L'INFORMATIQUE, UN APPRENTISSAGE DE PLUS OU UNE PISTE AU SERVICE D'AUTRES APPRENTISSAGES ?

Marie DUFLOT
Maître de conférences, Université de Lorraine
LORIA & INRIA Nacy Grand Est
marie.duflot-kremer@loria.fr

Résumé
Avec les changements de programmes, une introduction à l'algorithmique et la programmation arrive jusqu'en primaire. En effet, le programme des cycles 2 et 3 contient une partie d'initiation à la programmation, au travers de déplacements relatifs ou absolus. Pour ce faire, diverses activités avec ou sans ordinateurs sont suggérées afin d'acquérir et de perfectionner des premières notions de programmation.
Au travers d'exemples concrets, et avec un lien vers la formation Class'Code qui mêle apprentissage du code et de concepts au travers d'activités sans ordinateur, nous avons vu en quoi des activités d'informatique « débranchée », popularisées entre autres par l'équipe à l'origine du site Computer Science Unplugged (Bell et al. 1992) et Roberto Di Cosmo (Di Cosmo, 2015), permettent de travailler de manière ludique des concepts fondamentaux, tout en manipulant des objets concrets, en développant la collaboration et en permettant de structurer et transmettre ses idées. Nous avons également vu comment ces activités s'articulent avec des compétences, transversales ou non, que les enfants acquièrent et exploitent dans le cursus scolaire.

Cet article présente et complète ce qui a été fait lors de l'atelier. La section I présente la « pensée informatique », illustrant en quoi elle est déjà présente dans l'enseignement, et montre comment l'informatique s'insère dans le programme de primaire, en tant que telle ou comme moyen de travailler d'autres compétences. Dans un deuxième temps nous présentons en Section II quelques activités sans ordinateur qui permettent de découvrir et comprendre des principes informatiques de manière ludique. Nous présentons enfin en Section III un résumé des retours des participants ainsi que quelques conclusions tirées de cet atelier.

I - LA PENSEE INFORMATIQUE A L'ECOLE ELEMENTAIRE

1 Qu'est-ce que la pensée informatique ?

La pensée informatique, ou pensée computationnelle (*computational thinking* en anglais) est une notion, dont l'invention est attribuée à Seymour Papert (Papert, 1980) et popularisée entre autres par Jeannette Wing (Wing, 2006). En quelques mots, elle représente la capacité de formuler un problème, puis de concevoir et décrire sa solution en des étapes assez simples et précises pour qu'elles puissent être effectuées par un ordinateur. Pour se faire une idée un peu plus précise, voici une liste non exhaustive d'ingrédients impliqués dans la pensée computationnelle :

- La spécification : décrire précisément le problème, les informations dont on part ainsi que ce qu'on veut obtenir.
- L'algorithme : trouver quelles étapes permettent de résoudre le problème, convenir de quelles actions élémentaires sont adaptées et trouver quoi faire et dans quel ordre.
- La correction des bugs : une fois l'algorithme mis en œuvre, être capable d'identifier et de corriger les problèmes, souvent au travers de tests.
- La décomposition : résoudre un problème difficile en le découpant en plusieurs grandes étapes résolues séparément, chaque partie étant plus facile à comprendre et à corriger, puis recomposer ces étapes en une solution complète.

- La réutilisation : tirer avantage de la modularité en réutilisant un ou plusieurs modules de la solution d'un problème pour en résoudre un autre, similaire. Comprendre ce qui est utilisable directement et ce qui doit être adapté.
- La créativité : chercher la solution à un problème dans différentes directions, ou chercher une solution plus élégante ou plus facile à comprendre pour un même problème.
- L'optimisation : chercher à rendre une solution la plus efficace possible (en temps d'exécution par exemple).

Un des points remarquables de cette notion est que, si elle a été introduite en lien avec la programmation, elle peut s'appliquer à des domaines très variés. On se doute bien que la résolution d'un problème de mathématiques par exemple va mobiliser une bonne partie de ces compétences, mais on utilise aussi des aspects de pensée computationnelle dans la cuisine ou le bâtiment. Si j'ai une recette pour 4 personnes et que finalement nous sommes 6, est-ce que je change la quantité d'ingrédients ? la température du four ? le temps de cuisson ? Je dois construire un pont contenant 6 piles, puis-je les construire en même temps ? Combien de personnes au minimum doivent être là pour pouvoir le réaliser ? Et si j'ai 10, 20 ou 30 personnes de plus, est-ce que ça va vraiment accélérer le travail ?

En clair la pensée informatique est déjà très présente dans les enseignements scolaires, quand on construit un raisonnement logique, quand on explique les choses de manière détaillée, ou qu'on utilise la solution d'un problème connu pour en résoudre un nouveau. Nous utilisons tous et toutes la pensée informatique sans forcément le savoir, et elle n'est pas réservée aux informaticien•ne•s, loin de là.

2 L'informatique dans les programme des cycles 2 et 3

Depuis 2016, l'initiation à la programmation a fait son entrée dans les programmes des cycles 2 et 3. Contrairement aux programmes du cycle 4, cette nouvelle partie est ici intégrée au programme de géométrie et se concentre sur les déplacements. « Il s'agit de savoir coder ou décoder pour prévoir ou représenter des déplacements, de programmer les déplacements d'un robot ou ceux d'un personnage sur un écran. Des activités géométriques, consistant en la construction de figures simples ou de figures composées de figures simples, sont également proposées.» (EDUSCOL, 2016).

Le document contenant les ressources d'accompagnement du programme de mathématiques propose pour l'initiation à la programmation trois pistes pour les stratégies d'enseignement.

- Tout d'abord la démarche. Les enseignantes et enseignants sont invité•e•s à mettre les élèves dans une démarche de projet, en se basant sur cette notion de déplacement, et les inviter à expliciter leurs programmes. Les objectifs visés sont l'implication, l'abstraction et l'autonomie.
- Ensuite le langage qui doit être précis, ne pas laisser d'ambiguïté mais permettre de faire toutes les actions nécessaires.
- Enfin les activités doivent travailler sur le **déplacement absolu** (pour lequel l'effet d'une commande ne dépend pas de l'orientation du personnage/robot) et le **déplacement relatif** (si on avance de 3 pas, selon son orientation on ne va pas dans la même direction).

Pour atteindre ces objectifs, le document propose plusieurs pistes à combiner librement. Tout d'abord des activités « débranchées » donc sans ordinateur, puis avec des robots (Bee-Bot, Blue-Bot, Thymio…), et enfin sur machine, sur des sites web (comme Blockly ou code.org) ou avec un logiciel téléchargeable et utilisable sans connexion internet (Scratch, ScratchJr).

L'activité du robot (voir section **Erreur ! Nous n'avons pas trouvé la source du renvoi.**) s'intègre parfaitement dans ce cadre, car elle permet de travailler les déplacements relatifs et absolus, ainsi que de penser au passage entre ces deux langages. Elle est de fait très proche de l'activité de la fusée suggérée sur le site EDUSCOL (sur les déplacements absolus) et de la tournée du facteur (sur les déplacements relatifs).

