Démystifier les concepts informatiques par l'expérimentation

Dimitri Racordon, Didier Buchs

Dimitri Racordon & Didier Buchs

CUI, Université de Genève
dimitri.racordon@unige.ch, didier.buchs@unige.ch

Démystifier les concepts informatiques par l'expérimentation

Résumé
L'informatique s'étant aujourd'hui disséminée dans tous les domaines de la vie quotidienne, une compréhension générale de ses principes est devenue un outil indispensable. Malheureusement, force est constater que cet objectif est encore loin d'être atteint pour une majeure partie de la population, même dans les jeunes démographies. Ce document résume brièvement les expériences que nous avons conduites dans le cadre de campagnes de promotion des sciences informatiques. Nous en extrayons une approche plus générale et détaillons les avantages d'une telle démarche.

Mots clés : algorithmique, spécifications, apprentissage

1 Introduction

Malgré son omniprésence, l'informatique est une science peu connue du grand public. De plus, elle est employée comme synonyme de beaucoup d'autres termes qui ne la représentent pas en tant que science, mais plutôt son application à un domaine spécifique. Par exemple, les termes « numérique » ou encore « digital » sont souvent considérés comme équivalents, alors qu'ils correspondent plus à des applications ou technologies issues de la science elle-même. Outre la confusion qui plane au-dessus des termes, les concepts informatiques eux-mêmes sont également relativement méconnus. À tort, il est souvent admis qu'une maîtrise de l'utilisation d'une technologie implique une certaine connaissance de son fonctionnement. Par exemple, les principes sur lesquels repose le fonctionnement d'Internet sont généralement peu connus du grand public, alors que son utilisation est désormais monnaie

courante. Le mysticisme qui entoure l'informatique constitue un autre facteur intéressant. Si aujourd'hui les bases des mathématiques, de la physique ou encore des sciences humaines sont enseignées à tous, il en est tout autre pour l'informatique. Ce malheureux constat révèle que des termes tels qu'« algorithme », « complexité » ou encore « décidabilité » n'ont que peu si ce n'est aucune signification pour le grand public. Dans la communication de Baron et Bruillard (2001), un historique des difficultés de l'enseignement de l'informatique est relaté. Le problème apparaît comme étant lié à la fois à la démocratisation de l'usage des outils informatiques, de leur passage à une approche plus intuitive que raisonnée et la tardive émergence de la science elle-même. Une problématique similaire semble avoir existé dans l'enseignement d'autres sciences formelles comme les mathématiques.

Jacques Baudé, président d'honneur de l'EPI (Association Enseignement Public et Informatique – France) l'exprime de cette manière[1] :

> On nous dit encore, qu'il n'est pas besoin de savoirs savants, qu'il suffit de cliquer. Bel argument pour des marchands mais le rôle des enseignants n'est-il pas d'ouvrir ou d'entrouvrir, tant que faire se peut, les boîtes noires, de donner aux élèves les moyens de comprendre ce qu'ils font, de prendre du recul, et non pas d'en faire de simples consommateurs toujours dépassés ?

Dans ce document, nous proposons de définir un cadre permettant l'appréhension des fondements de l'informatique. Notre méthode s'appuie sur l'expérimentation comme un moyen de contrecarrer le mysticisme qui entoure cette science, en associant une signification concrète et familière à ses concepts, puis en offrant la possibilité de les manipuler de façon ludique et intuitive.

Nous ne sommes pas spécialisés en didactique et nos réflexions sont essentiellement basées sur l'expérience d'enseignement universitaire. Néanmoins, notre domaine d'activité en recherche s'intéressant aux aspects formels de l'informatique, ce qui par ailleurs explique notre goût pour l'abstraction et la modélisation formelle des systèmes, notre force est d'être à la croisée des approches théoriques et pratiques. Nous avons également une bonne connaissance des langages de programmation et des méthodes de développement du logiciel, nous donnant les outils nécessaires pour lier ce que nous appelons les concepts fondamentaux de l'informatique et leur application dans des domaines concrets. En outre, nous avons eu l'occasion de mettre notre démarche

1 <http://www.epi.asso.fr/revue/editic/jb-asti.htm>

d'apprentissage en pratique dans le cadre de diverses campagnes de promotion de l'université, et en particulier des études en informatique.

