Société de Calcul Mathématique SA
Outils d'aide à la décision

Algorithmes et programmes informatiques :
où agir pour améliorer l'efficacité et réduire les coûts ?

par Bernard Beauzamy

2007

Le grand public, lorsqu'il voit un processus où les décisions sont prises par un ordinateur - que ce soient les commandes passées par un supermarché lorsque les stocks baissent ou que ce soit l'envol d'Ariane – a tendance à dire : "c'est de l'informatique". C'est une idée fausse, parce qu'elle mélange deux choses très différentes : la conception d'un algorithme et la programmation de celui-ci, et cette idée fausse coûte cher, parce qu'elle empêche de voir où sont les véritables possibilités pour améliorer l'efficacité et réduire les coûts.

Un algorithme est la description d'une méthode pour résoudre un problème, pour effectuer une réalisation, pour parvenir à un but. Par exemple, pour un conducteur de bus à qui l'on a assigné la tâche "prendre une trentaine d'enfants chez eux et les conduire au musée", un algorithme pourrait être :

– ranger par ordre alphabétique la liste des noms des enfants ;
– aller chercher le premier, puis le second, ainsi de suite jusqu'au dernier ;
– lorsque tous sont dans le bus, se diriger vers le musée.

Un algorithme n'est pas, normalement, la seule solution d'un problème particulier ; il représente au contraire une méthode générale, que l'on doit pouvoir mettre en oeuvre sans changement dans un ensemble de cas particuliers. Dans l'exemple précédent, il ne s'agit pas de conduire Jacques, Jean et Paul au musée, mais d'élaborer des règles générales, qui seront encore valables le mois suivant, où l'on aura peut-être vingt enfants à conduire au lieu de trente.

Lorsque l'algorithme est conçu, on peut le programmer. Dans notre exemple, il est facile de concevoir la réalisation suivante : l'école dispose d'une base de données de tous ses élèves ; ceux qui doivent être emmenés au musée ce matin-là sont sélectionnés selon un certain critère (ils sont dans telle classe, ou ont eu de bonnes notes en histoire) : c'est un tri informatique. Ensuite, leurs fiches sont rangées par ordre alphabétique (ordonnancement). On connaît l'adresse de chacun, donc la distance entre deux élèves consécutifs. On peut alors réaliser un petit programme informatique qui sera la feuille de route du chauffeur : à 7 h il se présente chez Adam, puis à 7 h 03 il sera chez Barnabé (1 mn pour charger Adam, 2 mn pour aller chez Barnabé qui est à 2 km, etc.), etc., et l'arrivée au musée pourra être prévue, disons pour 7 h 52.

On voit clairement sur cet exemple les deux niveaux : le niveau conceptuel (imaginer l'algorithme) et le niveau programmation (établir la feuille de route). Et quand le grand public résume en disant "c'est de l'informatique", parce qu'il ne voit que le résultat, à savoir la feuille de route (qui apparaît généralement sous la forme d'un listing : feuille de papier avec des trous sur le côté), il se trompe, et plus précisément, il passe à côté de l'essentiel. Cette idée fausse est confortée par l'aspect économique : les décideurs ont tendance à considérer en premier ce qui coûte le plus cher, et dans le processus que nous avons décrit, la partie "concevoir l'algorithme" coûte infiniment moins que la partie de programmation. C'est complètement évident sur notre exemple, parce que l'algorithme consiste seulement en ceci : ranger les noms par ordre alphabétique, et mettre les adresses en regard, ce que n'importe qui –sauf à être totalement illettré– peut faire en 5 mn ; tandis que la programmation sera assez complexe : il faut accéder à des bases de données, y faire des opérations, des calculs de distances et d'horaires, et imprimer les résultats sous forme utilisable par le chauffeur. Disons que la programmation peut prendre une heure ou deux.

On me dira que cet exemple est trop simple et qu'il n'est pas significatif. Il est effectivement trop simple, mais il est significatif : la partie conceptuelle, qui relève des mathématiques, est toujours moins coûteuse que la partie programmation, qui relève de l'informatique, et ce dans un facteur de dix en général.

Ceci semblera étrange à plus d'un lecteur : on dit toujours que c'est la matière grise qui coûte cher. Comment se peut-il que la partie "intelligence" coûte moins cher que la partie "développement" ? Après tout, pour la première, il faut résoudre un problème, alors que pour la seconde il suffit d'écrire un programme, ce que n'importe quel programmeur peut faire si on lui donne les instructions convenables.