Outre les documents disponibles sur le site EDUSCOL, différentes initiatives, individuelles ou institutionnelles permettent aux enseignants de se former sur cette nouvelle discipline. Parmi elles,

Class'Code est un projet financé dans le cadre des PIA[1] qui réunit de nombreux partenaires, publics ou privés, afin de fournir une formation hybride gratuite à la pensée informatique (Class'Code, 2015). Ce projet mélange cours en ligne (MOOC) et temps de rencontres entre apprenants pour donner aux enseignants mais aussi aux éducateurs/animateurs un bagage pour initier les 8-14 ans à la pensée informatique. Et ce bagage proposé par Class'Code est à la fois théorique (éléments d'histoire de l'informatique, explications sur le binaire ou le fonctionnement d'Internet…) et pratique (expérimentation et ressources sur des activités informatiques, avec ou sans ordinateur).

3 Des activités informatiques au service des apprentissages

Lorsqu'on parle d'informatique, la première chose qui vient à l'esprit de nos interlocuteurs est l'ordinateur, l'outil informatique par excellence. Si son utilité dans la salle de classe (utilisation d'un TBI, de logiciels de traitement de texte, accès à des ressources en ligne…) est évidente, il ne faut pas oublier que l'informatique ne se limite pas à une utilisation (si experte soit-elle) de ces outils au travers de logiciels existants. En effet, l'informatique est avant tout un ensemble de concepts, ceux-là mêmes qui permettent à nos outils informatiques de fonctionner, et c'est autour de ces concepts que tournent les activités qui ont été présentées dans l'atelier.

Comme on s'intéresse aux concepts, on peut donc les présenter en se détachant de l'ordinateur, aux travers de jeux, d'énigmes, voire même de tours de magie. Pour cela, on utilise des objets hétéroclites comme des draps, des allumettes, des jetons, des objets en carton ou des cartes. Les participantes et participants sont invité•e•s à manipuler ces objets, à se déplacer, à échanger, à collaborer afin d'atteindre un objectif commun.

Les activités renforcent/mobilisent de ce fait des compétences variées. Pour les plus jeunes ou les enfants ayant des difficultés dans ce domaine, la manipulation développe la motricité fine. Les déplacements développent la motricité, la latéralisation et parfois la numération. Les échanges développent le langage et la nécessité de formaliser ses idées pour bien se faire comprendre, et les collaborations mettent en évidence les avantages et les difficultés à travailler ensemble, ainsi que la nécessité de mettre en œuvre une dynamique de groupe.

En plus de ces compétences transversales, et comme on le verra dans les activités présentées dans la suite, on peut travailler sur des compétences disciplinaires, notamment mathématiques, au travers de ces activités, et ce pour deux raisons :

- d'une part la mise en œuvre de l'activité peut nécessiter des connaissances mathématiques (calcul du « coût » d'une solution pour essayer d'optimiser ce coût, par exemple dans l'activité des marmottes de la section **Erreur ! Nous n'avons pas trouvé la source du renvoi.**, calcul d'angle dans des extensions possibles du jeu du robot section **Erreur ! Nous n'avons pas trouvé la source du renvoi.**)
- d'autre part certaines activités informatiques peuvent utiliser un jeu de données (comme dans le réseau de tri section **Erreur ! Nous n'avons pas trouvé la source du renvoi.**) qui, une fois l'activité maîtrisée avec des jeux de données simples, peut être choisi dans le domaine souhaité (calculs, vocabulaire, histoire…). Comme l'activité est rapide à exécuter, elle peut être réutilisée ponctuellement pour permettre de travailler de manière originale et ludique le calcul mental, l'ordre lexicographique, les dates historiques ou autres.

II - QUELQUES ACTIVITES D'INFORMATIQUE SANS ORDINATEUR

Lors de l'atelier, un panel d'activités sans ordinateur a été présenté. Ces activités sont pour la plupart disponibles en ligne avec une description de l'activité ainsi que des documents d'appui voire une vidéo explicative sur le site de l'auteure[2]. Nous nous sommes ensuite concentrés principalement sur trois

[1] Projets d'Investissement d'Avenir

[2] Accessible à l'adresse : https://members.loria.fr/MDuflot/files/med/

activités présentées ci-dessous. Tout d'abord, le jeu du robot, qui permet une première approche de la programmation au travers de déplacements relatifs et absolus. Ensuite le réseau de tri, qui permet d'apprendre à exécuter un algorithme tout en travaillant la coopération et d'aborder la notion de parallélisme. Enfin les marmottes au sommeil léger, une activité où l'on travaille sur l'optimisation, la recherche de la « meilleure » solution, tout en calculant et en abordant la notion de compression des données.

1 Le jeu des robots

Le jeu du robot est une activité parfaite pour découvrir la programmation. En effet elle permet, sur un langage limité et assez facile à appréhender, de commencer par s'approprier le langage, puis d'exécuter des programmes, d'en écrire soi-même voire de changer de langage de programmation. Il est tout à fait possible de ne réaliser qu'une partie de l'activité en s'adaptant aux objectifs que l'on poursuit, au temps disponible pour l'activité et à l'âge du public. On peut donc avancer plus ou moins vite dans l'activité en fonction des capacités du public, et en adaptant l'activité on peut même en aborder certains aspect à dès le cycle 1[3].

Ce jeu est de plus au cœur du programme d'initiation à la programmation aux cycles 2 et 3 car il aborde la programmation par le biais de déplacements et se prête facilement à la transposition à un véritable robot voire un logiciel de programmation comme ScratchJR ou Scratch.

1.1 Public

Cette activité a été testée avec des élèves de grande section de maternelle, de début de primaire (CP et CE1), ainsi qu'un groupe d'élèves d'ULIS TFC (Troubles du Fonctionnement Cognitif). Elle peut s'adapter à des élèves de cycle 1 à 3 en adaptant ses objectifs. Pour les plus grand•e•s, elle constitue une première étape, mais rencontre ses limites car ne se prête pas trop à l'écriture de boucles (pour dessiner un carré par exemple, voir extension Section **Erreur ! Nous n'avons pas trouvé la source du renvoi.**).

1.2 Matériel

Le matériel minimum pour réaliser l'activité est très limité. Il suffit de réaliser un paysage[4], par exemple à la craie dans la cour, ou avec des objets réels que l'on pose au sol. Pour préciser ce que représente un pas, on peut dessiner des points régulièrement espacés en formant un maillage. On peut également dessiner un quadrillage, ou même utiliser le carrelage au sol si la taille des dalles s'y prête. Pour plus de réutilisabilité, on peut tracer une fois pour toutes le paysage et le maillage sur un support (un grand drap par exemple) ce qui rend le paysage facile à installer. Il est également utile de préparer quelques programmes exemples pour que les enfants puissent tester des programmes existants avant de se lancer dans la réalisation de leurs propres programmes.