2 Résolution de problème : quoi et comment

Comme nous l'avons précisé en introduction, notre domaine d'activité principal s'intéresse aux aspects formels de l'informatique. Pour cette raison, nous choisissons de privilégier une vision abstraite et de masquer les aspects plus technologiques. Par là, nous entendons que nous préférons développer une compréhension des propriétés des algorithmes, ainsi que des données qu'ils manipulent, plutôt que de nous focaliser sur leurs applications. Lorsque nous aborderons nos expériences sur le terrain dans le chapitre 3, nous expliquerons par exemple comment nous avons pu tirer partie de robots, une application très technologique de l'informatique, pour discuter d'algorithmes répartis.

Ces aspects abstraits sont souvent nécessaires car ils précisent les informations que le contexte d'exécution des programmes demande. En tant que développeurs de logiciels, il nous paraît clair que ces notions sont primordiales. Nous pensons malheureusement qu'elles constituent le plus souvent un obstacle à la compréhension des fondements des problèmes à résoudre si le public ne parvient pas à percevoir l'implication concrète d'une opération ou propriété décrite de manière abstraite. L'acquisition de cette capacité n'est pas triviale. Le défi est cependant loin d'être insurmontable, et nous pensons qu'il peut être grandement simplifié en structurant de manière détaillée et systématique le processus de création d'un algorithme. Ce processus peut être décomposé en trois étapes, que nous détaillerons plus loin :

1. Classification des données
2. Description des propriétés de l'algorithme
3. Description des opérations de l'algorithme

L'une des difficultés de l'informatique est qu'elle requiert une extrême précision dans la description de ces trois étapes, entre autres parce que l'ordinateur doit être programmé avec soin pour pouvoir fournir des résultats

satisfaisants. Néanmoins cette nécessité peut être vue comme une opportunité, car hormis des travaux très manuels, beaucoup d'autres activités n'encouragent pas cette dimension. Précisons également qu'en utilisant une formulation abstraite, nous nous rendons indépendants d'une technologie ou d'un langage de programmation particulier. Nous utilisons donc une notation dite formelle (ou mathématique), dont le principal avantage se situe au niveau de son universalité. Alors qu'il existe d'innombrables langages de programmation et plateformes, tous accompagnés de leurs subtilités respectives, une notation formelle permet de fédérer les syntaxes et sémantiques de ces différents systèmes.

Pour exprimer ces problèmes, nous allons donc nous concentrer sur la description des données à traiter ainsi que sur les opérations à effectuer. Les opérations seront également abstraites, en utilisant un système permettant l'expression de leur état avant et après exécution de l'opération.

2.1 Classification des données

La représentation des données est une composante souvent négligée lorsque l'on parle d'algorithme. Elle est néanmoins essentielle, non seulement parce qu'elle permet une description précise des domaines d'entrée et de sortie d'un algorithme, mais aussi car elle peut contribuer à l'efficacité de l'algorithme lui-même. Beaucoup d'algorithmes simples vont travailler sur des données élémentaires. Historiquement, comme l'informatique était principalement utilisée pour résoudre des problèmes numériques, les algorithmes se contentaient de manipuler des nombres. Mais avec l'arrivée d'applications dans des domaines bien plus variés, une telle simplicité n'est plus suffisante. Il devient donc nécessaire de considérer d'autres types de données, plus complexes, qu'il convient de décrire minutieusement. Il existe une myriade de structures de données différentes. En fait, un pan important de la recherche en informatique consiste même à les étudier. Nous appelons cette étape la **classification**.

