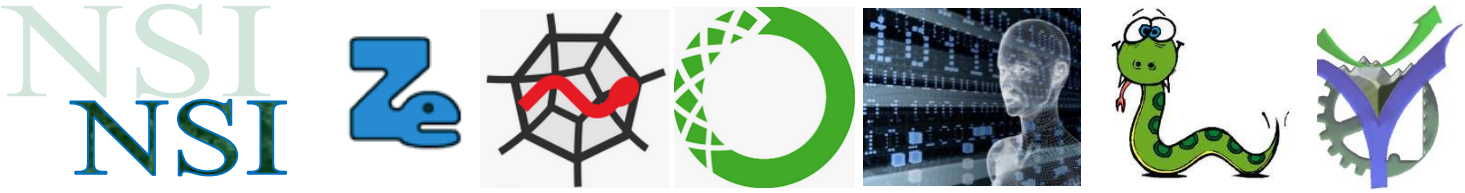
Contenus	Capacités attendues	Commentaires
Modèle relationnel : relation, attribut, domaine, clef primaire, clef étrangère, schéma relationnel.	Identifier les concepts définissant le modèle relationnel.	Ces concepts permettent d'exprimer les contraintes d'intégrité (domaine, relation et référence).
Base de données relationnelle.	Savoir distinguer la structure d'une base de données de son contenu. Repérer des anomalies dans le schéma d'une base de données.	La structure est un ensemble de schémas relationnels qui respecte les contraintes du modèle relationnel. Les anomalies peuvent être des redondances de données ou des anomalies d'insertion, de suppression, de mise à jour. On privilégie la manipulation de données nombreuses et réalistes.
Système de gestion de bases de données relationnelles.	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.	Il s'agit de comprendre le rôle et les enjeux des différents services sans en détailler le fonctionnement.
Langage SQL : requêtes d'interrogation et de mise à jour d'une base de données.	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.	On peut utiliser DISTINCT, ORDER BY ou les fonctions d'agrégation sans utiliser les clauses GROUP BY et HAVING.



3.4 Architectures matérielles, systèmes d'exploitation et réseaux

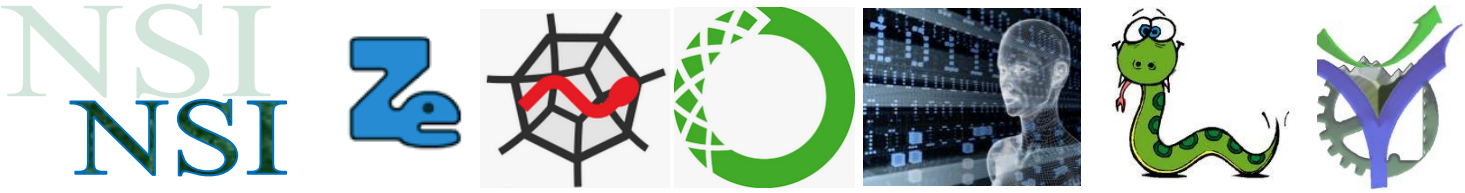
Contenus	Capacités attendues	Commentaires
Composants intégrés d'un système sur puce.	Identifier les principaux composants sur un schéma de circuit et les avantages de leur intégration en termes de vitesse et de consommation.	Le circuit d'un téléphone peut être pris comme un exemple : microprocesseurs, mémoires locales, interfaces radio et filaires, gestion d'énergie, contrôleurs vidéo, accélérateur graphique, réseaux sur puce, etc.
Gestion des processus et des ressources par un système d'exploitation.	Décrire la création d'un processus, l'ordonnancement de plusieurs processus par le système. Mettre en évidence le risque de l'interblocage *(deadlock)*.	À l'aide d'outils standard, il s'agit d'observer les processus actifs ou en attente sur une machine. Une présentation débranchée de l'interblocage peut être proposée.

Protocoles de routage.	Identifier, suivant le protocole de routage utilisé, la route empruntée par un paquet.	En mode débranché, les tables de routage étant données, on se réfère au nombre de sauts (protocole RIP) ou au coût des routes (protocole OSPF). Le lien avec les algorithmes de recherche de chemin sur un graphe est mis en évidence.
Sécurisation des communications.	Décrire les principes de chiffrement symétrique (clef partagée) et asymétrique (avec clef privée/clef publique). Décrire l'échange d'une clef symétrique en utilisant un protocole asymétrique pour sécuriser une communication HTTPS.	Les protocoles symétriques et asymétriques peuvent être illustrés en mode débranché, éventuellement avec description d'un chiffrement particulier. La négociation de la méthode chiffrement du protocole SSL (*Secure Sockets Layer*) n'est pas abordée.



