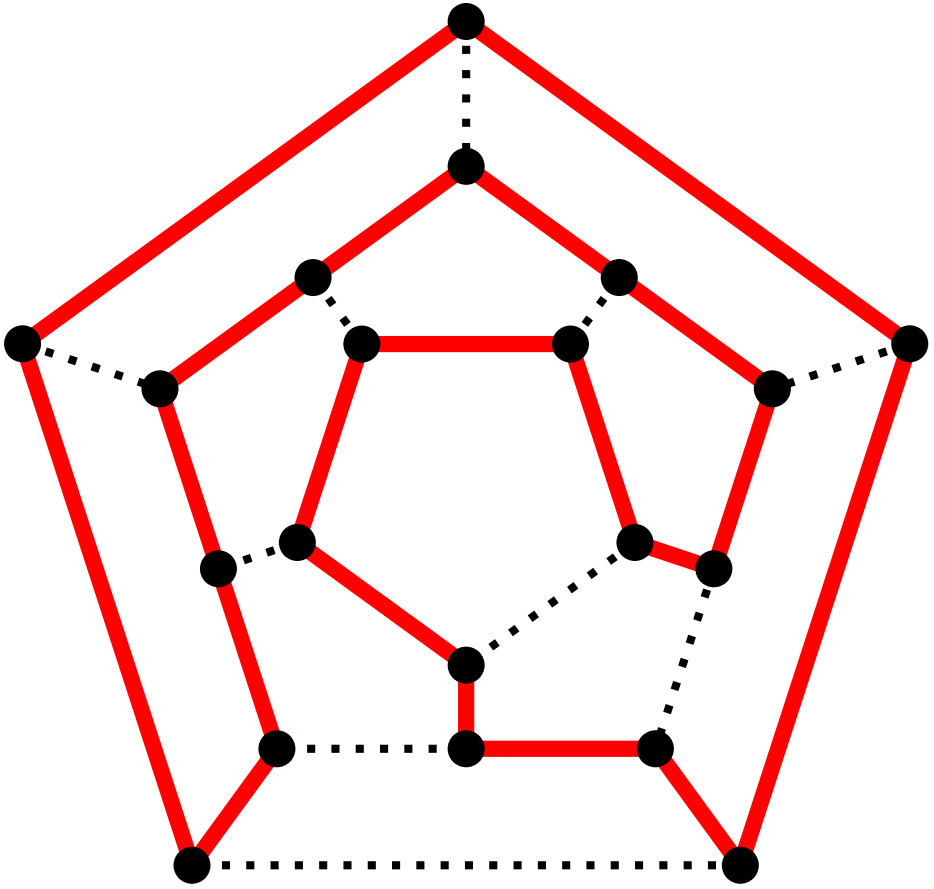
Les algorithmes



CS IRL

Computer Science In Real Life

À propos de ce document

- La page web du projet est ici : `http://www.loria.fr/~quinson/Mediation/CSIRL/`
- Les sources des ressources du projet sont entre autres ici : `http://github.com/jcb/CSIRL`
- Si vous le souhaitez, vous pouvez nous joindre ici : `discussions@listes.nybi.cc`

Crédits image

P1 : Chemin hamiltonien par Ch. Sommer, licence GFDL/CC-BY-SA
`http://en.wikipedia.org/wiki/File:Hamiltonian_path.svg`
P4 : Computer Science Major : CC-BY-NC `http://abstrusegoose.com/206`
P**??** : Electric City par Mathias M, licence CC-BY-SA `http://www.flickr.com/photos/mathias_m/342535332/`

CS IRL ? Qu'est ce que c'est ?

- Des activités présentant les bases de l'informatique, mais sans ordinateur
- Pour chaque activité, un support matériel est proposé pour permettre d'*apprendre avec les mains*
- Les activités sont rangées en séances cohérentes et progressives

🔎 Computer Science In Real Life : Computer Science est la science informatique en anglais, tandis que IRL est l'abréviation utilisée sur internet pour décrire la vraie vie, ce qui n'est pas sur internet.

Les séances existantes dans la série

- Les algorithmes : Qu'est ce qu'un algorithme ? Et une heuristique ? À quoi ça sert ?
- Codes et représentations : Comment les ordinateurs codent et manipulent les données *(à venir)*
- Turzzle : puzzle de programmation sans ordinateur *(à venir)*

Objectif de la séance algorithmique

- Expliquer ce qu'est un algorithme et à quoi ça sert quand on veut utiliser un ordinateur
- Montrer un aspect du travail d'un informaticien, et de celui d'un chercheur en informatique
- La durée envisagée est d'une heure et demi ou deux heures.
- Ce n'est donc pas un cours complet sur l'algorithmique, qui nécessite 25 à 50h au minimum.
  Cours pour aller plus loin (en 48h) : `http://www.loria.fr/~quinson/Teaching/TOP/`

🔎 Si vous êtes l'animateur, vous trouverez des conseils et des astuces dans le coin de l'animateur en page 19.