[3] Un projet pour décliner les activités du réseau de tri et du jeu du robot sur les trois classes du cycle 1 est en cours de mise en place dans une école maternelle de Nancy.

[4] Un document explicatif du déroulé de l'activité contenant également une version du plan est disponible au format pdf en suivant le lien : https://members.loria.fr/MDuflot/files/med/doc/robot/robot.pdf

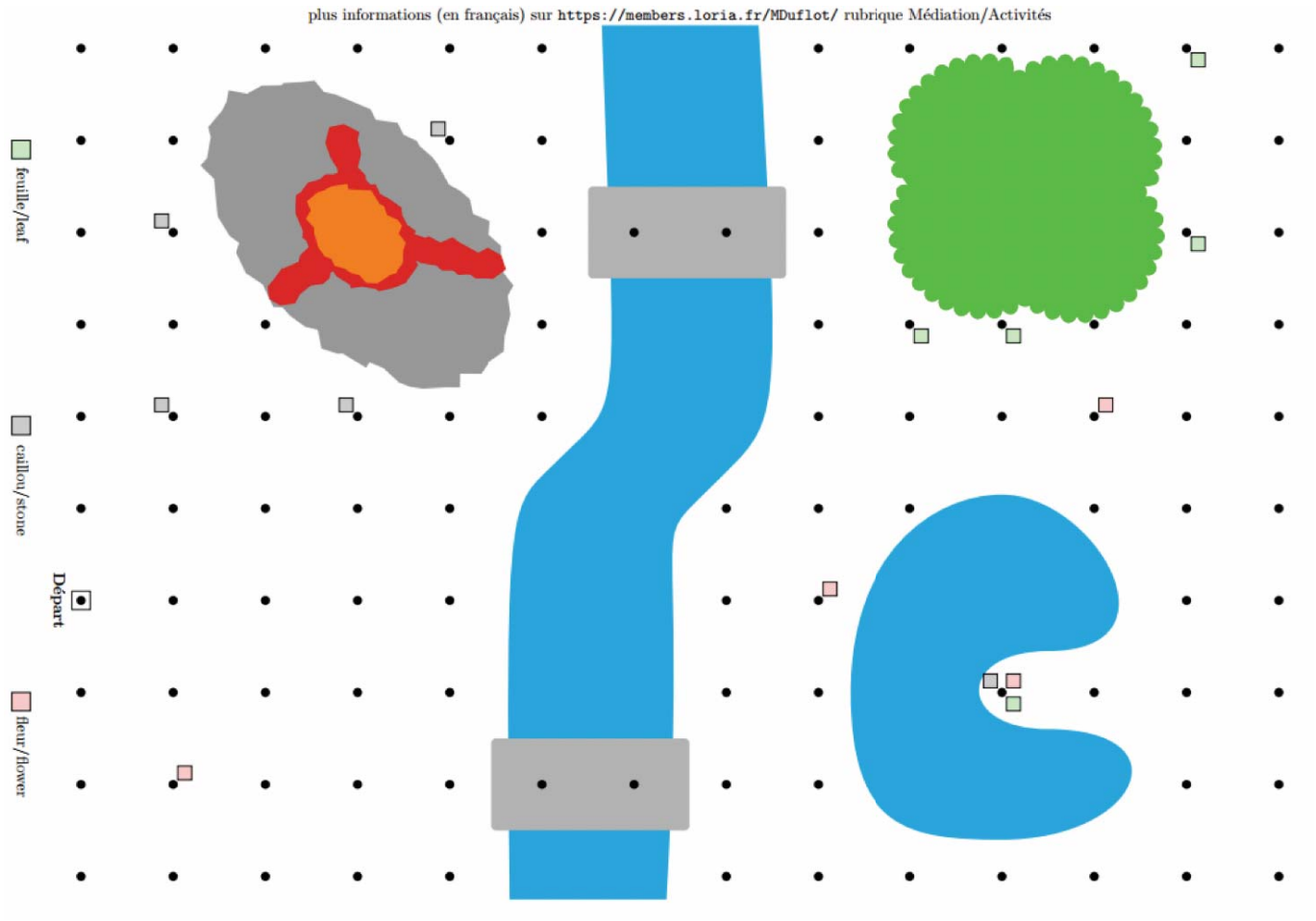


Figure 1. Plan utilisé pour le jeu du robot.

1.3 Langage

Comme décrit dans les ressources d'accompagnement du programme de cycles 2 et 3, la définition du langage de programmation est importante. Il doit être simple, non ambigu, et proche du langage utilisé par les enfants pour faciliter son appropriation.

Dans un premier temps, on ne va réaliser que des translations, en se déplaçant d'un pas en avant, en arrière, à gauche ou à droite sans jamais tourner sur soi-même. Le langage se compose donc uniquement de 4 flèches, avec la convention qu'une flèche pointant vers le haut correspond à avancer, et une vers le bas à reculer. Il est évidemment possible d'utiliser un langage textuel, mais les flèches permettent de réaliser l'activité avec les plus jeunes ou avec des enfants avec des difficultés de lecture.

Pour les plus jeunes, il est utile de commencer par s'approprier le langage, par exemple en les faisant tous et toutes faire les mêmes déplacements en même temps (à la manière d'un cours de fitness) en suivant une personne qui leur montre. On peut également, pour les enfants non latéralisés, surligner les flèches gauche et droite de deux couleurs différentes, et mettre des bracelets des couleurs correspondantes aux deux poignets de l'enfant. Elles et ils ne font donc plus un pas à gauche et un pas à droite mais un pas côté bleu et un pas côté rouge.

1.4 Tester des programmes

On explique d'abord que l'on ne peut se déplacer sur le paysage que dans la partie quadrillée. Tous les éléments dessinés/posés au sol ne doivent pas être touchés/piétinés. On ne peut ainsi traverser une rivière qu'en passant sur le pont, sur les points dessinés à cet effet. On propose différents programmes aux enfants, en spécifiant bien de quel point et dans quelle orientation on doit partir. On fait jouer à l'un ou l'une des enfants le rôle de la mémoire de l'ordinateur (qui lit les instructions) et l'autre le rôle du processeur/robot (qui exécute les instructions). Après quelques programmes corrects qu'elles et ils

doivent exécuter avant de dire quel déplacement il a permis, il est intéressant de leur donner un programme contentant un « bug ». Cela leur permet en effet de se rendre compte que, si un programme contient une erreur, le robot/ordinateur va l'exécuter sans se poser de questions (et par exemple tomber dans la rivière). L'ordinateur est rapide, précis, mais pas intelligent. Il ne fait que ce qu'on lui dit.

