LA COMPLEXITÉ ALGORITHMIQUE : ON DÉBRANCHE ET ON DÉNOMBRE

Basile Sauvage
Université de Strasbourg, IREM de Strasbourg

Résumé. Cette contribution a un triple objectif : découvrir des activités d'informatique débranchée ; analyser ces activités au regard du concept de complexité algorithmique ; débattre des liens avec le programme de mathématiques au lycée. Les participants à l'atelier ont expérimenté les activités, desquelles émerge la notion de complexité algorithmique. Cette dernière s'apparente à plusieurs notions mathématiques : le dénombrement, les suites, les fonctions, les limites, le logarithme et l'exponentielle. Nous discutons de l'articulation des notions informatiques et mathématiques, ainsi que de l'opportunité pédagogique d'un éclairage croisé.

La présente contribution relate un atelier dont le point focal est le concept informatique de complexité algorithmique. Nous nous intéressons à l'exploration de ce concept avec les élèves, à partir du cycle 4, et plus particulièrement au lycée. À cette fin, nous nous appuyons sur trois activités d'informatique débranchée. En un mot, il s'agit d'activités qui, pour étudier la science informatique, ne s'appuient pas sur des ordinateurs ou des machines numériques. Les trois activités proposées, exploitables avec les élèves, sont afférentes aux algorithmes et permettent de faire émerger cette notion de complexité algorithmique.

Dans la majeure partie de l'atelier, les participants ont eux-mêmes expérimenté les activités, en petits groupes de quatre personnes, répartis en îlots. Nous avons commenté le déroulement des activités et complété la réflexion, tant par des éléments théoriques d'informatique que par des suggestions pédagogiques. Enfin, nous avons débattu de l'articulation avec plusieurs notions mathématiques du programme de lycée (le dénombrement, les suites, les fonctions, les limites, le logarithme et l'exponentielle), et de l'opportunité pédagogique d'un éclairage croisé entre mathématiques et informatique.

La suite de ce document relate et commente l'atelier dans l'ordre de son déroulement.

Activité : la recherche du minimum

La première activité, illustrée figure 1, consiste à rechercher la carte de valeur minimale, parmi un ensemble de cartes. Les cartes sont initialement face cachée. Il est autorisé de retourner deux cartes à la fois pour les comparer. Les participants doivent proposer un algorithme pour trouver la carte de valeur minimale, et compter le nombre de comparaisons effectuées.

L'algorithme le plus classique commence par comparer les deux premières cartes. La plus petite des deux est alors comparée à la troisième, et ainsi de suite jusqu'à la dernière. On mémorise la position de la plus petite carte trouvée jusque là (successivement 7, puis 6, puis 3 sur la figure 1).

