Le Base-ball multicolore : pensée algorithmique et raisonnement*

Maryline Althuser[1], Nathalie Brasset[2], Anne Rasse[3], Jean-Marc Vincent[3] et Benjamin Wack[3]

[1]Collège Stendhal, Grenoble
[2]Lycée des Portes de l'Oisans, Vizille
[3]Université Grenoble Alpes

11 juin 2019

Résumé Les activités débranchées permettent d'aborder des concepts fondamentaux de l'informatique avec des participants de tous horizons. Nous explorons ici le problème du Base-ball multicolore, accessible pour des élèves du cycle 3 jusqu'au lycée, pour lequel ils élaboreront des solutions algorithmiques ; l'activité est adaptable au cycle 2 avec des objectifs différents et des modalités aménagées. Les formes classiques de raisonnement des mathématiques fournissent des pistes pour résoudre le problème avec rigueur.

1 Introduction : l'informatique débranchée

L'informatique débranchée, largement popularisée par [Bell *et al.*, 2014], consiste à créer des activités sur la base de manipulation d'objets concrets (ficelles, jetons, plaques de bois...) afin que les élèves découvrent des concepts fondamentaux de l'informatique sans nécessiter de maîtrise technique. Les activités se font pour la plupart en groupe, à partir de règles du jeu simples, pour stimuler la coopération et la communication. Les activités doivent être suffisamment guidées (soit par le questionnement du professeur, soit par le questionnement de l'activité elle-même) pour que l'élève intègre par lui-même le concept. On peut faire une analogie avec les situations de recherche (résolution de problèmes complexes), ainsi qu'avec la démarche scientifique présente dans toutes les disciplines scientifiques.

Comme le fait remarquer Roberto Di Cosmo (qui participa à la traduction française de [Bell *et al.*, 2014]), *« Si on peut passer des heures à cliquer sur une souris sans rien apprendre d'informatique, on peut aussi apprendre beaucoup d'informatique sans toucher une souris. »* Nous irons même un peu plus loin, en affirmant que ne pas utiliser l'ordinateur permet parfois une compréhension

*Travail réalisé avec le soutien de l'IREM de Grenoble, de la Maison Pour la Science en Alpes-Dauphiné, du rectorat de l'académie de Grenoble et d'Inria Grenoble-Rhône-Alpes

accrue de la science informatique, ce que semblent confirmer les travaux de [Brackmann *et al.*, 2017].

Cet article présente le problème du Base-ball multicolore et propose sa mise en œuvre avec des élèves dans le cadre d'une activité d'informatique débranchée. On étudie ensuite quelques algorithmes permettant de résoudre ce problème et on présente, pour finir, les notions qui peuvent être institutionnalisées suite à une telle activité. Pour les élèves de lycée, on peut prolonger l'activité par des questions sur la complexité des algorithmes.

Les stratégies de résolution possibles étant multiples, on se pose la question de leur vérification. Pour se se convaincre qu'une solution est correcte ou non, on est amené à mobiliser le raisonnement par l'absurde et la recherche de contre-exemple ; pour construire une solution à un problème de taille quelconque, c'est la récurrence qui intervient. Ces formes de raisonnement trouvent ainsi une opérationnalisation dans un contexte différent de celui habituellement rencontré par les élèves.

2 Description du problème du Base-ball multicolore

Historique On retrouve une première version de ce problème sous la forme du *jeu de l'orange* dans [Bell *et al.*, 2014] : l'activité y est très orientée vers la notion d'interblocage dans les réseaux, et cherche à faire prendre conscience aux élèves de la nécessité de coopérer pour résoudre ensemble le problème posé (en particulier parce qu'un participant peut bloquer les autres s'il s'arrête de participer dès qu'il a résolu « sa » tâche). Plus tard, on trouve dans [Quinson, 2014] la présentation sous forme de *Base-ball multicolore*, et des questions très algorithmiques (validité d'une solution, réduction d'un problème à un autre). La version de l'IREM de Grenoble présentée ici questionne les raisonnements logiques menant à l'écriture et à la validation d'un algorithme ; nous étudions aussi dans une moindre mesure la complexité algorithmique du problème et de ses solutions.

Situation et objectifs On dispose de cinq bases de couleurs différentes, disposées en cercle et comportant chacune deux emplacements. Par ailleurs, on dispose de deux pions de la même couleur associés à chaque base, sauf une d'entre elles qui ne possède qu'un seul pion. On a donc un pion de moins qu'il n'y a d'emplacements, et on dispose tous les pions de façon aléatoire sur les emplacements disponibles.

Le but du jeu est de déplacer les pions afin d'amener chaque pion sur la base correspondant à sa couleur.