2.2 Descriptions des propriétés : le quoi

Décrire les propriétés d'un algorithme permet de réfléchir au problème à résoudre. Il convient ici d'identifier précisément l'objectif de l'algorithme,

ainsi que la nature des données qu'il traitera. Nous appelons cette étape le **quoi**. Même si les propriétés d'un algorithme pourront avoir des implications importantes sur la description de ses opérations, comme nous en discuterons plus loin, il est important de ne pas se soucier durant cette étape de la méthode de résolution qui sera développée à l'étape suivante. Certes, en fonction de la difficulté que représente cette dernière, il conviendra peut-être d'ajouter des propriétés (ou contraintes) sur les données à traiter pour simplifier le problème. Mais en distinguant bien ces deux étapes (description des propriétés *puis* des opérations), il nous sera donné l'opportunité de comprendre pourquoi un problème est trop difficile (voire impossible) à résoudre étant donné les propriétés des données qu'il manipule.

2.3 Descriptions des opérations : le comment

Forts de la description abstraite d'un problème, il nous est désormais possible de fournir une procédure permettant de le résoudre. Cette procédure est généralement fournie sous la forme d'une liste d'opérations (ou instructions) à exécuter. Ces dernières correspondent en réalité à des transformations qui sont appliquées de manière successive aux données d'entrées. Nous appelons cette étape le **comment**.

C'est durant cette étape, et en fonction des propriétés décrites à l'étape précédente, que nous serons en mesure d'identifier la complexité de l'algorithme. Certains problèmes peuvent s'avérer trop difficiles ou même impossibles à résoudre. Le *problème de l'arrêt* (Turing, 1937) est exemple connu de problème indécidable. Ce dernier consiste à déterminer si un programme termine ou non, à partir de la description de ses opérations. D'autres problèmes peuvent être montrés indécidables ou trop coûteux en général, mais toutefois disposer d'une solution si l'on contraint davantage leurs données d'entrées (c'est-à-dire en modifiant le quoi).

2.4 Un exemple : le tri de valeurs

Afin d'illustrer les étapes décrites ci-dessus, nous proposons de nous intéresser au problème du tri de valeurs. Il s'agit ici de construire un algorithme permettant de trier une liste de valeurs (numériques ou non). Nous allons procéder avec la même approche méthodique que nous avons présentée,

c'est-à-dire en identifiant les données à manipuler, en décrivant le quoi puis finalement le comment.

En parallèle aux explications *en prose* de la classification, du quoi et du comment, nous nous efforcerons également d'adopter une notation formelle, dans le but de bien illustrer la correspondance entre notations et signification.

La classification

Pour pouvoir représenter des listes, il nous faut un moyen d'associer un rang à chaque valeur. Pour se faire, nous admettrons que les données d'entrée de l'algorithme seront fournies sous la forme d'un ensemble de paires rang, valeur.

Formellement, nous utiliserons E pour représenter le type (ou domaine) de données à trier. Nous utiliserons les nombres naturels $\mathbb{N}$ pour représenter le rang d'une valeur dans une liste, et définirons le domaine d'entrée et de sortie de notre algorithme par $\mathcal{P}(\mathbb{N} \times E)$. Par exemple, la liste de lettres [d, a, p] serait représentée par l'ensemble $\{(d, 0), (a, 1), (p, 2)\}$.

Le quoi

Forts d'une description précise des domaines d'entrée et de sortie de notre algorithme de tri, nous pouvons désormais spécifier les contraintes de l'opération. Il convient tout d'abord de réfléchir aux contraintes qu'il pourrait exister sur le type de données que nous souhaitons manipuler. Étant donné que nous souhaitons exprimer un algorithme de tri de valeurs, il paraît donc naturel de ne considérer que des valeurs qui peuvent être comparées. Parce qu'historiquement il est coutume de définir le tri de valeurs *numériques*, il est souvent admis qu'une telle relation existe entre tout couple de valeurs. Or, il n'est pas nécessairement évident ou même possible de comparer les éléments d'autres types d'ensembles. Par exemple, il n'y aurait aucun sens à vouloir trier une liste des fruits et des animaux. Et quand bien même notre liste ne serait composée que de fruits, il conviendrait encore de définir un critère de comparaison, comme la taille ou le poids. Nous contraindrons donc l'ensemble des valeurs à trier à disposer d'un tel critère.