Matériel nécessaire pour cette séance

- Des clous, dont un coloré
- Des petites planches de tailles différentes
- Des legos : cinq couleurs, avec à chaque fois deux pièces $\mathbf{2} \times \mathbf{2}$ et une pièce $\mathbf{4} \times \mathbf{2}$
- Une planche avec des clous plantés (mais qui dépassent) ; Une cordelette et un marqueur

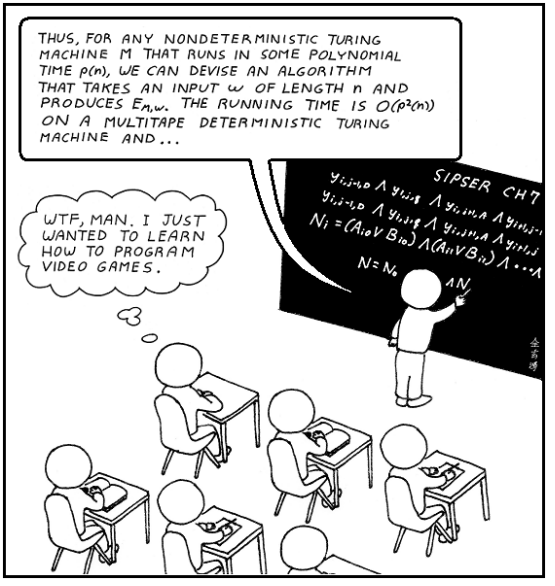
Computer Science IRL – La séance algorithmique

Introduction : les principales caractéristiques d'un ordinateur

- Il est très rapide : il peut calculer de 1 à 1 million en moins d'une seconde
- Il est parfaitement obéissant : il fait tout le temps exactement ce qu'on lui demande
- Il est absolument stupide : il exécute les ordres qu'on lui donne, sans la moindre capacité d'initiative.
  - Par exemple, si on demande à un ordinateur de s'arrêter, il le fait...
  - Autre exemple, quand j'indique à des amis comment venir chez moi, je leur donne des indications comme "troisième à droite" ou "à gauche au 2ieme feu". Si je me trompe dans mes indications ("à gauche" au lieu de "à droite") et que cela les ferait prendre l'autoroute à contre-sens, mes amis vont faire preuve de sens commun et ne pas appliquer la consigne. Les ordinateurs n'ont **aucun** sens commun.
  - Bug (n.m.) : consigne erronée donnée par un humain et appliquée bêtement par une machine.

Le travail d'un informaticien

- Se faire obéir d'un serviteur aussi stupide qu'un tas de fil demande un peu d'organisation
- Pour décomposer suffisamment les tâches à réaliser, il réfléchit à **comment** faire
  (un peu comme un cycliste qui descendrait du vélo pour se regarder pédaler afin d'expliquer ensuite comment faire)
- Pour chaque problème, il faut d'abord définir :
  - la situation initiale : le point de départ du problème
  - les opérations possibles : ce que j'ai le droit de faire pour faire évoluer la situation
  - la situation finale : ce vers quoi je veux tendre, l'état du problème quand je l'ai résolu



Activité : le jeu de Nim

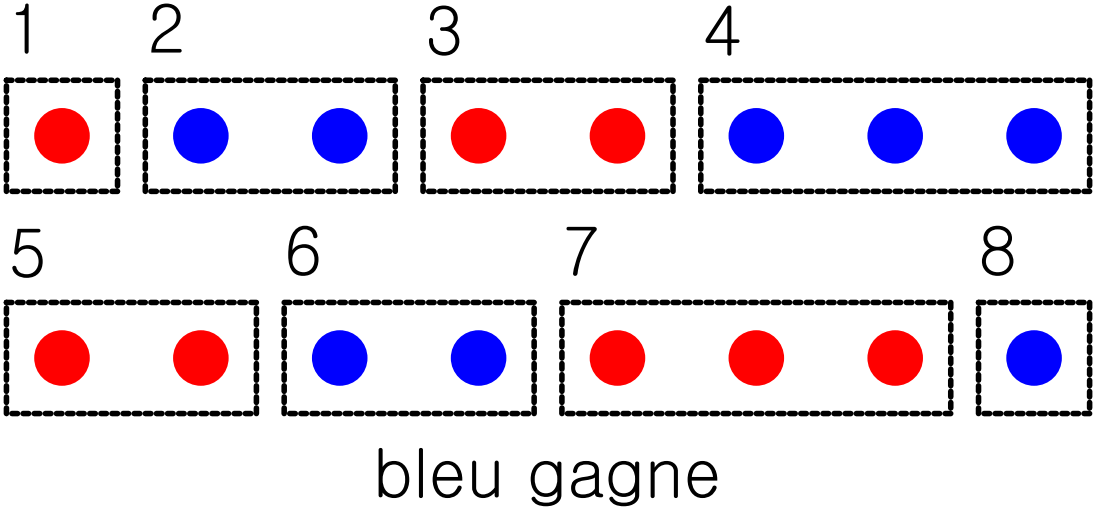
Voici un premier petit jeu simple, pour rentrer dans le sujet.

Matériel

- 16 petits objets (clous, allumettes, boulettes de papier ... peu importe !)

Règle du jeu

- Disposer les 16 objets sur une table
- Les deux joueurs prennent tour à tour 1, 2 ou 3 objets
- Le joueur qui prend le dernier objet à gagné



Ce qu'il faut retenir du jeu de Nim

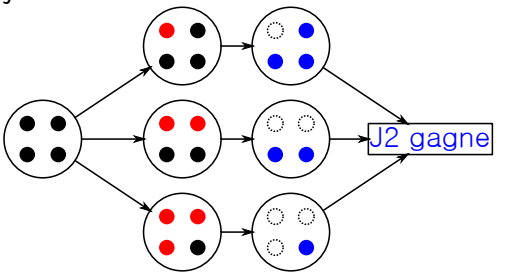
L'intérêt majeur de ce jeu est qu'il est sans suspense (voire, sans intérêt ;)

- Celui qui commence (J1) perd, car il existe un truc pour que J2 gagne à tous les coups
- Stratégie gagnante : Laisser 4, 8, 12 ou 16 objets à l'adversaire (un multiple de 4)