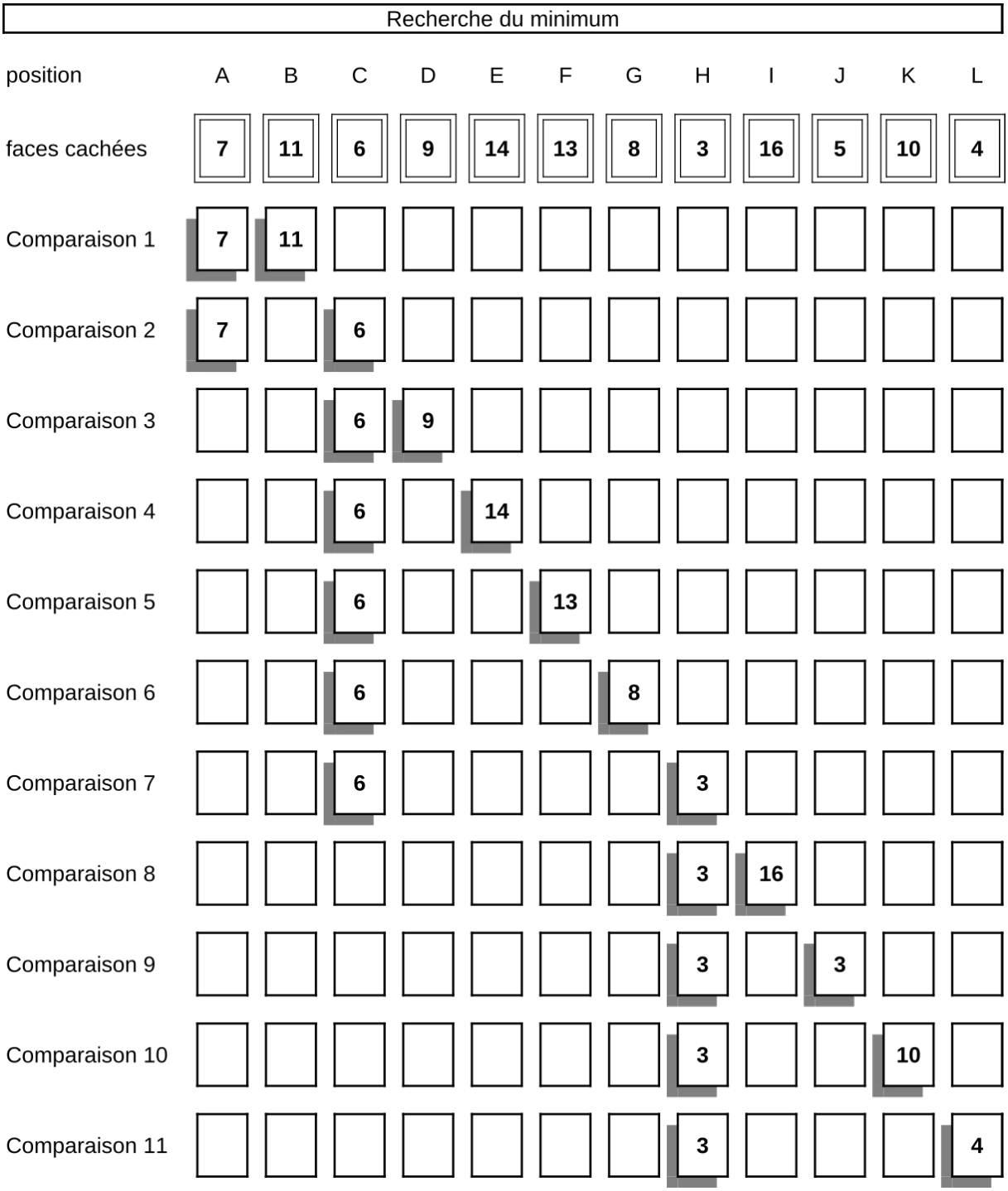

Figure 1. *Algorithme de recherche du minimum dans ensemble de valeurs, réalisé avec des cartes, faces cachées. Chaque ligne représente une comparaison, réalisée entre les deux cartes découvertes.*

On dénombre les comparaisons : 11 comparaisons pour 12 cartes, que l'on généralise à (n-1) comparaisons pour n cartes. Nous définissons informellement la notion de complexité T(n) comme le nombre d'opérations en fonction de la « taille du problème », en l'occurrence la taille des données (n). Pour cela il faut choisir une opération élémentaire représentative de l'algorithme, dans le cas présent nous choisissons la comparaison. On a ici une complexité dite linéaire T(n)=n-1.

Commentaires sur la conception et le déroulement de l'activité :

- Nous recommandons pas moins de 8 cartes pour éviter une mémorisation globale, pas plus de 16 pour limiter le temps de manipulation.
- Nous recommandons de numéroter les colonnes (ici de A à L) pour pouvoir les nommer lorsque les cartes sont face cachée. Si l'activité est réalisée au tableau, la numérotation peut être fixe (indépendante de l'ordre des cartes). À défaut, il est possible de numéroter le dos des cartes.
- Les valeurs des cartes ne doivent pas être les premiers entiers consécutifs {1, 2, 3, …} pour éviter que l'algorithme revienne à « chercher la carte de valeur 1 ».
- Il convient d'insister sur le fait que le processus doit être sans mémoire de l'ensemble des valeurs. Ainsi, on ne saura se satisfaire d'un algorithme comme « je parcours toutes les cartes et je me souviens de la plus petite ».