Formellement, soit E le domaine des données à trier, nous admettrons qu'il existe une relation $< \subseteq E \times E$ décrivant si un élément $e_1 \in E$ est plus petit qu'un élément $e_2 \in E$ (noté $e_1 < e_2$).

Enfin, nous considérerons que notre algorithme aura bel et bien correctement trié la liste qui lui aura été présentée si et seulement si :

1. Toutes les valeurs en entrée auront été préservées en sortie (i.e. pas de perte d'information)
2. Aucune valeur absente en entrée n'aura été ajoutée en sortie
3. Pour tout couple de paires rang, valeur, la paire avec le rang le plus petit sera associée à la valeur la plus petite.

Par exemple, si la liste $\{(d, 0), (a, 1), (p, 2)\}$ était présentée à notre algorithme, nous nous attendrions à obtenir la liste $\{(a, 0), (d, 1), (p, 2)\}$ en sortie.

Formellement, soit $l, l' \in \mathcal{P}(\mathbb{N} \times E)$ les listes d'entrée et de sortie respectivement, $\forall (i, e) \in l, \exists (i', e) \in l'$ (1re contrainte), $||l|| = ||l'||$ (2e contrainte) et $\forall (i_1, e_1), (i_2, e_2) \in l', i_1 < i_2 \Rightarrow e_1 < e_2$ (3e contrainte).

Le comment

La transformation de la description du quoi à une description du comment sous-entend généralement l'introduction d'algorithmes. Il est intéressant de constater que pour un énoncé de problème il correspond généralement plusieurs algorithmes. Dans certains cas il n'en existe pas ou que sur une version limitée du problème ; on dit alors que le problème est indécidable (domaine théorique de la calculabilité). Dans d'autres cas, les algorithmes possibles peuvent s'avérer très ou même trop coûteux en temps ou en espace (domaine théorique de la complexité). On peut dire que ce sont des limites naturelles liées aux possibilités d'automatisation.

Il existe une multitude d'algorithmes de tri qui respectent les contraintes que nous avons spécifiées ci-dessus. Dans le cadre de notre exemple, nous allons nous contenter d'en décrire un seul, appelé **bubblesort**. Le principe de cet algorithme est de déplacer successivement les valeurs inférieures de la liste vers le début de celle-ci, comme s'il s'agissait de bulles dans un liquide :

```
1. for let i in 0 .. |l| do
2.   for let j in i .. |l| do
3.     let (i, el) in l
4.     let (j, er) in l
5.     if el > er do
6.       l := l – {(i, el), (j, er)} | {(i, er), (j, el)}
```

Les premières lignes de l'algorithme décrivent deux boucles, c'est-à-dire une répétition d'instructions. L'objectif est de répéter les lignes 3 à 6 pour chaque paire de valeurs dans la liste à trier. Les lignes 3 et 4 extraient la paire de couples à comparer. La ligne 5 vérifie que la valeur à gauche ne soit pas plus grande que celle à droite, le cas échéant la ligne 6 inverse les valeurs associées au rang des couples extraits.

L'algorithme proposé ci-dessus est indépendant du langage de programmation que l'on pourrait utiliser pour l'implémenter. C'est là que nous souhaitons arrêter notre démarche intellectuelle, car nous sortirions du domaine abstrait et aurions à nous heurter aux problématiques liées à la technologie (sémantique du langage, mémoire à disposition, etc.). Plus tard, nous verrons même qu'en restant à ce niveau d'abstraction, il est possible *d'exécuter* cet algorithme de manière bien plus concrète, avec des objets physiques. Ceci souligne donc l'intérêt d'appliquer des abstractions.