Trois contraintes sont imposées sur les déplacements :

— on déplace un seul pion à la fois ;
— un pion ne peut se déplacer vers une base que si elle possède un emplacement libre ;
— un pion se déplacer uniquement vers les deux bases voisines, le long du cercle (il ne peut pas traverser le plateau de jeu).

Il y a donc exactement 4 coups valides à partir de toute disposition des pions.

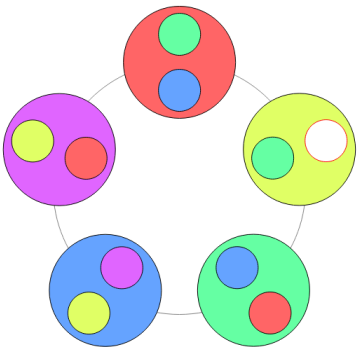

FIGURE 1 – Une configuration initiale

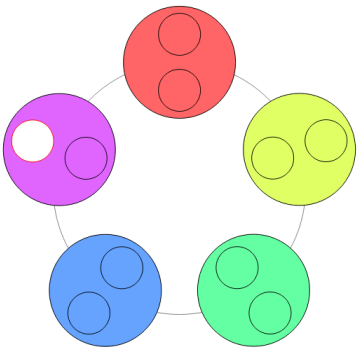

FIGURE 2 – Configuration finale

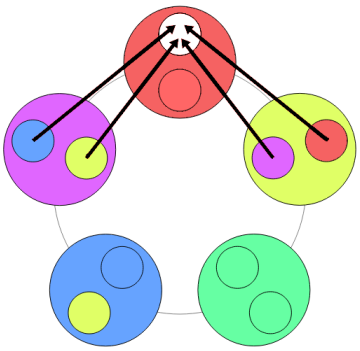

FIGURE 3 – Coups autorisés

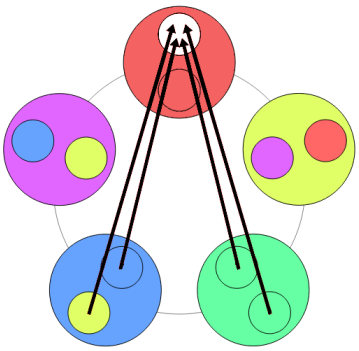

FIGURE 4 – Coups interdits

Le problème n'est en général pas très compliqué à résoudre sur une configuration donnée, mais on cherche ici une méthode fonctionnant à partir de toutes les configurations, autrement dit un algorithme. On s'attache en particulier à l'expression et à la rédaction de cette méthode.

3 Mise en œuvre en classe

L'activité est proposée à des élèves à partir du cycle 3. Cette situation est bien adaptée à un travail en groupes de 4 élèves environ.

Travail de groupe On propose une recherche en quatre étapes :

1. Manipulation libre

   Pour bien comprendre le problème posé, la première étape consiste, assez naturellement, à « jouer avec » et à chercher, en tâtonnant, des solutions. On laisse donc les élèves manipuler les pions sur les bases et essayer de trouver une solution à partir de différents états de départ choisis au hasard. En cas de difficulté à démarrer, on peut réduire le nombre de bases à 4.

   Une étape cruciale est de leur faire comprendre qu'on cherche une stratégie générale, et pas seulement une suite de déplacements en partant d'une situation particulière. On peut leur poser la question suivante : *Pensez-vous que l'on peut résoudre le problème à partir de n'importe quelle situation de départ ?*

2. Élaboration d'une stratégie

   Les élèves doivent maintenant essayer de suivre une stratégie jusqu'au bout.

   On leur fera comprendre l'intérêt de construire un algorithme et de le suivre jusqu'au bout. Si on arrive à formaliser un algorithme et que cet algorithme permet d'arriver à une solution quelque soit l'état de départ, alors cet algorithme peut être programmé sur un ordinateur et la résolution peut être automatisée.

   Une aide possible pour guider la construction d'une solution algorithmique : se donner des contraintes supplémentaires pour éviter la combinatoire du problème. Si on a une solution dans le cas contraint, on aura également une solution dans le cas général. Quelques exemples de contraintes qu'on peut proposer aux élèves :
   — tourner toujours dans le même sens ;
   — couper le cercle et se déplacer sur une ligne ;
   — restreindre le problème (à un nombre de bases plus petit par exemple) ;
   — commencer par résoudre un problème plus simple (comme déplacer ou échanger des pions particuliers).

3. Rédaction et validation d'une méthode

   Un passage obligé pour aller plus loin consiste à rédiger un algorithme candidat à la résolution du problème, ce qui peut s'avérer assez difficile. On peut donc commencer par l'expliciter à l'oral, et le clarifier par un dialogue avec l'enseignant. Ensuite, les élèves peuvent le mettre en œuvre avec les pions et les bases en choisissant un état de départ au hasard : l'un

des élèves du groupe sera l'ordinateur et un autre sera le programmeur. Le programmeur *ne voit pas* les positions des pions ; il dicte à l'ordinateur une série d'instructions qui vont permettre de remettre les pions sur la bonne base. L'ordinateur exécute mécaniquement les instructions proposées par l'ordinateur.