Activité : le tri par sélection

La seconde activité, illustrée figure 2, consiste à trier un ensemble de cartes, qui sont initialement face cachée. Il est autorisé de comparer deux cartes, et de déplacer une carte. Les participants doivent concevoir un algorithme de tri, et compter le nombre de comparaisons.

Le fait de proposer cette activité après celle de recherche du minimum met sur la piste du tri par sélection. La première étape consiste à chercher le minimum pour le placer au début. Dans l'exemple de la figure 2, la valeur 3 en position H est déplacée en position A ; les colonnes A à G sont décalées d'une place vers la droite (la valeur 7 en position B, etc.). Les étapes suivantes procèdent de la même manière sur les « cartes restantes », jusqu'à épuisement. Ainsi, pour n cartes, la première étape nécessite (n-1) comparaisons, la seconde (n-2), etc. Nous dénombrons ainsi le nombre total de comparaisons : $T(n)=n(n-1)/2$ comparaison pour n cartes. Cette complexité est dite quadratique car le terme dominant est n^2.

Commentaires sur la conception et le déroulement de l'activité :

- Nous recommandons pas moins de 8 cartes pour éviter une mémorisation globale, pas plus de 16 pour limiter le temps de manipulation.
- Nous recommandons de numéroter les colonnes (ici de A à L) pour pouvoir les nommer lorsque les cartes sont face cachée. Si l'activité est réalisée au tableau, la numérotation peut être fixe (indépendante de l'ordre des cartes). À défaut, il est possible de numéroter le dos des cartes, mais la numérotation des colonnes est alors modifiée quand les cartes sont déplacées (ce qui n'arrivait pas dans la recherche de minimum).
- Les valeurs des cartes ne doivent pas être les premiers entiers consécutifs {1, 2, 3, …}, sinon risque d'être proposé l'algorithme « placer la carte de valeur i en i-ième position ».
- Il convient d'insister sur le fait que le processus doit être sans mémoire globale, contrairement à ce que ferait un humain. Ainsi, on ne saura se satisfaire d'une instruction comme « je me souviens que la prochaine carte a pour valeur 3, en position E ».