3 Notre action sur le terrain

Dans le cadre de campagnes de promotion de l'université, et en particulier des études d'informatique, nous avons tenté d'appliquer la démarche que nous avons présentée ci-dessus pour mener diverses présentations et activités interactives. Entre autres, nous avons participé en 2016 à l'événement *Science Me!* (s. d.), une compétition annuelle dont le but est de présenter les sciences au grand public, par le biais d'une présentation scénarisée. En mettant en scène une sorte de simili de programme culinaire, nous avons expliqué la signification du terme *algorithme*, en tirant un parallèle avec une recette de cuisine. Nous avons également apporté notre contribution dans diverses journées d'initiation, notamment plusieurs éditions du TecDay (*Un collège se transforme en gigantesque laboratoire*, 2017), où nous animons chaque année une activité durant laquelle des élèves de 15 à 20 ans sont invités à mettre au point puis expérimenter un algorithme réparti.

Ces deux expériences ont été accueillies avec succès et nous ont confortés dans la pertinence de notre approche. Elles nous ont également permis de raffiner notre approche au cours du temps. Par exemple, nous

nous sommes rendu compte que l'expérimentation était une composante essentielle à la compréhension.

3.1 Nuit de la science

Une première expérience consistait en la mise à disposition des collégiens d'un environnement de programmation de robot afin de réaliser une tâche simple. Cette approche était très satisfaisante en termes d'enthousiasme et d'investissement des collégiens aux tâches proposées. Néanmoins nous déplorions à chaque événement que le public perçoive notre activité comme *un jeu avec des robots*, plutôt que d'un atelier de découverte de l'algorithmique.

C'est ce constat qui nous a poussés à nous éloigner de la technologie. Notre activité se focalisait trop sur le simple fait de déplacer des robots, et pas sur une réflexion sous-jacente quant à l'algorithme décrivant la tâche à réaliser, car les étudiants pouvaient se contenter *d'essayer* des séries d'instructions jusqu'à parvenir à la solution escomptée.

3.2 Science Me

Science Me! est une compétition amicale regroupant plusieurs équipes de scientifiques souhaitant partager leur vision de la science par le biais de présentations mises en scène. Introduire l'informatique dans ce type de cadre représente souvent un défi, car à l'inverse de sciences telles que la physique ou la chimie, il est difficile en informatique de présenter des résultats spectaculaires à un public profane. C'est pourquoi au lieu de partir des applications plus communes de l'informatique, telles que la programmation ou encore la robotique, nous avons fait le choix de miser sur une activité *a priori* complètement annexe : la cuisine. Nous avons représenté une recette de gaspacho sous forme de réseau de Petri (Reisig, 1985), un formalisme informatique permettant d'exprimer formellement tout type de processus. La figure 1 présente le réseau complet. Lors de la présentation, nous avons ensuite illustré pas à pas le déroulement de l'algorithme, en réalisant tout simplement notre recette. Non seulement ce format nous a permis d'expliquer la notion d'algorithme, mais également d'introduire une notation formelle. Les retours du public ont été très positifs et ont clairement confirmé

l'intérêt de notre approche. En utilisant un domaine *a priori* connu de tous pour illustrer un concept informatique, les notions que nous souhaitions transmettre se sont avérées bien plus faciles à communiquer que si nous étions partis d'une application plus directement liée.