Un 3e élève du groupe peut jouer le rôle de l'adversaire : il doit essayer de proposer une configuration de départ telle que les autres joueurs du groupe ne puissent pas résoudre le problème à l'aide de l'algorithme qu'ils ont proposé.

Pour valider cette étape, le professeur peut demander à ses élèves : « Avez vous trouvé un algorithme qui permette d'arriver, quelle que soit la situation de départ, à ce que tous les pions retrouvent leur base ? ». Le cas échéant, il est tout aussi intéressant de trouver un contre-exemple montrant qu'un algorithme est incorrect ou ne se termine pas.

4. Vérification

   Lorsqu'on rédige un algorithme qu'on a soi-même élaboré, il est tentant de laisser de l'implicite dans certains passages, au motif qu'on saura bien quoi faire quand la situation se présentera ; ce qu'une machine ne pourra bien sûr pas faire si on programme l'algorithme. Pour remédier à cet écueil et faire sentir la nécessité d'une rédaction rigoureuse, on peut suivre la méthodologie suivante :

   — Le groupe A rédige son algorithme sur une feuille qu'il confie à un autre groupe, qu'on appellera B.
   — Simultanément, un autre groupe (appelons-le C) aura donc confié son propre algorithme au groupe A.
   — La tâche du groupe A est alors de lire attentivement l'algorithme du groupe C, essayer de le comprendre, de l'exécuter, voire de déterminer s'il semble correct ou non.
   — Le groupe A désigne ensuite un rapporteur parmi ses participants, qui va aller réexpliquer leur *propre algorithme* au groupe C.
   — Si les groupes sont bien synchrones, un rapporteur du groupe B devrait au même moment venir réexpliquer son propre algorithme au groupe A.

   On peut ainsi, dans chaque groupe, faire prendre conscience des points mal formulés voire manquants dans sa rédaction, et aboutir à une rédaction plus compréhensible.

Mise en commun Groupe par groupe, les élèves peuvent enfin présenter l'un des algorithmes qu'ils ont formalisé au reste de la classe. Idéalement, on essaie de commencer par un algorithme qui fonctionne pour certains états de départ, mais pas pour d'autres ; si possible, on cherchera avec le reste de la classe comment modifier cet algorithme pour qu'il marche pour toutes les configurations de départ. Cette mise en commun est également l'occasion de faire le rapprochement entre deux algorithmes basés sur des idées identiques, mais formulés différemment.

Un outil pratique pour la mise en commun est proposé à l'adresse :
`https://iso.gricad-pages.univ-grenoble-alpes.fr/ISO/Base-Ball/`
qui permet de vidéoprojeter les déplacements des pions sur les bases pour illustrer une exécution d'un algorithme. Il propose également de choisir une situation

de départ aléatoire, de sauvegarder une situation pour y revenir plus tard, ou encore d'exécuter automatiquement certains des algorithmes mentionnés dans la section suivante.

4 Quelques algorithmes possibles

On étudie dans cette partie plusieurs « algorithmes » qui peuvent être proposés par les participants. On discute brièvement le principe qui a guidé leur conception, leurs défauts et leurs mérites respectifs.

4.1 Algorithme tournant

Dans n'importe quelle configuration du problème, on a un choix à faire entre 4 coups distincts (2 pions sur chacune des 2 bases adjacentes au trou).

À première vue, il n'est pas facile de choisir le « bon » coup parmi ces quatre, ni d'exprimer simplement comment on fait ce choix. Pour réduire cet espace de choix, on décide de ne tourner que dans un seul sens (horaire par exemple). Ainsi, un coup se résume à décider lequel de 2 pions on fait avancer.

Partant de ce principe, on arrive rapidement à un premier algorithme :

— On ne s'autorise à tourner que dans un seul sens. Ainsi, le nombre de coups possibles descend de 4 à 2 (car 2 pions tourneraient à l'envers).
— Parmi les 2 coups restants, on déplace le pion qui a la plus grande distance à parcourir avant d'arriver à sa base (si la distance est la même, c'est que les deux pions ont la même couleur ; les deux coups sont alors équivalents).
— Tant que tous les pions ne sont pas rentrés à leur base, on continue les déplacements.

Cet algorithme est simple et efficace, mais faux ! En effet, il existe des états de départs pour lesquels cet algorithme ne permet pas d'arriver à la solution. Par exemple, considérons la configuration de la figure 5. Sur chaque base, initialement un pion est à sa place, et ne sera donc jamais déplacé par notre algorithme. Quant aux pions mal placés, ils ne peuvent tourner que dans le sens horaire, et ne peuvent pas emprunter le chemin bloqué par les pions bien placés. Ils sont donc contraints à tourner en conservant leur ordre respectif, et n'ont aucune possibilité de permuter pour se retrouver dans le bon ordre (dès qu'un ou deux d'entre eux sont bien placés, les autres sont forcément sur la mauvaise base).