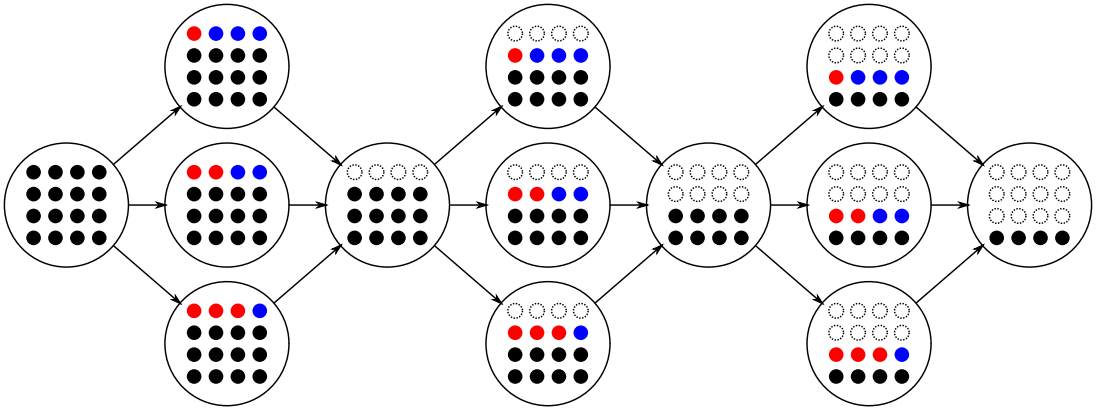
Se convaincre de l'efficacité de la stratégie gagnante

Prenons le dernier tour comme exemple. Il reste 4 objets, et J1 joue.

- Si J1 prend 1 objet, J2 en prend 3 (dont le dernier)
- Si J1 prend 2 objets, J2 en prend 2 (dont le dernier)
- Si J1 prend 3 objets, J2 en prend 1 (le dernier)



Dans ce cas, si J2 sait jouer, J1 perd à tous les coups.
En appliquant la même méthode, J2 peut guider le jeu de manière à passer de 16 objets à 12, puis 8 et enfin 4.
Donc, si J2 sait jouer, J1 a perdu la partie avant même de commencer.



Le rapport avec l'informatique

- Passer de la situation initiale à la situation finale à coup sûr demande d'avoir une *stratégie gagnante*
- C'est un **algorithme** en informatique, une recette de cuisine ou un manuel de montage de meubles
- Pour se faire obéir du tas de fils, l'informaticien cherche l'algorithme pour résoudre le problème, puis il écrit le **programme** (traduction de l'algorithme dans un langage informatique)

pour aller plus loin ...

- Sous quelle condition est-on sûr de gagner si le nombre de objets n'est pas un multiple de 4 ?
- Que faudrait-il changer pour gagner à coup sûr si les joueurs ne peuvent prendre qu'un ou deux objets à la fois ?

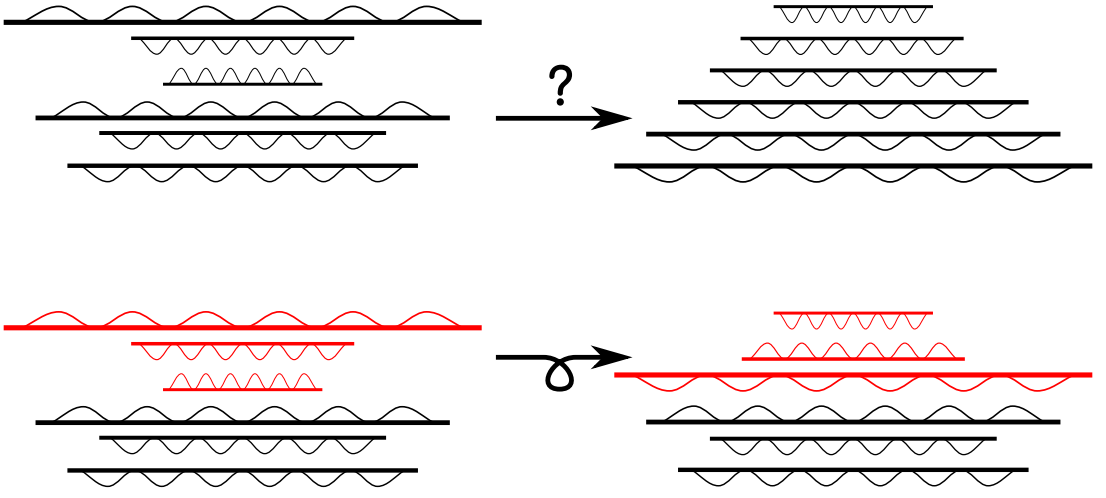
Activité : Le crêpier psycho-rigide

Matériel

- des planchettes en bois de tailles et de couleurs différentes (faces reconnaissables)
- éventuellement une pelle à tarte pour retourner les planchettes

Règle du jeu

- Installation : Faire une pile désordonnée de crêpes.
- Objectif : ranger les crêpes de la plus grande (en dessous) à la plus petite (au dessus), face colorée vers le haut.
- Coup autorisé : prendre une ou plusieurs crêpes sur le haut de la pile, et de les reposer à l'envers.