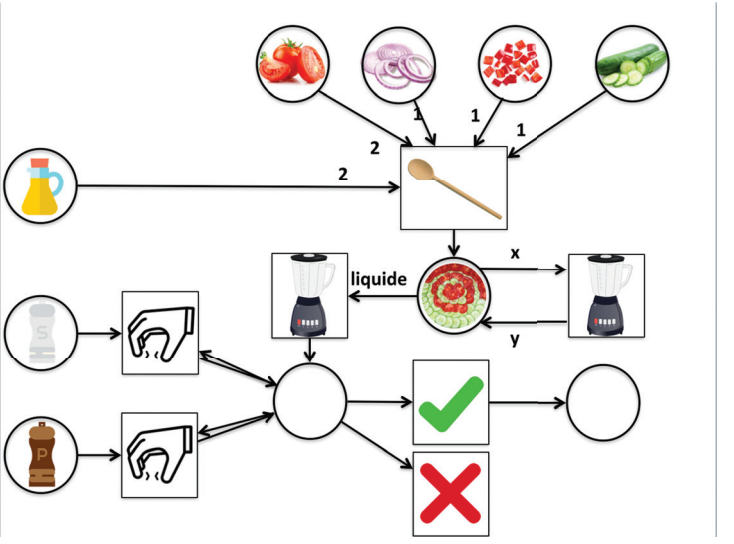

Figure 1 : Réseau de Petri représentant la préparation d'un gaspacho.

3.3 TecDay

Le TecDay est initiative de l'Académie Suisse des Sciences Techniques (SATW). Il s'agit d'une journée d'activités organisée dans les collèges de toute la Suisse, dont le but est d'offrir aux élèves la possibilité de rencontrer des chercheurs et ingénieurs afin que ceux-ci leur présentent leurs travaux. Sur deux années consécutives, nous avons proposé aux étudiants du collège Sismondi puis à ceux du collège Rousseau de se familiariser avec la notion d'algorithme. L'expérience consiste à mettre au point un algorithme distribué d'élection de leader. Dans un premier temps, le problème et ses contraintes sont présentés aux élèves, qui sont alors invités à trouver par eux-mêmes une solution. Les forces et faiblesses de ces solutions sont discutées en groupes, puis nous proposons notre propre alternative. Avec le support d'une plaquette (illustrée par la

figure 2) représentant la mémoire interne dont disposerait un ordinateur, nous exécutons notre algorithme en groupe, dans le but de bien comprendre chaque étape de son déroulement.

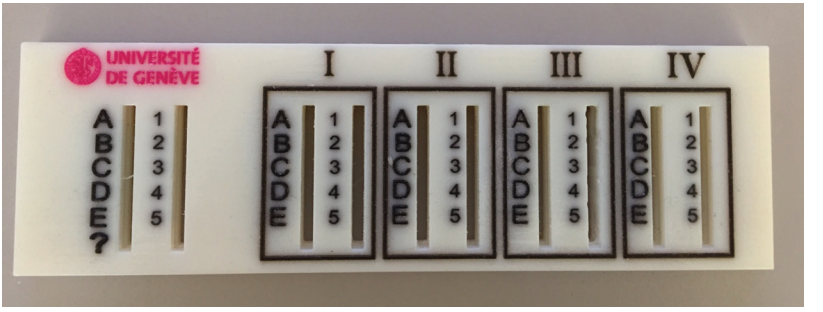

Figure 2 : Plaquette représentant la mémoire d'un ordinateur.

Les retours de cette expérience ont également été des plus encourageants, et ce sur les deux années. Un aspect intéressant du format du TecDay est qu'il nous permet de faire participer le public, en premier lieu de par une recherche de solution par rapport à un problème donné, puis par l'expérimentation.

4 Démarche générale

Dans nos trois expériences, l'élément clé est que nous avons approché des concepts informatiques abstraits (algorithmes et structures de données) par le biais d'une application concrète, parfaitement à la portée du public. Ceci permet de démystifier les termes, en proposant de les illustrer avec des notions plus familières, et de les expérimenter plus directement. Parce que nous faisons abstraction des considérations technologiques, notre approche est en fait très théorique. Néanmoins, parce que nous la lions très fortement à un domaine familier, elle peut être approchée avec bien plus de simplicité. En outre, si nos interventions au TecDay et à la nuit de la science peuvent être perçues comme plus proches d'une application classique de l'informatique, notre participation à *ScienceMe!* démontre que

des domaines *a priori* très éloignés peuvent toutefois servir de supports appropriés pour l'illustration de concepts abstraits.