L'algorithme tourne donc en boucle sans jamais trouver de solution ! Il ne mérite que le nom de « semi-algorithme », ou d'algorithme *partiellement correct*.

Cet exemple est intéressant puisqu'il est très facile de se convaincre (à tort) que cette méthode fonctionne : en effet, elle résout le problème dans presque toutes les situations initiales où aucun pion n'est dans sa propre base : 100% à 4 bases, 98,7% à 5 bases et 97,8% à 6 bases. C'est l'occasion de faire remarquer que s'il existe ne serait-ce qu'une seule situation de départ pour laquelle l'algorithme ne termine pas, alors celui-ci est incorrect, de la même façon qu'un seul contre-exemple suffit à invalider une proposition logique.

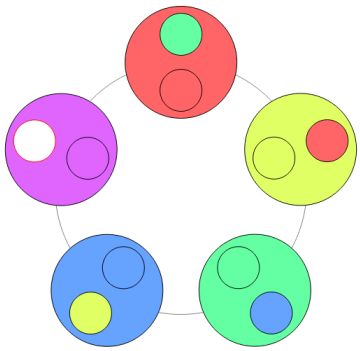

FIGURE 5 – Configuration bloquante pour l'algorithme tournant

4.2 Algorithme en ligne

Un deuxième algorithme proposé est inspiré du raisonnement par récurrence. Généralisons le problème avec n bases. Si on sait :

— placer deux pions sur leur base ;
— et résoudre le problème pour les $n-1$ bases restantes ;

alors on sait résoudre le problème à n bases.

Il faut tout de même prendre quelques précautions : le problème à $n-1$ bases doit rester comparable au problème de taille n.

— On « coupe » donc le cercle entre la base à 1 pion et une de ses voisines. Cette fois on restreint l'espace des coups possibles en transformant notre cercle en une ligne, comme sur la figure 6.
— On peut numéroter les bases de 1 (celle qui n'a qu'un pion à sa couleur) à n (la plus éloignée).
— On commence par remplir la base numéro n.

L'initialisation de la récurrence est fournie par le cas où on n'a plus qu'une base avec un seul pion à sa couleur, et pour laquelle le problème est trivialement résolu.

Il reste à démontrer qu'on sait remplir la base numéro n dans tous les cas.

1. Pour rapprocher un pion de la couleur n de sa base, on déplace d'autres pions (sans tenir compte de leurs couleurs) pour « amener le trou » à côté du pion à déplacer, du côté de sa base. Si le pion n est situé sur la base k, on cherche donc à amener le trou en $k+1$. Sur la figure 6, on cherche à approcher le pion marqué 5 de la base 5, on commence donc par effectuer les deux déplacements indiqués.

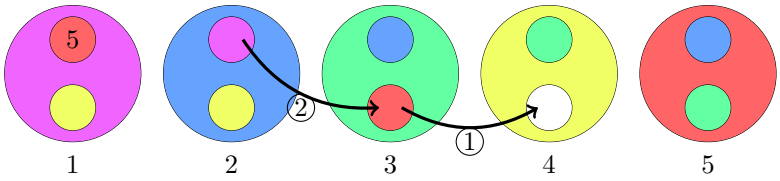

FIGURE 6 – Algorithme en ligne : « déplacer le trou » à proximité d'un pion de couleur n

2. On déplace ensuite ce pion dans le trou (base $k+1$).
3. Le trou est désormais à gauche de notre pion, sur la base k : on peut facilement amener le pion en $k+2$ en 2 déplacements, puis reprendre à l'étape 2 autant de fois que nécessaire pour que notre pion de la couleur n soit revenu à sa base, et on n'y touche plus.
   La figure 7 montre à la fois comment rapprocher le pion 5 de sa base, et replacer le trou à sa droite pour préparer l'itération suivante.

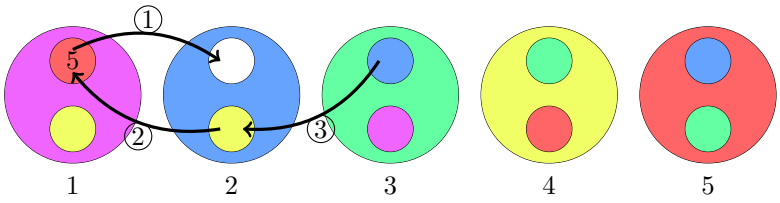

FIGURE 7 – Algorithme en ligne : faire progresser un pion de couleur n vers sa base

4. On procède de même avec l'autre pion de couleur n.

On peut maintenant ignorer la base n, qui est déjà traitée, et on recommence avec les bases restantes.