C'est également l'occasion de dire ou rappeler qu'en informatique beaucoup de monde fait des erreurs dans ses programmes. Ils sont souvent tellement grands et compliqués que c'est normal de manquer quelque chose. Seulement c'est le travail de l'informaticien•ne de vérifier ses programmes, de les tester et de trouver/corriger ses erreurs. Un tel rappel n'est pas inutile pour des enfants qui associent souvent erreur et échec. Chaque erreur doit être vue comme un pas de plus vers la solution, et le test nous permet de voir à quel moment dans le programme on a fait une erreur.

Pour les élèves les plus jeunes, des enseignant•e•s ont identifié de possibles difficultés. Avancer tout droit d'un point à un autre sans partir sur la diagonale peut être difficile. On peut donc tracer même en ligne plus fine le quadrillage entre les points. Le fait de lire un programme sur plusieurs lignes peut également être une difficulté. Il peut donc être utile de commencer par des programmes courts, ou de trouver des feuilles plus grandes pour les faire tenir sur une seule ligne.

1.5 Écrire ses propres programmes

Une fois le langage de déplacements bien maîtrisé, les élèves peuvent commencer à réaliser leurs propres programmes. Il faut donc leur donner le point de départ, l'orientation, et donner un objectif. On peut choisir de fixer un point de départ et une orientation et de s'y tenir pour tous les programmes, ou bien spécifier le point de départ à chaque fois, voire donner des points de départs différents à des groupes d'élèves pour qu'elles et ils puissent écrire et tester leurs programmes sans trop se marcher sur les pieds. Pour les objectifs, on peut commencer par des choses simples comme « aller près du volcan » ou « traverser la rivière », fixer un point précis et leur demander de s'y rendre.

Si on veut varier les programmes, on peut fabriquer des images d'objets et les poser sur des points précis du drap[5]. On ajoute donc une instruction supplémentaire : « ramasser », que l'on peut abréger avec la lettre R, qui permet de ramasser tous les objets qui sont sur notre point.

Les objectifs deviennent donc : « ramasser une fleur », « ramasser deux cailloux », « ramasser un objet de chaque » ou encore « ramasser deux feuilles et une fleur sans jamais marcher sur un caillou ». On peut bien évidemment calibrer la difficulté en fonction des facilités du groupe sur ce genre d'exercice.

C'est l'occasion de repréciser si besoin que l'ordinateur ne fait que ce qu'on lui demande. S'il passe sur un objet et qu'on ne lui demande pas de le ramasser, il le laisse sur place.

En donnant le même objectif à deux groupes différents, on met également en évidence que, pour une même tâche à réaliser, il existe plusieurs programmes corrects. La comparaison de ces programmes peut alors se faire sur le nombre et le type d'instructions à réaliser. Pour l'objectif de ramasser un objet de chaque, un robot qui roule très vite mais se baisse très lentement aura tout intérêt à aller sur un point sur lequel se trouvent les trois objets pour ne se baisser qu'une fois. Au contraire, un robot plus lent à se déplacer mais sans problème pour se baisser cherchera à prendre le chemin le plus court, quitte à se baisser plusieurs fois.

1.6 Changement de langage et équivalence

Les variantes du langage peuvent être introduites avant ou après l'écriture des programmes, au choix de la personne qui présente l'activité. Les modifications possibles pour ce langage sont de deux types :

- Tout d'abord pour se faciliter la vie, on peut avoir envie, au lieu de dessiner 4 flèches consécutives vers la droite →→→→, de noter plutôt 4→ ou 4 × →. Cela évite les erreurs en écrivant/lisant le programme si on manque une flèche par exemple. C'est alors une première idée de la boucle : on donne une instruction et on dit combien de fois la répéter. Le langage prend ainsi moins de place et gagne en clarté.

[5] Comme visible sur le plan du document https://members.loria.fr/MDuflot/files/med/doc/robot/robot.pdf

- On peut également supprimer les flèches latérales et en arrière, et les remplacer par des quarts de tour[6] (à droite ou à gauche). On peut donner aux enfants de nouveaux objectifs à réaliser avec ce langage, ou les inviter à réécrire leurs programmes précédents en utilisant cette nouvelle notation.

Pour les plus avancés, on peut essayer de trouver avec elles et eux un algorithme qui permettrait de traduire automatiquement un programme avec les translations en un programme avec des rotations. Il suffit, à chaque changement de direction dans les déplacements, de coder ce changement de direction à l'aide de quarts de tours, puis de transformer toutes les flèches en « avancer ».

Par exemple le programme ↑↑→← ↑ (avec des translations) se traduit en ↑↑ Droite ↑ Gauche Gauche ↑Droite ↑ (avec des rotations).

Il y a donc dans cette seule activité plusieurs parties de difficultés différentes, à aborder toutes ou à sélectionner, pour un public allant du cycle 1 au cycle 3.

1.7 Extensions possibles

L'activité du robot telle qu'envisagée ici n'est qu'une première étape. Elle a des limitations notamment sur les rotations (avec notre système de points on ne peut tourner que par quarts de tour) ou les boucles (on peut répéter plusieurs fois une flèche en écrivant 5↑ mais pas un ensemble d'instructions, pour dessiner un carré par exemple). On peut lever ces limitations en quittant le quadrillage et en traçant ses angles au sol, en mesurant ses pas etc. On peut même tracer au sol le parcours suivi (pour réaliser les figures géométriques mentionnées sur le site EDUSCOL). Mais c'est en général à ce moment-là qu'on commence à trouver pénible de faire les choses « à la main » et qu'il est bienvenu de passer sur un ordinateur qui calcule lui-même les angles, les distances quand on avance etc.

2 Le réseau de tri

Cette activité est tirée du site Computer Science Unplugged, créé entre autres par Tim Bell, un des acteurs majeurs dans la réalisation d'activités informatiques à destination des enfants, notamment sans ordinateur (Bell, 2012). La version présentée en atelier ne présente pas de différence notable avec la version originale. Cette activité propose, en permettant aux enfants de se déplacer et en les incitant à coopérer, d'aborder la notion de parallélisme (voir section 2.5) et d'exécution d'un algorithme.

2.1 Public

Le public visé par le réseau de tri est très large. En prenant le temps on peut amener des élèves de cycle 1 à réaliser l'activité (sans pour autant aborder l'aspect recul sur le parallélisme). Mais le fait de pouvoir varier les ensembles de cartes pour augmenter la difficulté et varier les domaines d'application (voir Section 2.4) permet à la fois d'intéresser et de faire réfléchir des élèves de primaire comme de secondaire, mais également de revenir ponctuellement sur cette activité pour travailler différentes compétences disciplinaires.

2.2 Matériel

Comme pour le jeu du robot, on peut dessiner son réseau de tri à la craie, ou le réaliser avec des objets (baguettes assez longues et cerceaux). Des modèles sont disponibles sur internet avec plus ou moins de valeurs à trier. Le réseau le plus classique pour ce genre d'activité ordonne 6 valeurs. C'est celui qui a été utilisé lors de l'atelier[7].

[6] Nous n'avons à ce jour pas trouvé de symbole satisfaisant pour les quarts de tour. Les lecteurs et lectrices ayant une idée à ce sujet sont invité•e•s à se manifester auprès de l'auteure de cet article.

