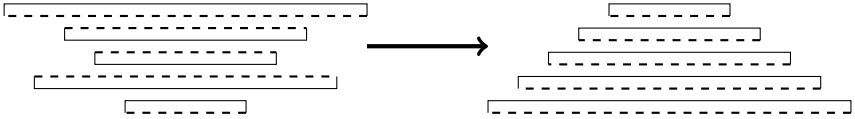
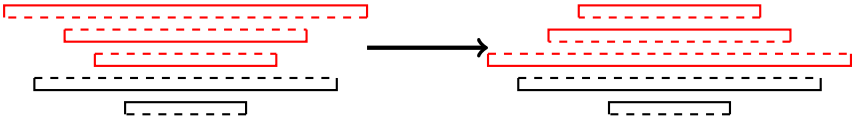
Le crêpier psycho-rigide

A la fin de sa journée, un crêpier dispose d'une pile de crêpes désordonnée. Le crêpier étant un peu psycho-rigide, il décide de ranger sa pile de crêpes, de la plus grande (en bas) à la plus petite (en haut), avec le coté brûlé caché.



Pour cette tâche, le crêpier peut faire une seule action : glisser sa spatule entre deux crêpes et retourner le haut de la pile. Comment doit-il procéder pour trier toute la pile ?



Matériel

- Des planchettes en bois de tailles et de couleurs différentes (faces reconnaissables)
- Une pelle à tarte pour retourner les planchettes (optionnelle)

Description d'un algorithme

L'algorithme permettant de résoudre le problème du crêpier est le suivant :

1. amener la plus grande crêpe en haut de la pile
2. mettre la face brûlée vers le haut
3. retourner toute la pile - la crêpe est rangée
4. recommencer en ignorant les crêpes rangées

Cet algorithme assez simple nous apprend deux choses. Premièrement, un algorithme n'a d'intérêt que si on peut l'expliquer - pire encore, que si on peut l'*expliquer à un ordinateur*. Il doit donc être **écrit sans ambiguïté**. Deuxièmement, un algorithme décompose le problème en une série de tâches simples. On appelle ce principe **« Diviser pour mieux régner »**.

Ce que nous venons de décrire est le cœur de métier des informaticiens : analyser un problème, le subdiviser en problèmes plus simples, formaliser le tout sous la forme d'un algorithme, et traduire l'algorithme dans un langage compréhensible par l'ordinateur.

Performance d'un algorithme

Le premier objectif de l'écriture d'un algorithme est qu'il résolve le problème. Le second objectif est qu'il le résolve le plus vite possible.

Évaluer la **performance d'un algorithme** nous permet de nous faire une idée du temps nécessaire à la résolution d'un problème de grande taille. Par exemple, combien de temps faudra-t-il à notre algorithme pour trier une pile d'un million de crêpes ?

De plus, s'il existe plusieurs algorithmes résolvant le même problème, l'évaluation de la performance nous donne un critère objectif pour savoir lequel est le plus efficace.

Évaluer la performance d'un algorithme

Pour évaluer la performance, on compte le nombre de « coups » nécessaires pour résoudre le problème dans le cas général. Pour le problème du crêpier :

- pour ranger une crêpe, il faut entre 0 coup (la crêpe est déjà rangée) et 3 coups (amener en haut, retourner, amener à sa place) ;
- pour n crêpes (cas général), il faut entre 0 coup (meilleure situation) et $3 \times n$ coups (pire situation). La performance de l'algorithme dépend donc beaucoup de l'état initial, mais on s'intéresse surtout aux cas intermédiaires, qui sont les plus probables.

Ici, n est une variable qui exprime la taille du problème. La performance d'un algorithme est notée comme une fonction de la taille du problème nommée O. Pour le crêpier, on peut donc écrire $O(3 \times n)$.

Le temps d'exécution variant selon l'état initial, la performance exprime l'ordre de grandeur du temps d'exécution. Pour des grandes valeurs de n, il n'est pas très utile de faire la distinction entre $O(n)$, $O(n+4)$ ou encore $O(3 \times n)$ — en particulier quand on compare avec un autre algorithme dont la performance est $O(n^2)$.

On simplifie donc en retirant les constantes pour ne garder que les termes importants. Pour le crêpier, on notera donc la performance de notre algorithme $O(n)$; on dit alors que la performance de l'algorithme est *linéaire*, car le temps de calcul croît linéairement avec n. Un algorithme avec une performance $O(n^2)$ est dit *quadratique*, tandis qu'un algorithme $O(2^n)$ est dit *exponentiel*.

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 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 $n \times log(n)$ étapes, car il n'accumule pas assez d'informations 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*.

Le coin de l'animateur

L'objectif de cette activité est de trouver un algorithme, de le faire verbaliser par les participants et d'en mesurer la performance.

- Expliquez les règles et demandez aux participants de tenter de résoudre le problème ;
- s'ils bloquent, conseillez-les. Par exemple : « essaye d'abord de mettre la grande crêpe en bas », ou encore « où doit se trouver la grande crêpe pour pouvoir l'amener en bas ? »
- Quand les participants ont trouvé l'algorithme, demandez-leur de l'expliquer.
- Demandez ensuite de calculer le nombre de coups nécessaires pour ranger la pile de crêpes. Le nombre de coups dépendant de l'état initial, faites-les généraliser en trouvant le nombre de coups maximal pour ranger une crêpe, puis n crêpes.
- Le discours sur le $O(n)$ 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/`).