Ce qu'il faut retenir du crêpier psycho-rigide

Un algorithme

- n'a d'intérêt que si on peut l'expliquer
- doit être suffisamment simple pour pouvoir l'expliquer à une machine
- «**Diviser pour mieux régner**» : on essaie toujours de décomposer un algorithme en tâches simples

L'algorithme que doit suivre le crêpier est :

- ramener la plus grande crêpe en haut de la pile
- retourner pour que la face brûlée soit vers le haut
- retourner la pile de sorte à mettre la plus grande crêpe en bas
- réitérer avec la crêpe de taille inférieure

Le rapport avec l'informatique

- l'informaticien passe son temps à trouver des algorithmes et à les expliquer à la machine
- le principe «**Diviser pour mieux régner**» est fondamental en informatique

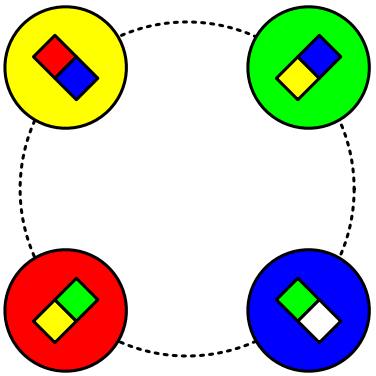
Activité : Base-ball multicolore

Matériel nécessaire

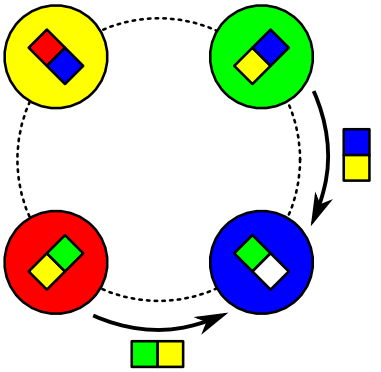
- Plusieurs équipes bien différentiables, chacune composée d'une maison et de deux bonshommes (des legos, des bouts de bois, des cailloux, du fil électrique de différentes couleurs, ou autres)
- 4 équipes au minimum. On peut mettre des équipes supplémentaires pour augmenter la difficulté.

Règles du jeu (exemple à quatre équipes)

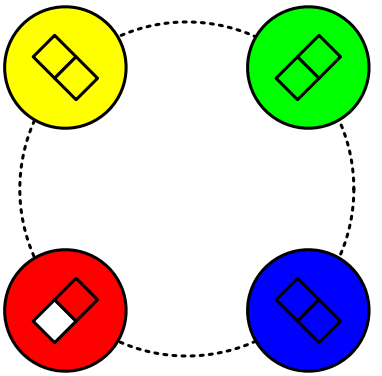
- Installation : disposer 4 maisons autour du terrain et répartir 7 bonshommes au hasard sur les maisons (le bonhomme restant n'est pas utilisé).
- Coup autorisé : déplacer un seul bonhomme à la fois, vers la maison contenant un seul bonhomme, depuis une des deux maisons voisines (interdit de traverser le terrain).
- Objectif : Ramener tous les bonshommes dans la maison de leur couleur.



État initial



Coup autorisé



État final

Objectif de l'activité

Le plus important dans cet exercice n'est pas tant de résoudre le problème que d'expliquer clairement comment on fait. On cherche donc l'algorithme permettant de résoudre le problème.

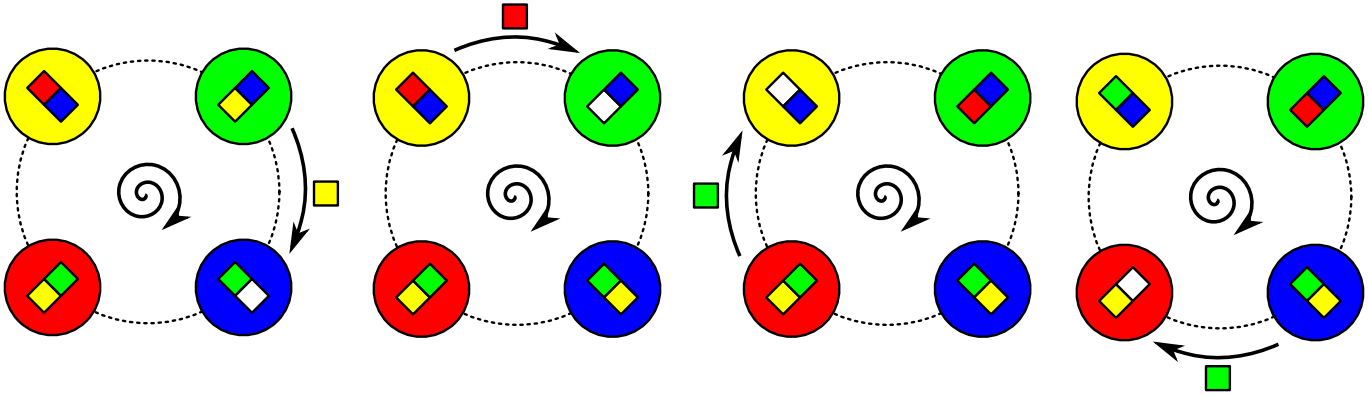
Un premier algorithme pour le base-ball multicolore

En suivant les règles du jeu, on observe que quelque soit la disposition des bonshommes, il existe toujours 4 coups possibles : déplacer vers la case vide un des 4 bonshommes présents dans les deux maisons voisines. Notre algorithme sera donc une méthode permettant de choisir à chaque étape quel coup jouer parmis les 4 possibles.

L'algorithme

- On ne s'autorise à tourner que dans un seul sens. Ainsi, le nombre de coups possibles descends de 4 à 2 (car 2 bonshommes tourneraient à l'envers).
- Parmis les 2 coups restants, on déplace le bonhomme qui a la plus grande distance à parcourir avant d'arriver à sa maison (Si la distance est la même, c'est que les deux bonhommes ont la même couleur - les deux coups sont donc équivalents).

Exemple d'exécution



Ici, nous n'avons représenté que les 4 premières étapes. mais l'algorithme arrive à la solution en 15 étapes.