Preuve de l'algorithme Il est assez clair que, lorsque cet algorithme s'arrête, c'est qu'on a rempli toutes les bases correctement. L'essentiel du travail consiste donc à démontrer qu'on y arrive en un nombre fini de déplacements :

— Amener le trou à proximité d'un pion (étape 1) demande au maximum $n-2$ déplacements.
— Amener un pion sur sa base (étapes 2 et 3) demande au maximum $n-1$ déplacements dudit pion et $2(n-2)$ déplacements pour « replacer le trou à droite ».
— Remplir la base n demande d'amener successivement 2 pions sur cette base, soit un total de $2((n-2)+(n-1)+2(n-2)) = 8n-14$ déplacements.
— Résoudre le problème demande de remplir correctement $n-1$ bases (et la base à 1 pion sera automatiquement remplie correctement).

Ce raisonnement nous donne d'ailleurs une bonne estimation du coût de l'algorithme : moins de $\sum_{i=n}^{2}(8i-14)$ déplacements. Le coût total est donc borné par un polynôme du second degré dont le terme dominant est $4n^2$, autrement dit la complexité de l'algorithme est en $\mathcal{O}(n^2)$.

4.3 Décomposition en sous-problèmes

Une troisième façon de construire un algorithme est de décomposer le problème en sous-problèmes. On a déjà appliqué en partie cette démarche de décomposition pour l'algorithme linéaire : pour remplir une base il suffit de savoir rapprocher un pion, et pour rapprocher un pion il suffit de savoir déplacer le trou.

On peut aussi se poser la question en ces termes : existe-t-il une opération de base telle que, si je sais l'effectuer, alors je suis sûr de savoir résoudre le problème ? On trouve pour cela deux bons candidats.

— **Si on sait échanger 2 pions voisins sans modifier les autres**
On peut alors implémenter un tri par insertion ou un tri à bulles, dont on sait qu'ils sont corrects. Il existe ainsi un certain nombre d'algorithmes classiques, souvent enseignés dans les cursus d'informatique, et dont on peut chercher à se rapprocher lorsqu'on met au point une solution pour un problème similaire.

— **Si on sait échanger un pion donné avec le trou sans modifier les autres pions**
On applique autant de fois que nécessaire cette opération pour placer à chaque fois un pion de la bonne couleur sur la base qui contient le trou. Dans le cas où le trou serait situé sur la base à un seul pion et où ce pion y serait déjà, il suffit de mettre dans le trou n'importe quel pion mal placé. Ainsi, on résout le problème en effectuant au pire $3n$ appels à une procédure qui est elle-même de complexité $\mathcal{O}(n)$.

La réalisation de chacune de ces deux opérations est détaillée en annexe A.

5 Variantes et prolongements

5.1 Autres mises en œuvre

Pour le cycle 2 voire en début de cycle 3, on peut proposer une mise en œuvre matérielle un peu différente :

— On fait porter aux enfants des chasubles ou des badges de couleur.

— On leur fournit des objets (bâtons, balles, ...) aux couleurs des chasubles.

Les enfants jouent alors le rôle des bases ; chacune de leurs mains correspondent à un des deux emplacements de la base, et les pions à déplacer sont matérialisés par les objets (on veillera à ce qu'il ne soit pas possible de tenir deux objets dans la même main).

Dans cette version, les élèves sont impliqués plus physiquement dans l'activité ; la participation active de chacun devient indispensable pour résoudre le problème.

En revanche il devient plus difficile de prendre le recul nécessaire à la résolution générale du problème, et on restera plus au niveau de l'exploration libre et de la recherche d'une stratégie.

Une dernière mise en œuvre est possible, qui demande plus d'espace mais offre l'avantage de mettre les élèves par groupes de 9, ce qui fait moins de groupes à suivre simultanément pour l'enseignant :

— Les élèves sont toujours munis de chasubles ou de badges de couleur.

— On dispose cette fois au sol des tapis aux couleurs des chasubles. Les tapis doivent être assez grands pour que 2 personnes se tiennent sur chacun.

Les enfants jouent alors le rôle de pions et ils se déplacent de tapis en tapis, qui matérialisent les bases.

5.2 À propos de complexité

Chez les élèves plus âgés, au lycée par exemple, on peut passer à un niveau d'abstraction supplémentaire et, dès la présentation de la situation, leur demander d'écrire un algorithme qui résout le problème quel que soit le nombre n de bases. On s'intéressera alors à l'ordre de grandeur du nombre de déplacements nécessaires en fonction du nombre de bases, comme souvent en algorithmique lorsqu'il s'agit de quantifier l'efficacité d'une solution.