[7] On peut le récupérer sur https://members.loria.fr/MDuflot/files/med/reseautri.html

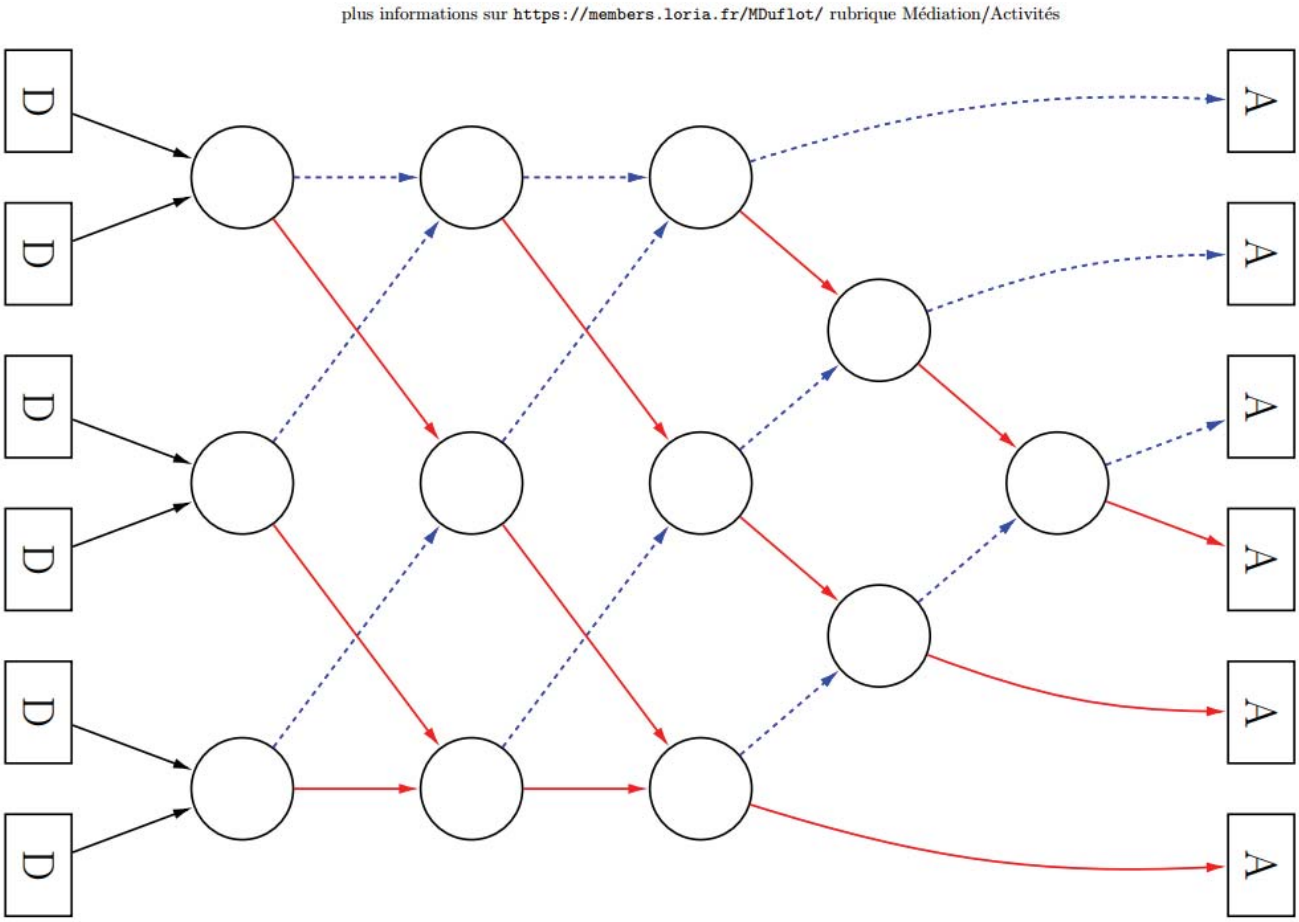


Figure 2. Plan du réseau de tri pour 6 valeurs

Il faut également réaliser quelques jeux de cartes à trier, à ajuster en fonction du niveau des élèves (voir dans les deux sections suivantes pour des idées de ce qu'on peut mettre sur ces cartes).

2.3 Réaliser son tri sur un réseau

Six élèves se positionnent sur les états de départ du réseau et on distribue au hasard à chacun et chacune une carte contenant une valeur. Pour les plus jeunes on peut réaliser des cartes contenant plus ou moins de points qu'elles et ils vont devoir compter. À partir du cycle 2, un jeu de cartes contenant les chiffres de 1 à 8 convient parfaitement.

Quand on lance l'activité, les élèves vont avancer sur le réseau en suivant les lignes dessinées et arriver dans un cercle, que l'on nomme centre de calcul. Comme deux lignes arrivent sur le même centre, deux élèves vont s'y rencontrer. Elles et ils vont alors comparer leurs cartes, la personne qui a la plus petite valeur va partir à gauche, et celle qui a la plus grande valeur part à droite[8], chacune des deux suivant une des deux lignes sortant du centre de calcul. Les élèves vont alors arriver dans un nouveau centre de calcul, y rencontrer un•e nouve•au•lle camarade, comparer et continuer ainsi jusqu'à atteindre un état d'arrivée. Quand tout le monde est arrivé, on regarde toutes les cartes et on vérifie si elles sont bien rangées dans l'ordre croissant. Les élèves vont réaliser que, sans coopération, on se trouve rapidement coincé avec trop de personnes dans un centre de calcul, et une personne laissée seule sur le réseau. C'est donc à celle ou celui qui va le plus vite d'attendre sa ou son camarade pour s'assurer avant de partir vers la prochaine étape que tou•te•s les deux ont compris où aller. Le but n'est pas d'aller le plus vite individuellement, mais d'aller le plus vite ensemble.

[8] Dans le cas où les deux valeurs sont égales, peu importe qui va à droite et qui va à gauche. C'est une réflexion intéressante qui mérite de mettre des cartes de même valeur.

On peut ensuite faire tourner différents groupes sur le réseau pour un même jeu de cartes et, pour pimenter le jeu, les chronométrer pour voir quel groupe trie ses cartes le plus rapidement.

2.4 Extensions, ou comment faire des mathématiques (ou plus) avec le réseau de tri

Une fois que l'activité du réseau de tri est maîtrisée, on peut varier les ensembles de cartes à loisir, avec pour seules limites les connaissances des élèves et son imagination. On peut donc imaginer ranger des images pour reconstituer une histoire dans l'ordre, des objets par taille, des nombres avec plus ou moins de chiffres, ou encore des calculs suivant leur résultat (en adaptant la difficulté du calcul à son public) ce qui nécessite d'effectuer le calcul rapidement. On peut également trier des mots selon l'ordre lexicographique, des objets en suivant leur poids ou encore des périodes historiques de la plus ancienne à la plus récente.