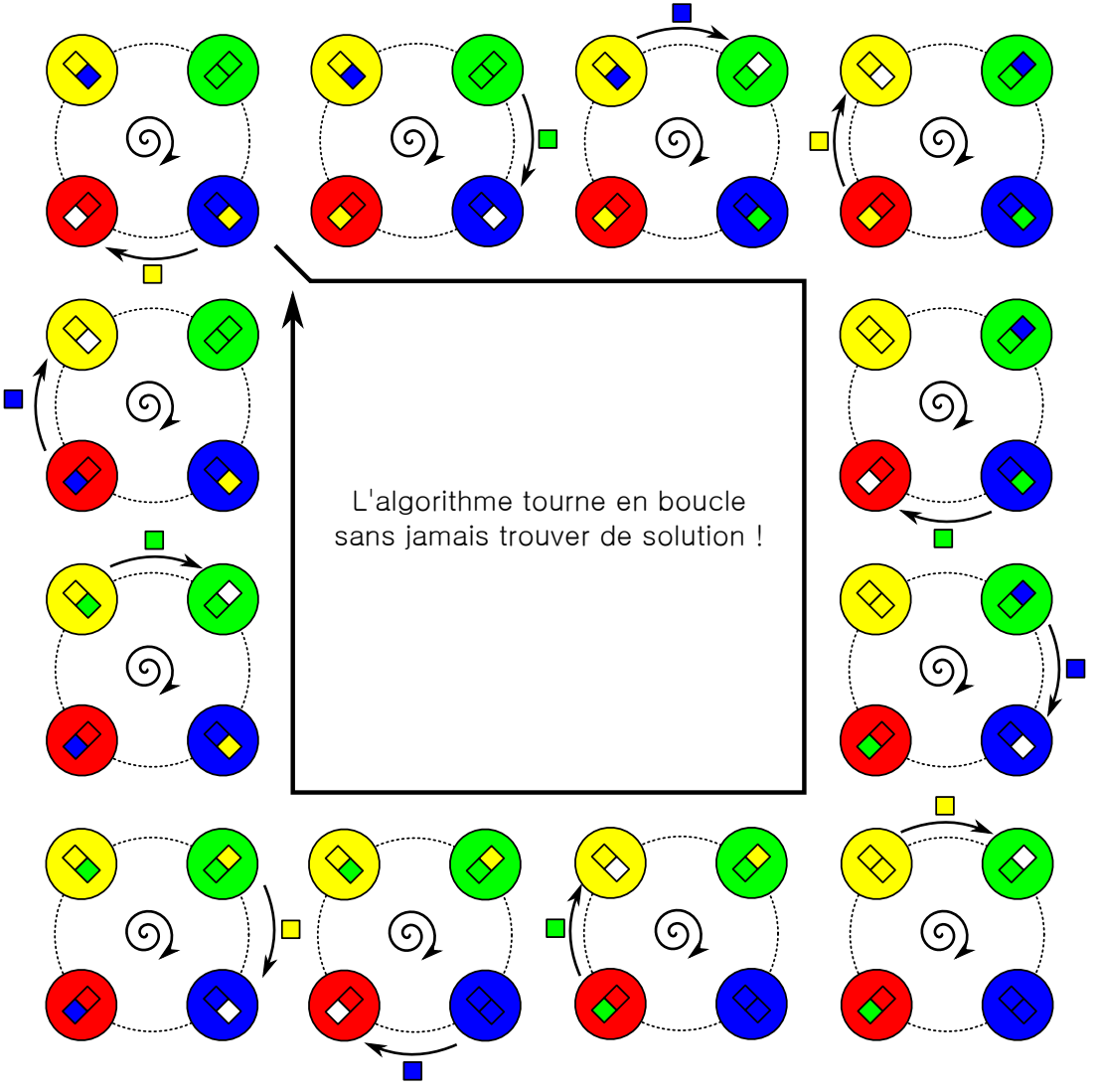
Étude du premier algorithme pour le base-ball multicolore

Cet algorithme semble attirant

- Il est très simple : on pourrait l'expliquer à un ordinateur
- Il est relativement rapide : 15 coups pour 7 bonshommes, ce n'est pas si mal
- Seul problème : cet algorithme est faux : dans certains cas, il ne termine jamais...

Exemple d'exécution incorrecte

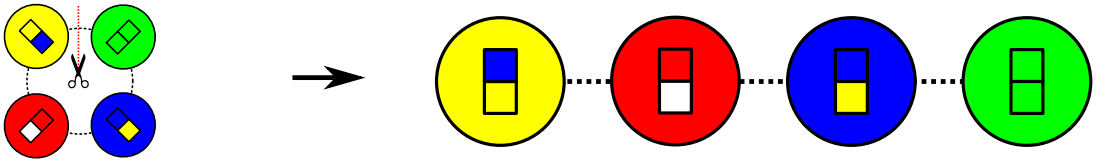
Il suffit de partir d'une situation gagnée et d'inverser deux bonshommes pour mettre notre algorithme en échec.



Un autre algorithme pour le base-ball multicolore

Apprendre de ses échecs : notre algorithme boucle parfois à l'infini

- Pour réparer cela, le plus simple est de s'interdire de boucler, en coupant le cercle.
- Pour ne pas se tromper, le plus simple est de placer les maisons en ligne.