Remarquons que le meilleur algorithme possible fera au moins de l'ordre de n^2 déplacements dans le pire cas. En effet, chaque pion doit être déplacé au moins autant de fois que la distance à sa base dans la configuration initiale. Or il existe des configurations dans lesquelles la somme de toutes ces distances est de l'ordre de n^2, par exemple lorsque chaque pion est placé de façon diamétralement opposée à sa base.

Par ailleurs, on connaît des algorithmes qui résolvent le problème en faisant de l'ordre de n^2 déplacements au pire, comme on l'a montré à la section 4.2 pour l'algorithme en ligne.

La complexité du problème (c'est-à-dire la complexité du meilleur algorithme permettant de le résoudre) est donc connue, et elle est de $\Theta(n^2)$.

6 C'est de l'informatique

Les activités débranchées étant souvent fortement scénarisées, un écueil possible est que les élèves n'en mémorisent que le thème et ne soient pas capables de réinvestir les notions sous-jaçentes dans d'autres situations. Il est donc indispensable de conclure l'activité par un temps d'institutionnalisation dans lequel on explicite les notions travaillées.

Nous citons ici trois messages qu'on peut véhiculer à l'aide du Base-ball multicolore, en fonction du niveau des participants.

6.1 Raisonnement

Un parallèle est possible avec les activités de démonstration menées en cours de mathématiques. Ainsi, pour montrer qu'un algorithme ne convient pas, comme l'algorithme tournant mentionné plus haut, il suffit de trouver un contre-exemple... ce qui n'est pas toujours facile à mettre en évidence !

A contrario, pour montrer qu'un algorithme est correct il est possible de :

— Tester tous les cas : on vérifie que l'algorithme trouve la solution au problème dans toutes les situations initiales ; cela s'avère infaisable en pratique.

— Écrire une preuve : on démontre que l'algorithme fonctionne dans le cas général.

— Montrer qu'on peut se ramener à un algorithme classique, celui-ci ayant été prouvé par ailleurs.

6.2 Construction et expression des algorithmes

Une difficulté en algorithmique est de trouver « l'idée » qui permettra de résoudre le problème à coup sûr, et de nombreux élèves souffrent du syndrôme de la page blanche dans cette situation. On pourra consulter [Pólya, 1945] pour approfondir cette question.

On peut ici leur expliquer qu'un algorithme consiste à décomposer la solution du problème en une suite d'étapes élémentaires ; pour y arriver, une bonne façon de procéder est de subdiviser le problème en problèmes plus simples, et de résoudre chacun d'entre eux séparément, comme on l'a fait pour certains des algorithmes proposés plus haut.

Une fois la stratégie générale trouvée (et une fois qu'on est raisonnablement convaincu de sa capacité à résoudre le problème systématiquement), il est important de rédiger le tout sous la forme d'un algorithme, en recherchant certains critères :

— **compréhensible** : Le lecteur ciblé (humain ou machine) doit comprendre chaque terme employé. Il est donc parfois nécessaire de définir ces termes précisément, avant de présenter l'algorithme.

— **non ambigü** : Chaque instruction doit être déterministe, ne laissant place à aucun doute de la part du lecteur. De même, l'ordre d'exécution des instructions doit être bien définie.

Ce n'est qu'une fois l'algorithme formalisé qu'on peut réellement l'étudier, puisqu'on peut alors s'appuyer sur sa structure pour en faire la démonstration ou encore évaluer sa complexité.

6.3 Optimalité

Le problème du Base-ball multicolore relève d'une classe de problème générique qui correspond à chercher un chemin dans un graphe. Une configuration C est une représentation du plateau de jeu avec les pions disposés sur les bases. L'espace des configurations $\mathcal{C}$ est donc l'ensemble de toutes les configurations possibles du jeu. Une action sur une configuration consiste à passer d'une configuration à une autre, dans notre cas c'est choisir, parmi les 4 pions voisins du trou lequel bouger vers le trou [1]. Les actions sont ainsi représentées par un ensemble $\mathcal{A}$ d'arcs reliant des configurations voisines, et la structure formelle sur laquelle on travaille est un graphe orienté sur l'ensemble $\mathcal{C}$ des configurations [2].

Les questions posées se formulent donc :

Étant donné un graphe $\mathcal{G} = (\mathcal{C}, \mathcal{A})$, une configuration initiale C_{init} et une configuration finale C_{finale}

1. Existe-t'il un chemin dans $\mathcal{G}$ de C_{init} à C_{finale} ?
2. Écrire un algorithme qui construit un tel chemin.
3. Écrire un algorithme qui construit un plus court chemin (en nombre d'actions) de C_{init} à C_{finale} .