En clair, une fois qu'on a investi dans l'activité, on peut ressortir son réseau de tri pour un oui ou pour un non, sans avoir à tout réexpliquer et donc en perdant très peu de temps, pour travailler mettons le calcul mental ou la comparaison de grands nombres et ce de manière différente/ludique.

2.5 Parallélisme

Cette activité a été initialement créée pour illustrer une notion importante en informatique, incontournable dans les super ordinateurs permettant de réaliser en un temps raisonnable des calculs gigantesques : le parallélisme.

L'idée est toute simple : si on doit transférer le contenu d'un gros tas de sable dans la remorque d'un camion et qu'une seule personne est disponible, cela va prendre un certain temps. Si plusieurs personnes équipées de pelles viennent transférer le tas, on va nettement diminuer le temps total. Si le temps pour une personne était de 5 minutes et que 5 personnes mettent une minute, on peut se dire qu'un gain de 4 minutes ne change pas grand-chose. Par contre si la personne seule avait dû y passer 5 jours et qu'à 5 une seule journée est nécessaire, le sable va pouvoir partir le jour-même et être utilisé dans un chantier où il est sûrement attendu.

En informatique c'est exactement la même chose. Un processeur peut a priori réaliser un calcul à la fois. S'il a 200 calculs à faire il va les faire à la suite et cela lui prendra 200 étapes de calcul. Si maintenant on a un ordinateur qui dispose de plusieurs unités de calcul (on parle d'ordinateur « multi-cœurs ») alors il peut réaliser plusieurs calculs en même temps, « en parallèle » et cela améliore d'autant son efficacité. Sur le réseau de tri, on a fait 12 comparaisons (il suffit de compter les cercles) mais on ne les fait pas une à la fois. Lors de la première étape de calcul, on a fait 3 comparaisons en parallèle (elles sont alignées), puis encore 3 à la deuxième étape etc. Pour savoir le nombre d'étapes de calcul nécessaires, il suffit de compter le nombre de « lignes » de centres de calculs. On en a 5. Le parallélisme dans ce cas-là nous a permis de passer de 12 étapes à seulement 5.

Ce qui paraît étrange c'est qu'on avait 12 étapes, qu'on peut faire 3 choses à la fois, et qu'on n'obtient pas au final 12/3=4 étapes. Ceci est dû au fait que, sur la fin, on n'a plus beaucoup de comparaisons à faire mais elles dépendent les unes des autres. Pour savoir qui comparer sur la dernière ligne il faut voir le résultat des comparaisons précédentes. La difficulté pour faire du calcul en parallèle, c'est justement de trouver les calculs qui sont indépendants les uns des autres, pour pouvoir les réaliser en même temps. Et s'il y a trop de dépendances, rajouter des unités de calculs supplémentaires ne servira à rien car on ne pourra pas toutes les faire travailler en même temps.

3 Les marmottes au sommet léger

Cette activité a été créée pour le Module 2 (Manipulez l'information) du projet Class'Code. Au travers d'un univers montagnard, elle présente un algorithme destiné à la compression de données, en abordant la notion d'optimisation d'une solution. Elle parvient à montrer de manière ludique et facile d'accès un concept a priori complexe. L'idée est qu'une troupe de marmottes décide de se créer un nouveau terrier en vue de la prochaine hibernation. Seulement, ces marmottes ont le sommeil léger et elles vont devoir choisir la forme de leur terrier ainsi que la répartition dans les chambres de telle sorte qu'elles se dérangent le moins possible lors de leurs réveils hivernaux.

3.1 Public

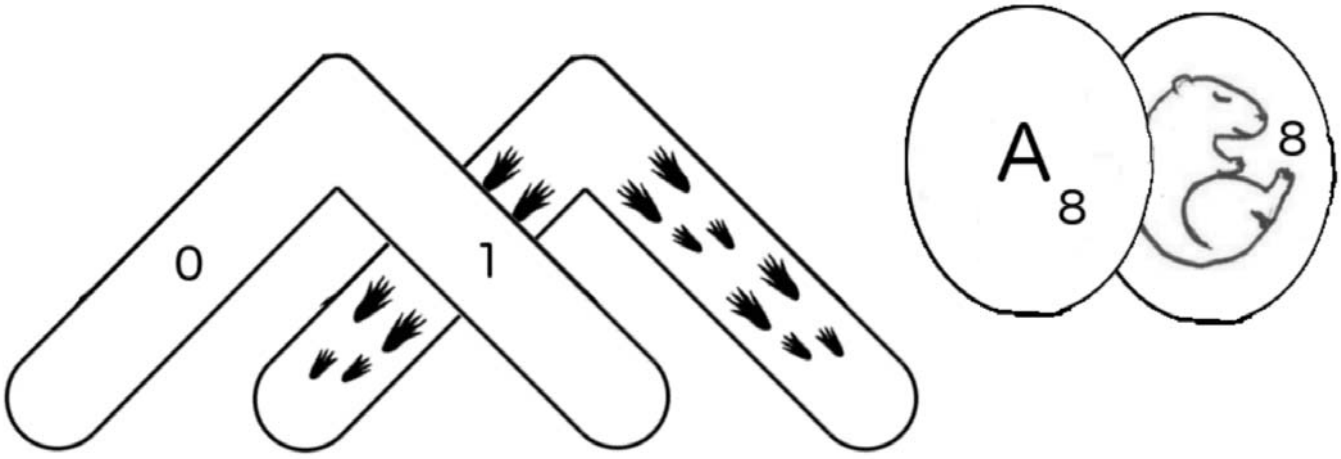
Contrairement aux deux activités précédentes, cette activité se prête plutôt à un public de fin de cycle 2 ou de cycle 3. Elle a été testée du CE2 à la 6e. Les raisons pour lesquelles elle semble moins adaptée pour les publics plus jeunes sont : le pouvoir d'abstraction nécessaire pour essayer d'optimiser la solution et le recours régulier à des additions et des multiplications pour évaluer chaque solution trouvée.

3.2 Matériel

Pour cette activité, il est nécessaire de reproduire ou d'imprimer le matériel disponible sur la page de l'auteure[9]. Il se compose d'un ensemble de 8 couloirs formant des angles droits, et de 9 marmottes. Pour la préparation il faut découper les éléments (en laissant un peu de place autour de chaque marmotte), puis :

- prendre chaque couloir côté verso et, en mettant le coude vers le haut, écrire un 0 sur la branche de gauche et un 1 sur la branche de droite,
- pour chaque marmotte, écrire dans la place laissée autour de la marmotte lors du découpage, un nombre seul ; ce nombre désignera le nombre de réveils de la marmotte pendant l'hiver ; il est souhaitable de mettre des nombres en majorité différents (sinon l'optimisation du terrier n'aura pas trop d'intérêt) ; pour pouvoir expliquer l'intérêt informatique de l'activité, nous vous conseillons de ne pas prendre ces nombres au hasard, mais plutôt de choisir une phrase ne contenant pas plus de 9 caractères, espace inclus (à moins de vouloir utiliser plus d'un kit), par exemple BARBARA A RASE BASILE LE BEAU BARBIER et compter le nombre d'occurrences de chaque caractère, ici B : 6, A : 8, R : 5, Espace : 6, S : 2, E : 5, I : 2, L : 2, et U : 1, puis pour chaque marmotte d'écrire au dos un des caractères de notre phrase et son nombre d'occurrences (par exemple A 8) et, dans la place laissée autour de la marmotte lors du découpage, le nombre seul (ici 8),
- si possible plastifier le tout et mettre du scratch adhésif grattant à l'avant des extrémités des couloirs, ainsi qu'au dos des coudes des couloirs et des marmottes.