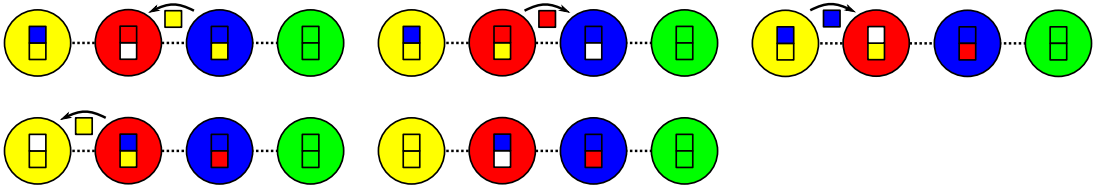


Apprendre de ses réussites : le crépier

- On a cherché à réduire la taille du problème à peu à peu
  (il y a 7 crèpes à trier. La plus grande va définitivement à sa place ; il reste 6 crèpes à trier)
- On s'est fixé des objectifs intermédiaires, qui décomposent le problème en étapes que je sais faire
  (mettre la plus grande en haut pour parvenir à la mettre en bas)

Nouvel algorithme

- On s'occupe d'abord des bonshommes de la première maison, et on n'y touche plus ensuite.
- On répète pour la deuxième maison, et ainsi de suite pour toutes les autres.
- Pour ammener les bonhommes dans leur maison, on déplace tous ceux qui gènent.
- Pour déplacer ceux qui gènent, on déplace le trou pour leur faire de la place.



- On peut maintenant oublier les bonshommes de la première maison, qui sont à leur place définitive.
- On recommence de la même manière avec la deuxième maison, et ainsi de suite ...

Ce qu'il faut retenir du base-ball multicolore : corrections d'algorithmes

Cet algorithme n'est pas tellement plus complexe ou plus long que le précédent, mais il est correct, lui.

Comment être sûr de la correction de cet algorithme ?

- On pourrait tester tous les cas. Ici, il n'y a pas de limite au nombre de maisons - il est donc impossible de vérifier tous les cas, tout comme il est impossible de compter jusqu'à l'infini. Cependant, on peut se contenter d'une preuve partielle en se limitant à tester tous les cas succeptibles d'être rencontrés - par exemple jusqu'à 20 ou 50 maisons.
- On pourrait écrire une preuve mathématique. Ce n'est pas trivial, mais les chercheurs en informatique en ont écrite des plus difficiles.
- Cela ressemble vraiment à un algorithme classique (même si cela ne prouve rien, au fond).

Qu'est ce qu'un algorithme classique ?

- Les informaticiens apprennent par cœur des algorithmes (abstraits) à l'école.
- Face à un problème nouveau, on cherche à se raccrocher à des problèmes connus.
  - On se raccroche en trouvant des analogies ou en décomposant en plusieurs problèmes connus.
  - Par exemple, quand des collègues informaticiens jouent au crêpier, ils demandent avant tout si c'est "une tour de Hanoï".
- Ici, notre algorithme est proche d'un "tri à bulle", autre algorithme bien connu. Mais cette ressemblance ne suffit pas à prouver la correction de notre algorithme. Pour la prouver, on pourrait démontrer que notre algorithme est un cas particulier du tri à bulle.

Les algorithmes de tri sont ultra classiques en informatique

- Ils sont assez simple pour expliquer les grandes lignes aux élèves
  (comme «diviser pour régner» et autres grandes idées similaires – récursivité, algorithmes gloutons, . . .)
- Les ordinateurs trient très souvent des données, car beaucoup de problèmes sont plus simples après
  (trouver un livre donné est plus simple dans une bibliothèque rangée, par exemple)
- Les musiciens font leurs gammes, les informaticiens débutants apprennent leurs algorithmes

Que font les chercheurs en informatique ?

- Certains d'entre eux améliorent les algorithmes connus, ou en inventent de nouveaux
- Il faut également démontrer la correction de ces algorithmes
- Quand plusieurs algorithmes existent, on étudie leurs performances respectives
- (d'autres chercheurs améliorent matériel et logiciel, établissent des modèles, etc)

Ce qu'il faut retenir du base-ball multicolore : performance d'algorithmes

Comment comparer la performance des algorithmes ?

- Simplement en comptant les étapes. Par exemple sur le crépier, placer la grande crêpe prend au pire 3 coups - et c'est pareil pour les crêpes suivantes. Donc, dans le pire des cas notre algorithme prendra $\mathbf{3 \times n}$ coups pour trier la pile.
- La performance de mon algo dépend beaucoup de la situation initiale :
  - Si c'est déjà trié, c'est de la chance, je n'ai rien à faire (meilleur cas).
  - Si j'ai vraiment pas de bol, je dois faire les 3 étapes pour chaque crêpe (pire cas).
  - En pratique, j'ai souvent une situation initiale intermédiaire (cas moyen).

  Il faut bien comprendre que ceci ne dépend pas vraiment de l'algorithme, mais plutôt de la situation initiale. Le pire cas n'est pas un bug de l'algorithme, mais une situation initiale qui n'aide pas vraiment notre façon de faire (pour estimer la performance d'un cas moyen, il faut utiliser des probabilités).
- En pratique, une estimation de la performance est suffisante. Savoir qu'un algorithme nécessite environ $\mathbf{n^2}$ étapes suffit, inutile de préciser que c'est $\mathbf{n^2 + 4}$ ou $\mathbf{n^2 - 2}$, ou même $\mathbf{5 \times n^2}$ - pour des grandes valeurs de $\mathbf{n}$ c'est sensiblement la même chose... On note cette estimation de la complexité $\mathbf{O(n^2)}$.

À la recherche du meilleur algorithme possible

- On arrive parfois à montrer qu'on a le meilleur algorithme possible. Par exemple on ne peut pas trier les éléments en moins de $\mathbf{n}$ étapes, car on doit forcément tous les considérer.
- On peut aussi prouver qu'un tri comparatif ne peut pas se faire en moins de $\mathbf{n \times log(n)}$ étapes, car il n'accumule pas assez d'information pour choisir la bonne permutation en moins d'étapes.
- Mais la plupart du temps, on ne sait pas prouver que l'algorithme connu est le meilleur possible. C'est alors le meilleur *connu*, sans être forcément le meilleur *possible*.