On se ramène ainsi à un problème classique d'algorithmique de graphe et les solutions algorithmiques sont nombreuses. Il est donc possible de résoudre

1. Dans certains cas les deux pions dans la base voisine sont de même couleur, donc deux actions produisent la même configuration finale

2. En remarquant que les actions sont réversibles on pourrait se restreindre à un graphe non orienté.

le problème à l'aide d'une solution générique (algorithme de [Dijkstra, 1976] ou algorithme A^* par exemple) qui s'appliquerait dans de nombreuses situations similaires : casse-têtes combinatoires, trajectoires de robots, etc. Le coût de l'algorithme sera significativement plus important que pour ceux exposés plus haut [3] , mais on connaîtra le nombre minimal de coups permettant de résoudre le problème.

Conclusion

Le Base-ball multicolore est un exemple d'activité d'informatique débranchée, exploitable à différents degrés de profondeur. À partir d'un même problème, il permet d'aborder de nombreuses notions et travailler des démarches spécifiques à l'algorithmique. Plus généralement, les objectifs pédagogiques de ces activités d'informatique débranchées couvrent un large champ des compétences à acquérir pour une formation à l'informatique. Elles permettent notamment aux élèves, selon leur niveau, de :

- s'approprier un problème, exécuter des cas particuliers, étudier éventuellement un problème plus simple ;
- mettre en œuvre une démarche conduisant à la construction d'un algorithme ;
- apprendre à décomposer un problème en sous-problèmes ;
- dégager des principes généraux, émettre des conjectures ;
- étudier ces algorithmes avec des exemples ou contre-exemples à leur portée ;
- reconnaître une situation relevant d'un algorithme classique ;
- écrire un algorithme sous une forme rigoureuse ;
- estimer le coût d'exécution d'un tel algorithme ;
- établir des preuves partielles ou complètes de correction et de terminaison de ces algorithmes.

Le lecteur souhaitant approfondir le sujet trouvera un panel représentatif d'activités dans [Collectif, 2017], ainsi qu'un catalogue alimenté régulièrement sur [Collectif, 2014].

Références

[Bell *et al.*, 2014] Bell, T., Witten, I. H. et Fellows, M. (2014). *Computer Science Unpplugged*. Publié sur le Web. `https://csunplugged.org`.

[Brackmann *et al.*, 2017] Brackmann, C., Román-González, M., Robles, G., Moreno-León, J., Casali, A. et Barone, D. (2017). Development of computational thinking skills through unplugged activities in primary school. *In Proceedings of The 12th Workshop on Primary and Secondary Computing Education (WiPSCE 2017)*, pages 65–72, Nijmegen, Netherlands.

[Collectif, 2014] Collectif (2014). Pixees – ressources pour les sciences du numérique. `https://pixees.fr/category/support-pedagogique/activite/activite-debranchee`.

3. Pour s'en convaincre, il suffit d'estimer l'ordre de grandeur du nombre de configurations distinctes.

[Collectif, 2017] Collectif (2017). L'informatique débranchée. *Tangente éducation*, 42-43.

[Dijkstra, 1976] Dijkstra, E. W. (1976). *A Discipline of Programming*. Prentice-Hall.

[Pólya, 1945] Pólya, G. (1945). *How to solve it ?* Princeton University Press.

[Quinson, 2014] Quinson, M. (2011-2014). Sciences manuelles du numérique. Publié sur le Web. `http://people.irisa.fr/Martin.Quinson/Mediation/algo1-livret.pdf`.

A Détail des procédures d'échange

Pour des raisons de place, nous présentons les procédures utilisées en section 4.3 avec des bases disposées en ligne dans les figures, mais nous ne nous interdisons pas d'utiliser tous les déplacements disponibles sur le cercle. Il est de toute façon pertinent ici de toujours considérer le chemin le plus court entre les pions (ou trou) que l'on cherche à échanger, ce qui nous ramène à un algorithme « en ligne ». La longueur des chemins considérés est alors d'au plus $\frac{n}{2}$.

A.1 Échange de deux pions voisins

Supposons qu'on cherche à échanger deux pions voisins A et B : dans le cas particulier où le trou est déjà présent sur une des bases contenant nos 2 pions, on constate que 2 déplacements suffisent (voir figure 9). Dans le cas général, il convient alors simplement d'y ajouter un maximum de $2 \times \frac{n}{2}$ déplacements pour « rapprocher le trou » puis le remettre à sa place (si on suit le chemin le plus court). Ainsi, sur la figure 8, on déplace temporairement deux pions, qui sont marqués par un X sur la figure 9 où on effectue l'échange, et enfin on remet les pions X à leur place comme sur la figure 10.