Nous en déduisons une démarche plus générale :

1. Identifier un domaine d'application familier du public ciblé
2. Identifier les concepts informatiques qui peuvent être appliqués au domaine
3. Mettre au point une expérience pratique permettant d'illustrer les concepts identifiés

En partant du domaine d'application, nous nous assurons que le public puisse comprendre l'intérêt de l'approche informatique. Par exemple, dans le cas de la recette de cuisine, il apparaît évident le besoin de disposer d'une marche à suivre détaillant les étapes de réalisation. L'introduction de l'algorithmique devient alors naturelle. De la même manière, de la réflexion autour des contraintes liées à la création d'une procédure d'élection, il apparaît évident le besoin de décrire précisément les objectifs et contraintes associés.

Si nos expériences se sont principalement concentrées sur la notion d'algorithme, notre approche peut être appliquée à d'autres concepts informatiques. Par exemple, nous évoquions dans l'introduction qu'Internet reposait sur des concepts relativement méconnus. En partant de l'Internet comme domaine d'application familier, une possible activité consisterait à étudier les différents composants nécessaires à son fonctionnement. Une première étape inviterait le public à identifier ces composants, dans le cadre d'une discussion de groupe. Puis, une seconde étape consisterait à distribuer le rôle de chaque composant, avant de jouer le transfert d'une donnée d'un serveur à un client.

5 Mise en œuvre ludique d'algorithmes

Dans le chapitre précédent, nous avons donné une démarche générale pour l'approche de concepts informatiques. Dans ce chapitre, nous proposons d'illustrer cette démarche avec un exemple d'activité pédagogique. Nous reprendrons l'algorithme de tri présenté dans le chapitre 2, et adapterons son processus de création à notre activité.

Notre objectif sera de trier un jeu de 52 cartes dans l'ordre classique, c'est-à-dire de l'as au roi, de la couleur pique suivie de cœur, carreau et trèfle.

La classification

Dans le chapitre 2, nous avons représenté les listes par l'intermédiaire de couples rang, valeur. Nous ferons de même ici, en utilisant un paquet de cartes simplement numérotées de 1 à 52. Ce second jeu de cartes représentera les rangs, alors que le premier représentera des valeurs. Dès lors, nous pourrons introduire deux notions fondamentales : les ensembles (ou types) ainsi qu'une poignée d'opérations sur ces ensembles.

Dans le chapitre 2, nous avons exprimé formellement les listes sous la forme d'un produit cartésien $\mathbb{N} \times E$. En fait, ceci correspondra dans notre activité à former l'ensemble de tous les couples de cartes qui contiennent une carte de chaque paquet. Nous pourrons même représenter cela physiquement, sous la forme d'une sorte de support que le public pourra utiliser pour associer les cartes (voir figure 3).

Nous pouvons déjà souligner l'illustration par des moyens concrets d'une notion abstraite : le produit cartésien. En fait, il est intéressant de constater qu'en fournissant des supports pour plus un nombre arbitraire n de cartes, il est possible de représenter des n-uplets, c'est-à-dire des représentants de produits cartésiens de n ensembles (i.e. $S_1 \times S_2 \times \cdots \times S_n$).

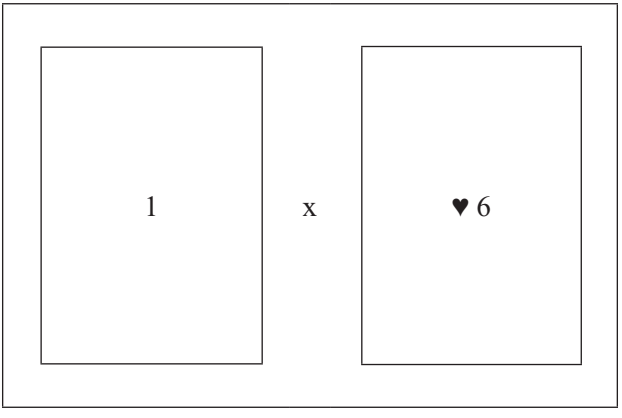

Figure 3 : Support pour couple rang, valeur.