À la recherche de problèmes difficiles

- On peut classifier les problèmes en fonction de la performance des algorithmes les résolvant.
  (cela permet de se forger un sens commun de ce qui est faisable avec un ordinateur et éviter les problèmes si difficiles qu'ils sont quasi impossibles)
- Il existe énormément de problèmes relativement simples pour lesquels personne ne connaît de bon algorithme, sans que personne n'arrive non plus à démontrer qu'un tel algorithme n'existe pas.
- L'activité suivante sera l'occasion d'explorer un peu cette classification des problèmes très durs.

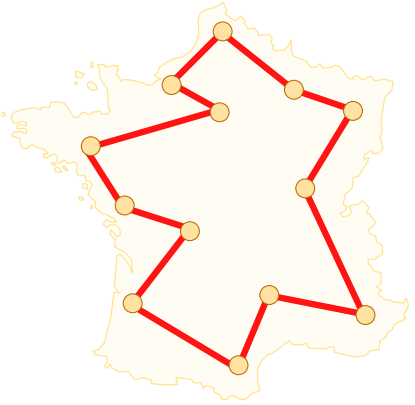
Activité : le plus court chemin

Matériel nécessaire

- Une planche avec des trous au hasard,
- autant de longs clous que de trous,
- une ficelle suffisamment longue et qui ne soit pas élastique,
- un marqueur.

Règles du jeu

- Situation initiale : les clous sont mis dans les trous, leurs têtes dépassent de la planche, et un bout de la ficelle est attachée à un clou.
- Comment jouer : faire passer la ficelle une fois et une seule par tous les clous de la planche, puis revenir au point de départ.
- Objectif : obtenir le chemin le plus court possible. À chaque fois qu'un record est battu, on fait une marque sur la ficelle pour le mémoriser.



Objectif de l'activité

- On peut construire un très grand nombre de chemins différents (pour $\mathbf{10}$ clous, $\mathbf{10! = 10 \cdot 9 \cdot 8 \cdot \ldots \cdot 2 = 3628800}$), et il est très difficile de trouver le meilleur chemin à coup sûr.
- A la place, on va chercher des méthodes (algorithmes) pour construire des chemins courts, et comparer leurs résultats.

Ce qu'il faut retenir du plus court chemin

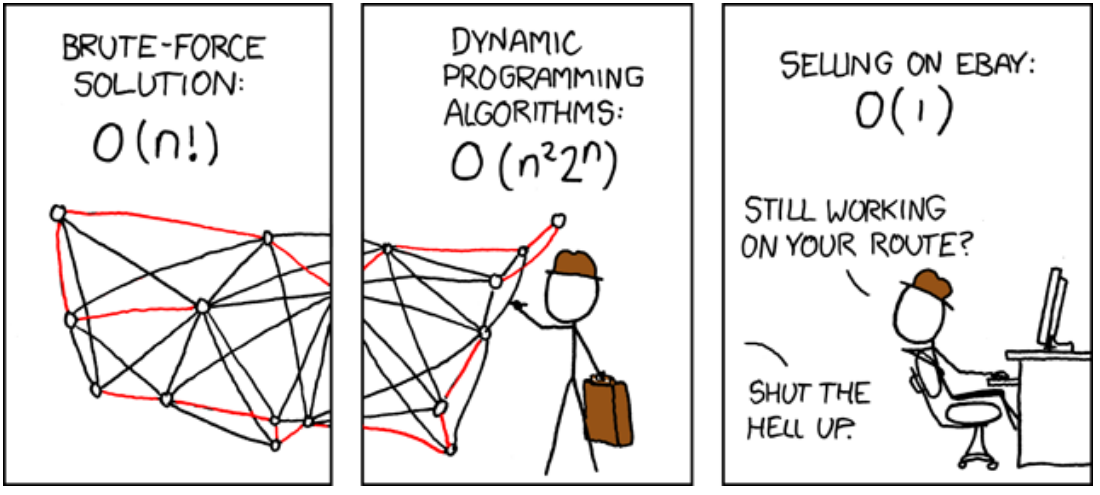
- Certains problèmes (dont le problème du plus court chemin) sont trop compliqués pour trouver la solution optimale en un temps raisonnable. Dans de telles situations, on préfère souvent trouver une solution approchée très rapidement, plutôt que de chercher très longtemps la solution optimale. Les méthodes utilisée pour trouver rapidement des solutions approchées sont des heuristique.
- Le problème du chemin le plus court parait bête mais il fait très difficile, et a de très nombreuses applications dans la vie de tous les jours (comment minimiser la tournée du facteur, la longueur des pistes d'un circuit imprimé, les déplacements d'un bras robotique ...).

Trouver la solution optimale

- Dans le cas du chemin le plus court, l'approche naïve consiste à calculer la longueur de tous les chemins et comparer les résultats pour trouver le plus court. La complexité d'une telle approche s'écrit **O(n!)**. Il existe cependant des algorithmes plus efficaces - par exemple, l'algorithme de Held-Karp a une complexité de $\mathbf{O(n^2 2^n)}$. Voici une petite comparaison de l'augmentation de la quantité de calculs nécessaires à mesure que **n** augmente :

nombre de sommets	5	10	15	20
méthode naïve **O(n!)**	120	3628800	1307674368000	2432902008176640000
Held-Karp $\mathbf{O(n^2 2^n)}$	800	102400	7372800	419430400

- Ce tableau nous montre qu'avec la méthode naïve, en testant un milliard de chemins par seconde, il faudrait plus de **77 ans** pour trouver le chemin le plus court entre 20 sommets ! En comparaison, l'algorithme Held-Karp met moins d'**une demi seconde** pour trouver le même résultat !