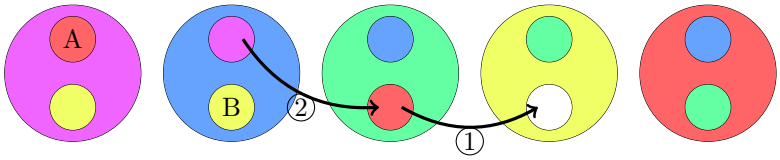

FIGURE 8 – Échange de pions voisins : amener le trou à proximité des pions à échanger

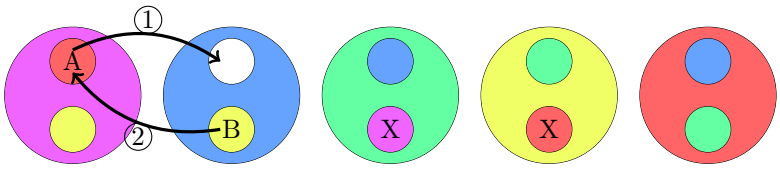

FIGURE 9 – Échange de pions voisins : effectuer l'échange

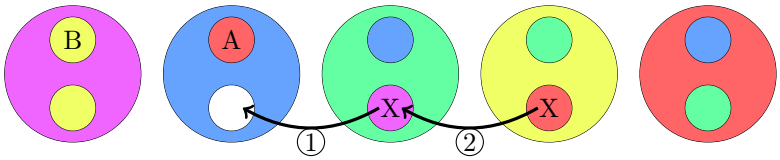

FIGURE 10 – Échange de pions voisins : ramener le trou et les autres pions à leurs places initiales

A.2 Échange d'un pion avec le trou

L'opération qui consiste à placer un pion arbitraire A dans le trou sans modifier le reste est un peu plus complexe que la précédente (ce qui est normal car elle nous permet d'effectuer un déplacement global et non plus local) mais il reste possible de l'effectuer en $\mathcal{O}(n)$ déplacements (on rappelle qu'ici n désigne le nombre de bases, dans une version généralisée du problème) :

— sur tout le chemin entre le pion A à déplacer et le trou, chaque base contient deux pions que l'on nomme X et Y, comme sur la figure 11 ;
— on déplace chaque pion X d'une position en direction du trou ;

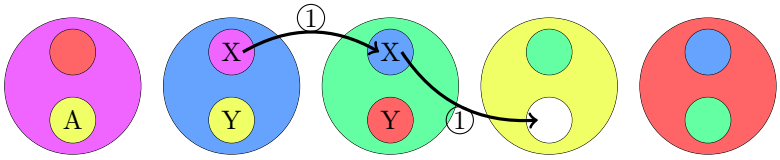

FIGURE 11 – Placer un pion dans le trou : décaler les pions X pour amener le trou à proximité du pion

— on rapproche le pion A qu'on voulait déplacer de sa destination en l'échangeant avec un Y, puis en ramenant un X à sa place (la figure 12 montre 2 itérations de cette procédure) ;
— en répétant cette série de 3 déplacements, le pion arrive au voisinage de sa destination, les X reviennent à leur place initiale, et les Y sont décalés ;

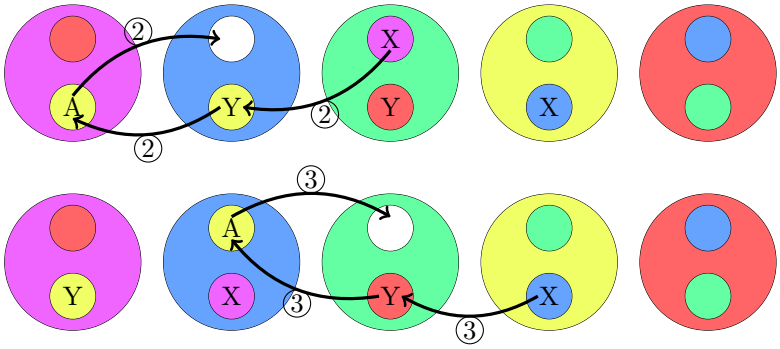

FIGURE 12 – Placer un pion dans le trou : amener ce pion à sa place

— enfin, comme sur la figure 13, on déplace le pion A à l'emplacement voulu, puis on remet les Y à leur place initiale, ce qui a aussi pour effet d'amener le trou à l'emplacement initial du pion A.

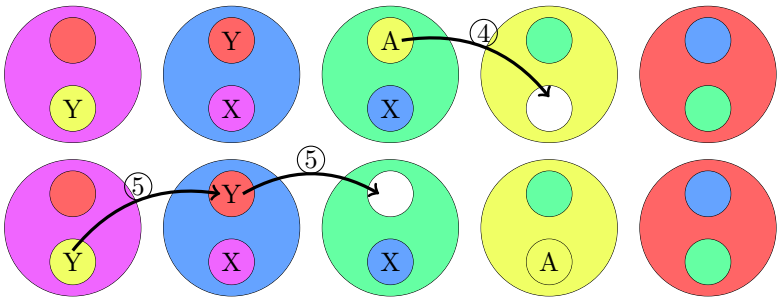

FIGURE 13 – Placer un pion dans le trou : ramener les pions Y à leurs places, et le trou sur la base voulue