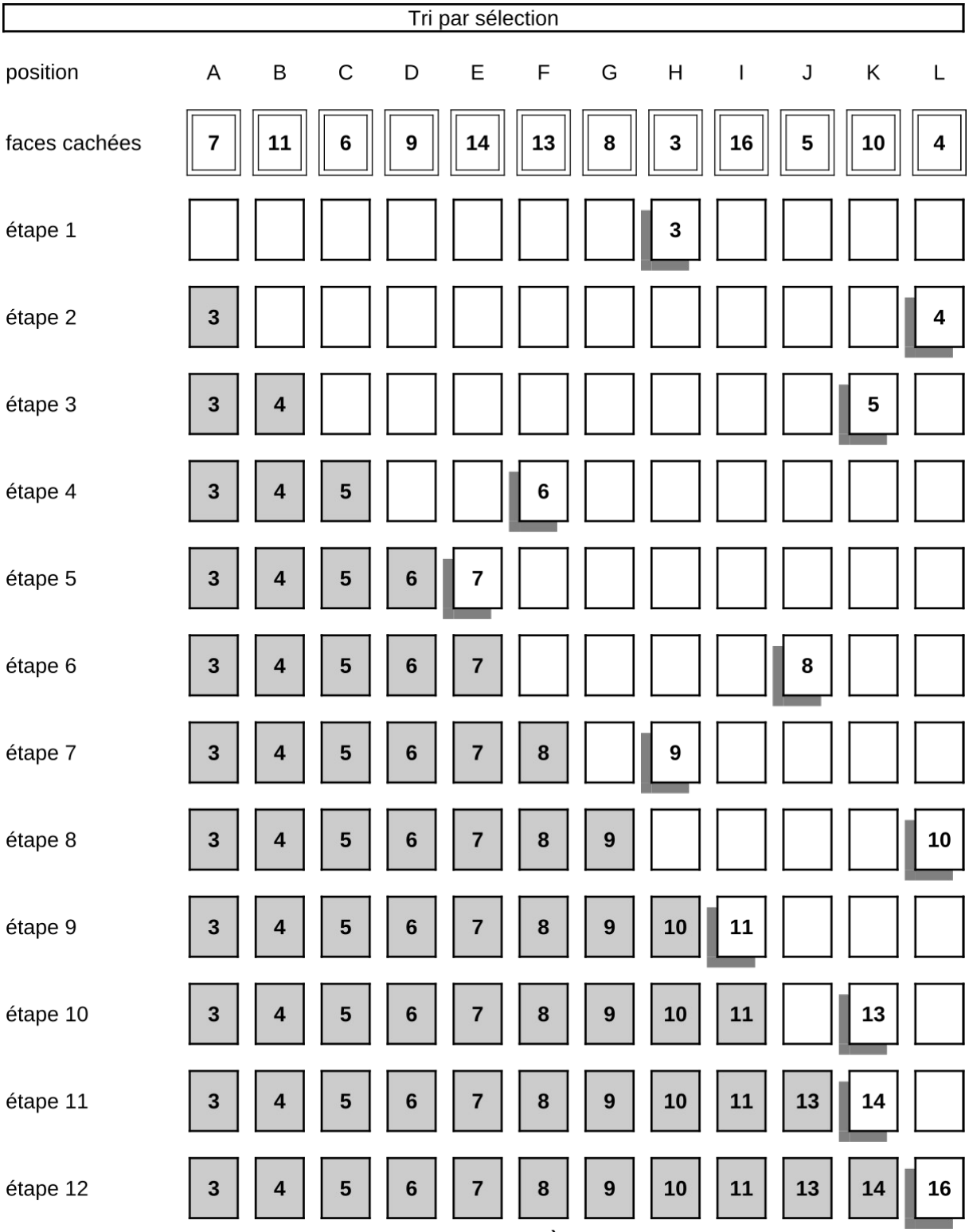

Figure 2. *Algorithme du tri par sélection du minimum. À chaque étape, parmi les cartes cachées restantes (blanches), le minimum (carte blanche, face visible) est sélectionné et placé à la fin des cartes déjà triés (grises, face visible).*

Réflexion sur la complexité asymptotique

La complexité d'un algorithme mesure la quantité de ressources qu'il utilise : généralement le temps de calcul, comme ici, ou bien l'occupation mémoire. Elle permet de comparer l'efficience des algorithmes entre eux. Elle est fonction de la « taille du problème » à résoudre (notée n), à savoir le nombre de valeurs (cartes) dans les exemples précédents. La complexité *asymptotique* quantifie mathématiquement la croissance de la complexité quand n tend vers l'infini. Dans la pratique informatique, c'est un outil pour prévoir le comportement d'un algorithme lorsque n « grandit » : si l'on double la taille du problème, le temps de calcul sera multiplié par 2 pour un algorithme linéaire, par 4 pour un algorithme quadratique.

Nous n'aborderons pas ici les outils mathématiques sous-jacents (croissances comparées et notations de Landau). Nous nous contenterons de faire remarquer qu'ils permettent de se débarrasser des termes négligeables en l'infini et de la constante multiplicative dans l'expression de la complexité. La recherche du minimum est un premier exemple, dont la complexité est notée $T(n) = \Theta(n)$, faisant disparaître la constante additive. Le tri par sélection est un second exemple, dont la complexité est notée $T(n) = \Theta(n^2)$, faisant disparaître le terme n négligeable devant n^2, et la constante multiplicative 1/2. Cela facilite la comparaison des complexités entre elles.

Le sujet qui nous occupe est de se faire une représentation du comportement asymptotique. Pour cela, nous imaginons le scénario où l'on cherche un nom mal orthographié dans un annuaire de n noms. Le premier algorithme renvoie le nom le plus proche (apparenté à une recherche du minimum en complexité linéaire). Le second algorithme trie l'annuaire par proximité décroissante (tri par sélection en complexité quadratique) et renvoie les premiers. En supposant une machine qui réalise 10^9 comparaisons par seconde (ce qui est un ordre de grandeur réaliste), nous construisons le tableau suivant :