Le paradoxe s'éclaire si l'on considère ceci : l'algorithmie relève des mathématiques, qui est une science vieille de six mille ans ; dans bien des cas le problème considéré a déjà été étudié sous une forme ou sous une autre, et l'expert sait utiliser les "briques" antérieures. De plus, une méthode mathématique a un caractère absolu : si les hypothèses sont bien celles que l'on a prévues, la conclusion sera infailliblement celle que l'on attend.

La programmation, au contraire, n'est pas une science mais, comme dit Donald E. Knuth "un art". Elle se fait lentement, dépend énormément du contexte (quel type de machine ? Quel système d'exploitation ? Quel langage de programmation ?). De plus, il n'existe aucun moyen d'être sûr que le logiciel que l'on écrit fera exactement ce que l'on attend de lui, et cela seulement : dans des programmes qui font des centaines de milliers de lignes, voire des millions, comment être sûr que tel cas n'a pas été oublié ? Que telle instruction ne causera pas un conflit avec telle autre ? C'est typiquement la situation rencontrée par Ariane V : une "brique" logicielle, datant d'Ariane IV, avait semble-t-il été oubliée dans le programme de guidage, ou plus exactement, on avait oublié d'évaluer les conséquences qu'elle pourrait avoir dans une situation spécifique nouvelle.

Cette préoccupation, appelée "validation des logiciels" (vérifier qu'ils font bien ce qu'ils devraient faire), est un souci majeur, pour lequel on ne dispose d'aucune méthode générale : il faut tester chaque logiciel, en espérant que les tests seront assez nombreux, et corriger chaque

erreur que l'on décèle, en espérant qu'il n'y en aura pas d'autres : voilà pourquoi c'est long et pourquoi l'ensemble coûte cher.

Pourtant, en dépit de cette dissymétrie des budgets respectifs, c'est la partie conceptuelle, la partie algorithmique, qui détermine le coût et l'efficacité de l'ensemble du programme !

Reprenons l'exemple de notre bus, et imaginons que nous ayons une contrainte horaire : pour des raisons impératives, tous les enfants doivent être au musée à 7 h 45, alors que notre méthode précédente ne les y mettait qu'à 7 h 52. Comme le chauffeur ne prend son service qu'à 7 h, quel est l'unique recours du directeur de la Compagnie de bus ? Acheter un bus plus rapide ! Quand on a fixé l'itinéraire et les temps de départ et d'arrivée, l'unique paramètre sur lequel on puisse jouer est en effet la vitesse. Mais un nouveau bus, surtout un bus plus rapide, plus performant, coûte très cher !

Là encore, cet exemple est tout fait significatif : si la méthode de travail est fixée, l'unique façon d'améliorer les performances est de remplacer le matériel par du matériel plus sophistiqué, qui coûte dix à cent fois plus cher, à lui seul, que l'algorithmie et la programmation réunies ! Il faudra, suivant les cas, des moteurs plus puissants, des caméras plus sensibles, des asservissements plus élaborés, des circuits plus rapides, bref, il faudra payer fort cher, sous forme de matériel sophistiqué, l'imprévoyance dont on a fait preuve au début en adoptant un algorithme simpliste.

Car, simpliste, notre algorithme de départ l'était : ranger par ordre alphabétique est peut-être la première méthode qui vient à l'esprit, mais ce n'est probablement pas la meilleure, en termes de temps de trajet. Cette branche des mathématiques qui s'appelle la "recherche opérationnelle" dispose de bien d'autres méthodes pour connecter entre eux des points sur une carte, de telle sorte que le temps total de trajet soit minimal. Demandons à un mathématicien professionnel de les rechercher pour nous. Peut-être, selon le type de contraintes, y mettra-t-il une demi-journée ou une journée, qu'importe ? Le programme informatique, quant à lui, en sera fort peu modifié, mais le temps réel de trajet pourra être réduit de 10 % ou 20 % par rapport aux solutions les plus évidentes : celles auxquelles on penserait au premier chef. Et nous pourrons conserver notre vieux bus et son chauffeur asthmatique.

La situation est la même à bord d'un voilier en régate : vous voulez remonter au vent. Celui qui est fin barreur y parviendra avec des réglages adéquats de voile et de route ; il s'écartera peu de la trajectoire optimale, et achèvera la course avec une grande économie de moyens. Le néophyte, au contraire, ne saura pas serrer le vent au plus près ; comme il est riche, il acquerra, pour compenser, une voilure plus vaste ou un bateau plus long ; mais il fera davantage de manoeuvres, perdra du temps, finira la course au milieu des rires et, de plus, il sera tout mouillé.

La nature sait punir avec ironie celui qui en méconnaît les lois.

Bernard Beauzamy