Le quoi

Dans le cadre de cet exemple, décrire le quoi représente une sous-activité très intéressante. En effet, s'il paraît très simple d'énoncer l'objectif d'un tri, déterminer toutes les contraintes qui décrivent son succès n'est pas si évident. Il sera ici l'occasion de mener une discussion de groupe et inviter le public à se rendre compte de l'imprécision probable d'une description profane. Une erreur fréquente sera sans doute de ne considérer que l'ordre de la liste produite par l'algorithme, mais pas la préservation des valeurs qui auront été fournies en entrée. Sans cette condition, la liste vide serait une réponse parfaitement acceptable à n'importe quelles données d'entrée.

Un autre axe sera de discuter de l'ordre lui-même. Traditionnellement, un jeu de cartes est trié de l'as au roi, en partant de la couleur pique, suivie de cœur, carreau et trèfle. L'énonciation précise d'un tel ordre représente également un exercice intéressant.

Le comment

Finalement, la dernière étape de l'activité consistera à décrire un moyen de parvenir à réaliser l'objectif principal (i.e. le tri du jeu de cartes), tout en respectant les contraintes qui auront été identifiées lors de l'étape précédente. Il sera là encore l'occasion d'entreprendre une discussion de groupe pour tenter de coucher sur papier une méthode de résolution. Tout comme nous le faisons au TecDay, nous pourrons ensuite proposer une ou plusieurs solutions, et discuter de leurs tenants et aboutissants. Il est à noter que le tri constitue un problème qui se prête extrêmement bien à ce genre d'exercice, car il n'est pas très compliqué ni d'identifier différents algorithmes, ni d'en comprendre les implications, notamment en termes de complexité.

Muni des jeux de cartes et des supports permettant de représenter des éléments de produit cartésien, le public finira par être invité à expérimenter un ou plusieurs des algorithmes qui auront été élaborés au préalable.

6 Conclusion

Tout comme les mathématiques et la physique, l'informatique souffre souvent d'une image trop abstraite, ayant tendance à dissuader le grand public

de s'y intéresser. Néanmoins, il apparaît évident que l'enseignement de ses concepts fondamentaux est devenu indispensable dans le contexte actuel. Dans ce document, nous avons montré qu'en adoptant une approche moins théorique et plus basée sur la réflexion et l'expérimentation, l'apprentissage de notions jugées *a priori* complexes et très abstraites peut être grandement facilité. En particulier, nous avons montré qu'il était relativement aisé d'introduire la notion d'algorithme dans des domaines familiers du grand public, et ainsi démystifier le terme ainsi que le concept qu'il représente. Nous avons également montré qu'une approche similaire pouvait être utilisée pour l'introduction d'autres notions et concepts. Et finalement, nous avons illustré l'application de notre démarche dans la réalisation d'une activité pédagogique.

Notre démarche est issue de réflexions sur les expériences que nous avons menées lors de campagnes de promotion de l'université. Ce cadre représente pour nous une opportunité intéressante d'expérimenter diverses approches pédagogiques, afin de raffiner notre discours et notre approche.

Références

Baron, G.-L., & Bruillard, E. (2001). Une didactique de l'informatique ? *Revue française de pédagogie*, 163–172. Reisig, W. (1985). *Petri nets : An introduction*. New York, NY, USA : Springer-Verlag New York, Inc.

Schiper, A. (2016). *Découvrir le numérique : une introduction à l'informatique et aux systèmes de communication*.

Presses polytechniques et universitaires romandes.

Science me! (s. d.). <http://scienceme.unige.ch>. (Accessed : 2017–07-12).

Turing, A. M. (1937). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London mathematical society*, *2*(1), 230–265.

Un collège se transforme en gigantesque laboratoire. (2017). <goo.gl/LA8xh9>. (Accessed : 2017–07-12).