	Complexité linéaire T(n)=n		Complexité quadratique $T(n)=n^2$	
	T(n)	Temps	T(n)	Temps
$N=10^3$ annuaire individuel	10^3	$10^{(-6)}$ sec.	10^6	$10^{(-3)}$ sec.
$n=10^6$ annuaire national	10^6	$10^{(-3)}$ sec.	10^{12}	17 minutes
10^9 annuaire mondial	10^9	1 sec.	10^{18}	32 ans

Ce tableau permet de commenter l'intérêt prédictif de la complexité asymptotique. Sans avoir besoin de tests réels, nous observons qu'un algorithme linéaire passera bien mieux à l'échelle qu'un algorithme quadratique. Même pour un annuaire de taille modérée, ce dernier serait très loin d'un temps interactif, auquel nous sommes pourtant habitués pour ce genre de recherche.

Activité : qui est-ce ?

Nous réalisons l'activité, illustrée figure 3, imitant le jeu du qui est-ce ?, qui consiste à deviner un personnage parmi n candidats, en posant des questions à réponse oui/non. Les quatre arbres de décision de la figure 3 sont distribués sans personnages, qui sont à part. Dans un premier temps, les participants placent les personnages dans les feuilles des arbres. Dans un second temps ils suivent l'enchaînement de questions imposé par un des arbres afin de deviner un personnage mystère. Dans un troisième temps ils déterminent le nombre de questions posées pour parvenir au but.