Ce qu'il faut retenir du plus court chemin

Trouver une solution approchée

Le coin de l'animateur

Trucs et astuces pour s'assurer que le message passe bien

Remarques générales

- Il faut vous approprier les activités. N'hésitez pas à ne pas suivre les consignes à la lettre.
  Ces activités sont des bases de discussion avec les participants, il n'y a pas d'évaluation à la fin.
- Une question récurrente des participants est de savoir ce que l'animateur fait, en recherche.
  Pensez à préparer une présentation compréhensible de vos recherches, avec le domaine général, ses difficultés et applications et quelques mots de vos préoccupations propres.
  Exemple : Le domaine de mon travail est le parallélisme : est ce que ranger sa chambre va plus vite à 2 ou 3 ? oui. à 3000 ? Non, on perd du temps à se coordonner. Et pourtant, la météo de demain est calculée en utilisant plusieurs milliers d'ordis en même temps, ce qui est difficile. Dans ce domaine, mon travail à moi est d'établir des instruments scientifiques (simulateurs ou parcs de machines), comparables aux télescopes ou microscopes des physiciens, et qui servent d'outils aux scientifiques du domaine.

À propos du mot d'introduction

- On peut faire cette présentation soit au début, soit juste après l'activité sur le jeu de Nim.
- Commencer directement par un petit jeu permet d'éviter que les participants ne décrochent avant même qu'on ne commence.

À propos du jeu de Nim

- L'objectif de cette activité est simplement d'introduire la notion d'algorithme
- On propose le jeu avec le participant, mais sans dire trop vite qu'on a un truc. S'il y a plusieurs participants, on jouera avec plusieurs personnes, pour laisser sa chance à chacun. On peut faire une sorte de petit tournois.
- Il faut bien sûr laisser commencer le participant pour gagner à coup sûr. S'il insiste pour ne pas commencer, on peut le faire (et rattraper la stratégie gagnante à la première erreur du participant)
- On n'introduit l'existence du truc pour gagner que plus tard, quand on gagne à plate couture
- Si on perd, c'est à dire si on n'a pas réussi à appliquer la stratégie gagnante, il faut proposer un match en 3 (ou en 5 en cas de coup dur ;)
- On peut amener le participant à découvrir la stratégie gagnante en groupant les clous par paquets de 4 au lieu de la disposition pyramidale.
- Si l'un des participants connaît déjà la stratégie gagnante du jeu, il peut remplacer l'animateur dans une partie avec d'autres participants

Le coin de l'animateur

Trucs et astuces pour s'assurer que le message passe bien

À propos du jeu du crêpier psycho-rigide

- L'objectif de cette activité est de trouver un algorithme et de le faire verbaliser par les participants
- On propose au participant de d'abord tenter de le résoudre intuitivement, sans réfléchir
- Si le participant bloque, on peut lui donner un conseil : « Une bonne première étape est de se débrouiller pour mettre la grande en bas »
- Si le participant bloque toujours, on peut lui donner un second conseil : « où est-ce que la grande devrait être pour pouvoir la mettre en bas ? »puis le guider pour l'étape suivante.
- On essaie ensuite de faire expliquer l'algorithme par le participant. On gagne à ce que ce soit le participant et non l'animateur qui explique aux autres, avec ses propres mots.

Le coin de l'animateur

Trucs et astuces pour s'assurer que le message passe bien

À propos du base-ball multicolore

- L'objectif de cette activité est d'introduire les notions de correction et performances d'algorithmes
- Il faut laisser les participants chercher un peu en les faisant verbaliser
- S'ils sont sur le point de trouver l'algo juste, on introduit très vite l'algo faux pour préserver un enchaînement logique : "oui, ok, mais je vais vous montrer une façon de faire rigolote"
- Quand l'algo juste est établi, et avant de parler de performance, on peut appliquer sur une variante :
  - Chaque participant prend une couleur (une maison pour commencer) et ses pieds
  - Chaque participant prend une couleur (une maison pour commencer) et ses pieds
  - Chaque participant (sauf 1) prend un bonhomme dans chaque main
  - À chaque étape, celui qui a une main libre prend un bonhomme dans la main d'un voisin
  - (attention, c'est fastidieux à 8 ou 9 couleurs, il vaut mieux faire deux rondes car l'algo semble $\mathbf{O(n^2)}$)
- Expérimentalement, l'algo qui tourne converge très souvent vers la solution à 5 maisons, mais converge souvent vers la boucle infinie quand il y a plus de couleurs. Ne tentez pas le diable ;)
- Dans la disposition linéaire, il est plus simple de mettre la couleur avec un seul bonhomme à une extrémité, et commencer par remplir la maison de l'autre extrémité. Sinon, on se retrouve avec une maison remplie de un seul au milieu, et il faut comprendre que la solution passe par le stockage temporaire d'un pion de la maison d'à coté sur le trou.
- Le discours sur le $\mathbf{O(n)}$ est volontairement approximatif. On veut faire sentir les choses ; faire un vrai cours prend une douzaine d'heures (cf. `http://www.loria.fr/~quinson/Teaching/TOP/`).
- Il serait intéressant de prouver effectivement la correction de l'algorithme linéaire, ainsi que de quantifier la probabilité de fonctionner de l'algo qui tourne en fonction du nombre de maisons
- Au passage, le crépier ne ressemble pas du tout aux tours de Hanoï : l'histoire ressemble un peu, mais la résolution est très différente (il y a $\mathbf{2^n - 1}$ étapes à Hanoï et $\mathbf{3 \times n}$ au crépier...)