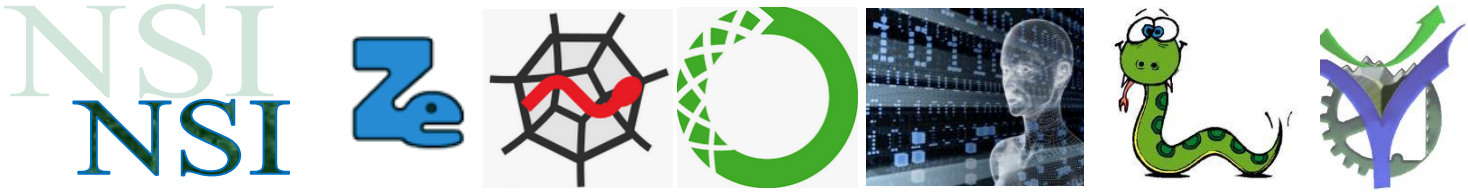
3.5 Langage et programmation

Contenus	Capacités attendues	Commentaires
Notion de programme en tant que donnée. Calculabilité, décidabilité.	Comprendre que tout programme est aussi une donnée. Comprendre que la calculabilité ne dépend pas du langage de programmation utilisé. Montrer, sans formalisme théorique, que le problème de l'arrêt est indécidable.	L'utilisation d'un interpréteur ou d'un compilateur, le téléchargement de logiciel, le fonctionnement des systèmes d'exploitation permettent de comprendre un programme comme donnée d'un autre programme.
Récursivité.	Écrire un programme récursif. Analyser le fonctionnement d'un programme récursif.	Des exemples relevant de domaines variés sont à privilégier.
Modularité.	Utiliser des API (*Application Programming Interface*) ou des bibliothèques. Exploiter leur documentation. Créer des modules simples et les documenter.	
Paradigmes de programmation.	Distinguer sur des exemples les paradigmes impératif, fonctionnel et objet. Choisir le paradigme de programmation selon le champ d'application d'un programme.	Avec un même langage de programmation, on peut utiliser des paradigmes différents. Dans un même programme, on peut utiliser des paradigmes différents.
Mise au point des programmes. Gestion des bugs.	Dans la pratique de la programmation, savoir répondre aux causes typiques de bugs : problèmes liés au typage, effets de bord non désirés, débordements dans les tableaux, instruction conditionnelle non exhaustive, choix des inégalités, comparaisons et calculs entre flottants, mauvais nommage des variables, etc.	On prolonge le travail entrepris en classe de première sur l'utilisation de la spécification, des assertions, de la documentation des programmes et de la construction de jeux de tests. Les élèves apprennent progressivement à anticiper leurs erreurs.



3.6 Algorithmique

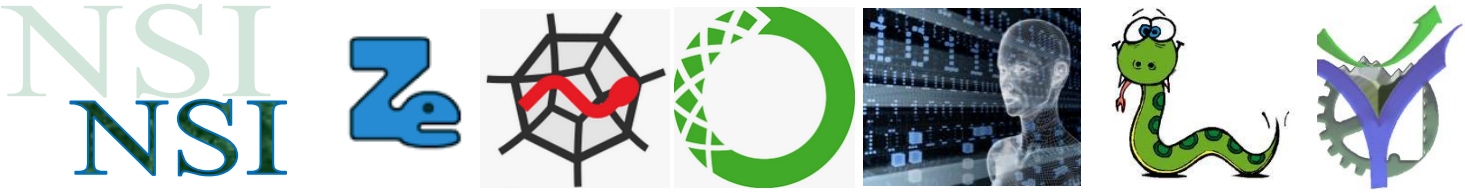
Contenus	Capacités attendues	Commentaires
Algorithmes sur les arbres binaires et sur les arbres binaires de recherche.	Calculer la taille et la hauteur d'un arbre. Parcourir un arbre de différentes façons (ordres infixe, préfixe ou suffixe ; ordre en largeur d'abord). Rechercher une clé dans un arbre de recherche, insérer une clé.	Une structure de données récursive adaptée est utilisée. L'exemple des arbres permet d'illustrer la programmation par classe. La recherche dans un arbre de recherche équilibré est de coût logarithmique.
Algorithmes sur les graphes.	Parcourir un graphe en profondeur d'abord, en largeur d'abord. Repérer la présence d'un cycle dans un graphe. Chercher un chemin dans un graphe.	Le parcours d'un labyrinthe et le routage dans Internet sont des exemples d'algorithme sur les graphes. L'exemple des graphes permet d'illustrer l'utilisation des classes en programmation.
Méthode « diviser pour régner ».	Écrire un algorithme utilisant la méthode « diviser pour régner ».	La rotation d'une image bitmap d'un quart de tour avec un coût en mémoire constant est un bon exemple. L'exemple du tri fusion permet également d'exploiter la récursivité et d'exhiber un algorithme de coût en $n \log_2 n$ dans les pires des cas.
Programmation dynamique.	Utiliser la programmation dynamique pour écrire un algorithme.	Les exemples de l'alignement de séquences ou du rendu de monnaie peuvent être présentés. La discussion sur le coût en mémoire peut être développée.
Recherche textuelle.	Étudier l'algorithme de Boyer-Moore pour la recherche d'un motif dans un texte.	L'intérêt du prétraitement du motif est mis en avant. L'étude du coût, difficile, ne peut être exigée.