On peut également ne mettre que les 0 et 1 au dos des couloirs, et laisser les marmottes vierges de toute notation pour pouvoir changer sa phrase d'une fois sur l'autre.



3.3 Creuser son terrier

Une fois le matériel distribué (un kit pour 2 élèves est l'idéal), expliquer les trois règles pour creuser le terrier.

1. À partir de l'entrée du terrier (qui est le coude d'un couloir, celui le plus haut) comme du bout d'un couloir, on ne peut faire partir que deux branches de couloir maximum.

[9] Vous pouvez télécharger le kit à imprimer https://members.loria.fr/MDuflot/files/med/doc/complet.pdf ou encore un document complet décrivant l'activité https://members.loria.fr/MDuflot/files/med/doc/fichemarmottes.pdf

2. Au bout d'un couloir, on met soit un coude de couloir (et donc deux nouvelles branches) soit une marmotte qui dort, mais par les deux.
3. Une fois toutes les marmottes placées, il faut compter les déplacements des marmottes. Le nombre près de la marmotte représente le nombre de fois qu'elle se lève dans l'hiver. Si une marmotte se lève 5 fois et est à distance 4 (= 4 morceaux de couloir) de l'entrée, elle va parcourir 5 × 4 = 20 morceaux de couloirs. On fait la somme pour toutes les marmottes et on essaie d'avoir un total le plus petit possible.

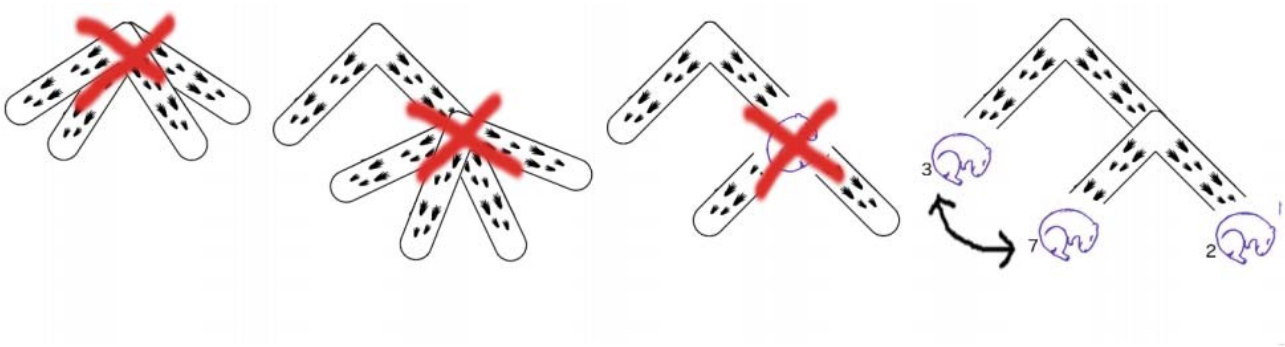


Figure 3 – Exemples ne satisfaisant pas une des règles de construction du terrier. De gauche à droite deux exemples violant la règle numéro 1, un exemple violant la règle numéro 2 et un terrier non optimal (donc ne respectant pas la règle numéro 3).

Les justifications de ces règles sont les suivantes :

1. Si on construit trop de chemins partant du même point, la structure risque de s'effondrer, et donc deux branches sera le maximum.
2. Les marmottes ont le sommeil léger. Si on en fait dormir une au milieu d'un chemin que d'autres vont emprunter en se levant, elles vont lui marcher dessus, la réveiller et lui ruiner son hibernation. Il faut donc mettre les marmottes au bout d'un couloir avec plus rien derrière.
3. Le sommeil de nos marmottes est tellement léger que même le simple bruit des pas de marmottes les dérange, ce qui risque d'altérer la qualité de leur sommeil. Elles doivent donc trouver le moyen de minimiser les déplacements (et donc les bruits de pas qui vont avec).

Construire un terrier respectant les deux premières règles est relativement simple, mais arriver à obtenir le terrier avec le moins de déplacements est plus complexe. Avec des jeunes élèves on peut commencer avec un plus petit ensemble de marmottes. Lors des expérimentations, les enfants arrivent en général à voir qu'il faut mettre les marmottes qui se réveillent le plus souvent plus près de l'entrée que celles qui se réveillent moins. Mais par contre décider de la forme du terrier (des couloirs très longs et d'autres très courts, ou plutôt des moyens) est une chose qui demande réflexion, et en pratique est réalisée par tâtonnements. C'est là qu'on peut lancer la partie suivante.

3.4 Un algorithme pour optimiser son terrier

Pour trouver un terrier optimal[10], il y a une méthode infaillible, qui ne demande pas trop de calculs :

- on choisit les deux (ou deux parmi les) marmottes qui se lèvent le moins souvent, et on les relie par un morceau de terrier. Sur le coude, on note le nombre de fois que ce morceau est emprunté (donc la somme des réveils des deux marmottes, par exemple 2 + 3 = 5),
- on recommence exactement la même chose, mais les deux marmottes reliées à l'étape précédente comptent maintenant pour une seule marmotte qui se réveillerait 5 fois,
- on continue, jusqu'à ce que toutes les marmottes soient reliées en un seul terrier,
- et comme on a noté des nombres sur les coudes de terriers au fur et à mesure on a juste à faire la somme de ces nombres pour compter le nombre de déplacements[11].

[10] qui dans la grande majorité des cas n'est pas unique

On peut retrouver cette méthode en vidéo sur Class'Code (Class 'Code , 2015) dans le module 2 (Manipulez l'information)[12].

3.5 Marmottes et compression

Il se trouve que l'algorithme donné ci-dessus n'a pas été choisi pour sa simplicité, dans un simple but pédagogique. Il date de 1952 et est dû à David Albert Huffman, professeur en informatique. Bien évidemment, s'il ne servait qu'à creuser des terriers, on n'en parlerait plus. Seulement, il a un impact crucial en informatique, dans la compression de données. Si on retourne notre terrier, on se rend compte que les couloirs contiennent des 0 et des 1, et que ce ne sont plus des marmottes mais des lettres qui se trouvent au bout des couloirs. D'ailleurs c'est le nombre d'occurrences de chaque lettre qui a déterminé le nombre de réveils de la marmotte de l'autre côté.