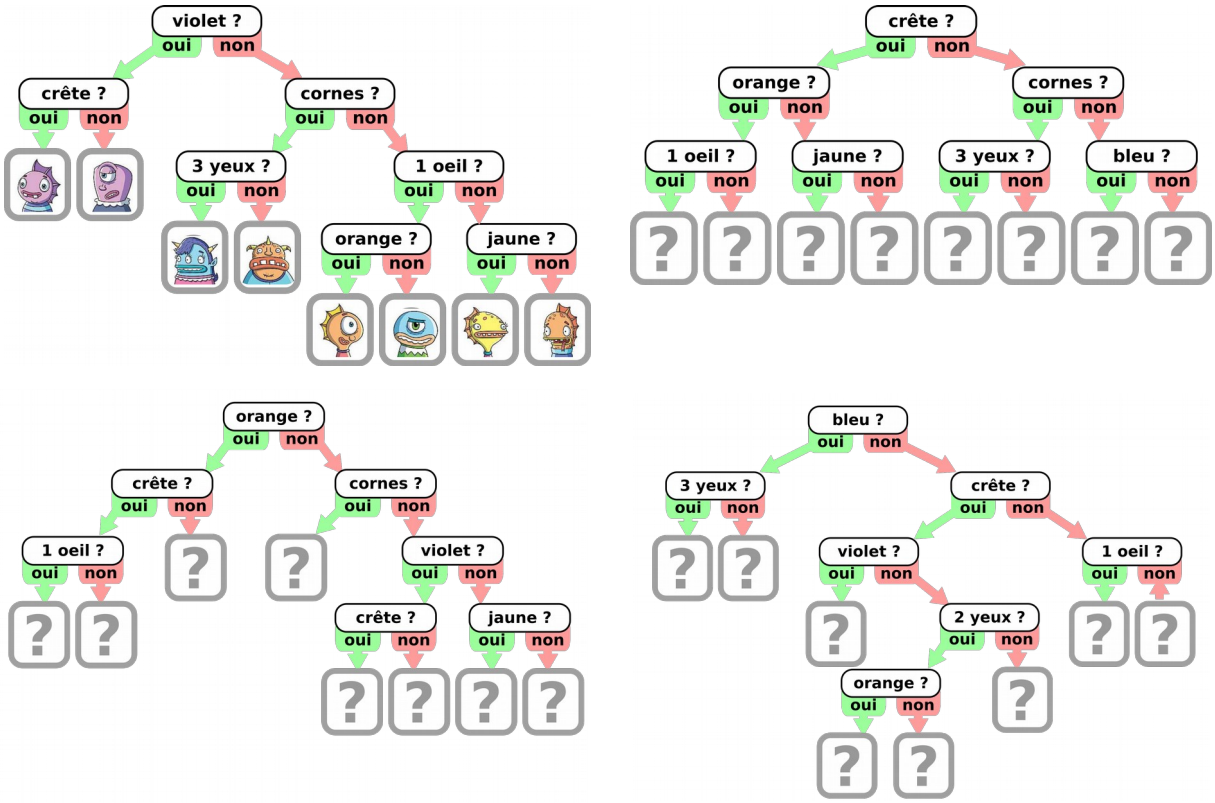

Figure 3. *Arbres de décision pour le jeu du « qui est-ce ? ». Les personnages à deviner sont dans les feuilles (en bas). Une suite de nœuds en partant de la racine (en haut) indique la suite de questions à poser. Les supports originaux pour cette activité sont disponibles auprès de l'auteur, ou auprès du groupe informatique à l'IREM de Strasbourg.*

On observe que les arbres déséquilibrés permettent de trouver parfois en peu de questions, parfois en beaucoup de questions (2 à 5 sur l'arbre en bas à droite). Un arbre équilibré permet de trouver en un nombre fixe de questions (3 sur l'arbre en haut à gauche). On peut vérifier qu'un arbre équilibré est plus rapide en moyenne. Ces observations sont mises en parallèle avec les notions de complexité dites « au mieux », « au pire », et « en moyenne », lorsqu'un algorithme a une complexité qui varie en fonction des données (pour une taille n fixée). Certains participants éprouvent une difficulté à faire le lien entre la complexité et le nombre de questions posées. En effet, il y a là une subtilité, si ce n'est une ambiguïté : dans le jeu réel, lorsqu'une question est posée, il faut parcourir tous les personnages restants ; ainsi une question peut engendrer n opérations.

L'arbre équilibré permet d'illustrer le logarithme en base 2 et l'exponentielle. Il n'est pas nécessaire ici de savoir définir formellement ces fonctions sur les réels, on peut se contenter

de les définir en nombre entiers (d'où l'intérêt que l'arbre soit équilibré). Pour n feuilles (i.e. personnages), le nombre de questions à poser est égal à la profondeur $p=\log_2(n)$ de l'arbre. Réciproquement, le nombre de feuilles $n=2^p$ croît exponentiellement en fonction de la profondeur. Si l'on rajoute un étage dans l'arbre, le nombre de personnage est doublé : $2^{(p+1)}=2*2^p$ qui rappelle la règle de calcul pour $2^{(a+b)}$, tandis que $\log_2(2n)=\log_2(n)+1$ qui rappelle la règle de calcul pour log(ab).

Lien avec les notions mathématiques au programme du lycée

Nous ouvrons un débat sur le lien entre la complexité algorithmique d'une part, et différentes notions de mathématiques d'autre part. La discussion est amorcée par deux questions : Comment mobiliser les mathématiques en informatique ? Comment mobiliser l'informatique en mathématiques ? La discussion, quoique courte, a fait émerger deux pistes de travail. La première piste est d'exercer les élèves au dénombrement et au calcul sur différents algorithmes. La seconde piste est un travail sur les relations fonctionnelles (log et exponentielle), et une appropriation à travers les règles de calcul. Le lien avec les suites et les limites, bien qu'évident, n'a pas été discuté.

Conclusion

Nous avons exploré trois activités d'informatique débranchée, qui permettent d'explorer avec les élèves différents aspects de la complexité algorithmique. Concernant la réception faite à cet atelier, nous noterons trois éléments saillants : le tableau des complexités converties en temps de calcul a marqué les esprits; l'exploration du logarithme à l'aide de l'arbre a intéressé les participants ; dans le « qui est-ce ? », le lien entre la complexité et le nombre de questions posées a posé question. Nous avons ébauché des pistes de réflexion sur la convergence entre des activités mathématiques et informatiques, qui mériteraient d'être approfondies. Notre démarche a été de partir de la pédagogie de l'informatique, pour s'ouvrir aux notions mathématiques. Il serait enrichissant d'aborder ces questions sous l'angle de la didactique des mathématiques. Il y a là une opportunité de créer des activités mixtes entre informatique et mathématiques, à même de favoriser la transposition et les apprentissages des deux côtés.