3.7 Langages et programmation

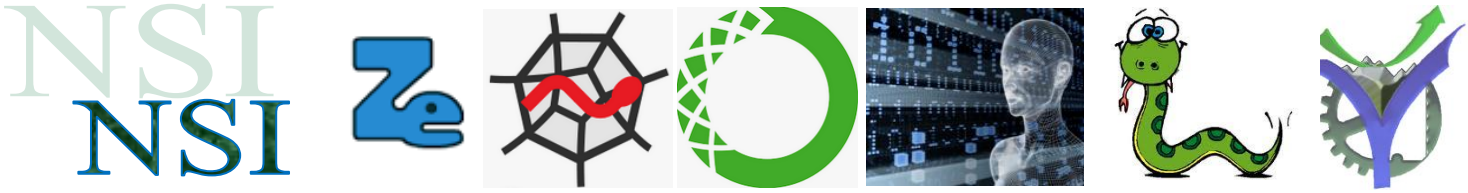
Contenus	Capacités attendues	Commentaires
Constructions élémentaires	Mettre en évidence un corpus de constructions élémentaires.	Séquences, affectation, conditionnelles, boucles bornées, boucles non bornées, appels de fonction.
Diversité et unité des langages de programmation	Repérer, dans un nouveau langage de programmation, les traits communs et les traits particuliers à ce langage.	Les manières dont un même programme simple s'écrit dans différents langages sont comparées.
Spécification	Prototyper une fonction. Décrire les préconditions sur les arguments. Décrire des postconditions sur les résultats.	Des assertions peuvent être utilisées pour garantir des préconditions ou des postconditions.
Mise au point de programmes	Utiliser des jeux de tests.	L'importance de la qualité et du nombre des tests est mise en évidence. Le succès d'un jeu de tests ne garantit pas la correction d'un programme.
Utilisation de bibliothèques	Utiliser la documentation d'une bibliothèque.	Aucune connaissance exhaustive d'une bibliothèque particulière n'est exigible.

⇦



3.8 Algorithmique

Contenus	Capacités attendues	Commentaires
Parcours séquentiel d'un tableau	Écrire un algorithme de recherche d'une occurrence sur des valeurs de type quelconque. Écrire un algorithme de recherche d'un extremum, de calcul d'une moyenne.	On montre que le coût est linéaire.
Tris par insertion, par sélection	Écrire un algorithme de tri. Décrire un invariant de boucle qui prouve la correction des tris par insertion, par sélection.	La terminaison de ces algorithmes est à justifier. On montre que leur coût est quadratique dans le pire cas.
Algorithme des k plus proches voisins	Écrire un algorithme qui prédit la classe d'un élément en fonction de la classe majoritaire de ses k plus proches voisins.	Il s'agit d'un exemple d'algorithme d'apprentissage.
Recherche dichotomique dans un tableau trié	Montrer la terminaison de la recherche dichotomique à l'aide d'un variant de boucle.	Des assertions peuvent être utilisées. La preuve de la correction peut être présentée par le professeur.
Algorithmes gloutons	Résoudre un problème grâce à un algorithme glouton.	Exemples : problèmes du sac à dos ou du rendu de monnaie. Les algorithmes gloutons constituent une méthode algorithmique parmi d'autres qui seront vues en terminale.



4. Ressources

4.1 ProNote

Toujours vérifier les messages, travaux à faire, QCM et cahier de texte tout au long de l'année.

4.2 Sites dédiés

PG : pour les documents de terminales voir sur le site :

http://sti2dvox.patgue.com

ChF : Open ClassRoom (idem année de 1ère)

4.3 Tout le contenu de la classe de 1ère NSI PG :

Voir ici (! 8 Go)

http://sti2dvox.patgue.com/NSI_1ERE/Doc_1ere_NSI_PG_2019-2020.7z