En informatique, toute donnée est stockée en binaire, avec des 0 et des 1. Le texte ne fait pas exception. Et il se trouve que, étant donné un texte, l'algorithme de Huffman permet d'associer à chaque lettre un code binaire (que l'on lit en suivant le chemin de l'entrée du terrier à la lettre), choisi en fonction des fréquences d'apparition des lettres, de sorte que le code binaire du texte complet soit le plus petit possible. On parle donc de compression. L'idée est que le codage standard des lettres (appelé code ASCII) associe à chaque caractère un code de 8 chiffres binaires (8 bits). Ce codage n'est pas optimal et le secret du codage de Huffman est d'associer de plus petits codes aux lettres les plus fréquentes, quitte à associer des codes plus longs aux lettres qui apparaissent peu. Un peu comme dans une cuisine on range à portée de main les couverts/assiettes dont on se sert tous les jours mais on peut stocker l'appareil à raclette en haut du placard ou dans le grenier. Comme on l'utilise peu, ce n'est pas si grave d'avoir à le chercher plus loin, surtout si ça permet d'accéder plus rapidement à d'autres choses plus souvent utilisées. À titre d'exemple, cet algorithme appliqué au texte intégral de Cyrano de Bergerac permet un taux de compression de 40%. On divise quasiment par deux la taille du document.

Ce codage de Huffman a aussi une autre propriété très importante : il est très facile de compresser un texte et de le décompresser. Pour la compression, pour chaque lettre à coder on lit les 0 et 1 sur le chemin de l'entrée à cette lettre, et on les note. Pour la décompression, on part de l'entrée du terrier et on lit les 0 et 1 du texte compressé en suivant le chemin correspondant dans le terrier. Si on arrive à une lettre on la note, on se replace à l'entrée du terrier, et on continue à lire nos 0 et nos 1. C'est d'ailleurs une application sympathique faire avec les élèves : on choisit un arbre de Huffman précis que l'on met à disposition de tou•te•s et on leur donne un texte compressé en binaire, à décompresser.

Et ce n'est pas encore tout. Car même si vous ne connaissez pas David Albert, vous utilisez son codage tous les jours sans le savoir. Il se trouve que les formats compressés habituels suivent un algorithme qui se compose de deux étapes. La première peut être complexe et est spécifique au type de donnée (image, son…) mais une fois ceci fait, on applique… l'algorithme de Huffman. En regardant des vidéos au format mpeg, des images jpeg, en écoutant des musiques en mp3 ou en ouvrant un fichier zip… on utilise l'algorithme de Huffman.

III - RETOURS ET CONCLUSIONS

Tout au long de l'atelier, les échanges ont été enthousiastes et riches. Les participantes et participants ont permis d'identifier des notions en lien avec les programmes ainsi que de clarifier et/ou d'enrichir certains aspects, présentés dans les parties précédentes.

Il est ressorti qu'au final ces activités pouvaient être réutilisées dans différents cadres. Elles peuvent tout d'abord être présentées comme des activités informatiques en tant que telles, en conformité avec le programme des cycles 2 et 3 (EDUSCOL, 2016) pour la partie initiation à la programmation. On peut

[11] Cette méthode permet de calculer les déplacements non pas par marmotte mais par pièce de terrier. Elle peut également s'appliquer à la section précédente quand les élèves galèrent trop avec leurs multiplications. Elle permet du coup de ne faire plus que des additions.

[12] C'est plus précisément dans la Partie 2 (optimisez le codage de l'information) Vidéo 4 (compression de l'information)

également proposer ces activités pour travailler la « pensée informatique », cette capacité à décomposer un problème en étapes simples et précises, à trouver le lien entre plusieurs problèmes pour éventuellement réutiliser (une part de) la solution déjà disponible. Comme cette pensée informatique est utile dans d'autres champs de l'enseignement comme la résolution de problèmes mathématiques, les activités présentées peuvent fournir une autre façon de travailler cette compétence transversale. Enfin les activités peuvent se justifier de par le fait qu'elles font également travailler des notions non informatiques. Le calcul du niveau de bruit associé à un terrier dans l'activité des marmottes fait travailler le calcul (additions, multiplications, comparaisons). Dans le réseau de tri, on travaille bien entendu les comparaisons, mais elles peuvent être lexicographiques, numériques ou se baser sur une première étape d'analyse (trier les résultats d'un calcul, des périodes historiques,…) qui peuvent ainsi toucher à différents domaines du programme.

Lors des discussions, nous avons également abordé le ciblage du public pour les différentes activités. Si certaines nécessitent de bonnes capacités d'abstraction et de calcul (par exemple les marmottes) et se prêtent plutôt à une mise en œuvre en fin de cycle 2 voire cycle 3, pour d'autres il est possible, en travaillant la progressivité et en revoyant les objectifs à atteindre, de les aborder en début de primaire voire même au cycle 1 (réseau de tri, robot). Un projet de mise en œuvre de ces deux dernières activités, à décliner tout au long du cycle 1, est d'ailleurs en cours avec une école maternelle de Nancy.

Ces activités sans ordinateur permettent donc dans un cadre ludique, souvent utile pour déjouer les appréhensions/blocages des élèves, de travailler à la fois des compétences transversales et disciplinaires. Ce sont des activités demandant peu de matériel, rapides à s'approprier, qui permettent une première approche de la pensée informatique. Elles ne remplacent évidemment pas les activités sur ordinateur, nécessaires pour la validation de ses programmes/algorithmes/idées par une machine. Cependant, elles permettent de séparer deux difficultés : l'introduction des notions informatiques d'une part, et le domptage du matériel (ordinateur, robot, logiciel,…) d'autre part.

IV - BIBLIOGRAPHIE, WEBOGRAPHIE

BELL T. & ROSAMOND F., Casey N. (2012) Computer Science Unplugged and Related Projects in Math and Computer Science, *The Multivariate Algorithmic Revolution and Beyond*. Lecture Notes in Computer Science, **vol 7370**. Site http://csunplugged.org/.

Class'Code (2015) Site http://www.classcode.fr/,

MORE M. & GALI S. (2017). Faire de l'informatique sans ordinateur à l'école. *43ème Colloque COPIRELEM*.

DI COSMO R. (2015) Enseigner et apprendre les sciences informatiques à l'école. *Site d'Interstices* https://interstices.info/jcms/c_47072/enseigner-et-apprendre-les-sciences-informatiques-a-lecole

EDUSCOL (2016) Ressources d'accompagnement du programme de mathématiques (cycle 3) - Initiation à la programmation aux cycles 2 et 3, *disponible sur http://eduscol.education.fr/cid101461/ressources-maths-cycle-3.html*, 9 pages.

PAPERT S. (1980) *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

WING, J. (2006) Computational thinking. *Communications of the ACM* **49(3)**, 33-35. Disponible à l'adresse http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf