from the past → LIVRE.ppy

Ce texte est paru dans le numéro spécial de décembre 1976 du Bulletin de liaison de la section « Informatique et enseignement » de l'INRDP (Service des études et recherches pédagogiques). Que pouvait-on attendre de l'informatique dans l'enseignement, compte tenu du fait qu'il n'était pas question - à cette date - de l'enseigner pour elle-même ?

Clefs pour l'introduction de l'informatique dans l'enseignement secondaire

Monsieur le Professeur Jacques Arsac

L'expérience de l'introduction de l'Informatique dans l'Enseignement du second degré est maintenant bien lancée. Débutée par la formation des professeurs de toutes disciplines, soutenue par l'installation d'ordinateurs dans plusieurs lycées, elle attend d'être jugée à ses fruits. Sa portée est d'ailleurs considérable, et les enseignants d'autres pays, les Etats-Unis notamment, la suivent avec le plus grand intérêt. Elle a été au centre des préoccupations du colloque international qui s'est tenu sous l'égide de l'I.F.I.P. (Fédération Internationale pour le traitement de l'information) à Marseille en septembre 1975. Aussi n'est-il pas vain de reposer la question : pourquoi l'

informatique au lycée, et que peut-on en attendre, compte tenu notamment de ce que, dans l'état actuel de l'expérience, il n'est pas question de l'enseigner pour elle-même ? Je reprendrai ici les conclusions du colloque international de Sèvres, 1970, en les actualisant un peu.

Il faut d'abord écarter radicalement l'idée que l'informatique entre au lycée parce que c'est une technique importante qui risque d'affecter plus ou moins la vie de tous les jours et qui pèse lourd dans l'économie, voire l'indépendance d'un pays. On pourrait en dire autant de l'automobile ou de l'avion. Cela ne justifie pas leur entrée dans l'enseignement secondaire général.

Il est admis maintenant que l'informatique est une science, avec son objet spécifique : l'information et ses traitements, avec ses méthodes, ses outils (l'ordinateur étant le plus important, mais l'informatique n'est pas plus la science de l'ordinateur que l'astronomie n'est celle du télescope...). En tant que telle, elle fait partie du patrimoine culturel de la fin du 20^{e} siècle. Elle a sa place dans tout enseignement scientifique. Mais là encore, on n'atteint qu'une partie de la réalité. L'astronomie est une science, et son importance est grande si l'on éprouve la moindre curiosité quant à l'univers qui nous entoure. Cela ne suffit pas à lui valoir une place de choix au lycée.

C'est pourquoi le colloque de Sèvres soutenait

l'introduction de l'informatique parce que « *une de ses caractéristiques est de créer chez les élèves une mentalité algorithmique, opérationnelle, organisatrice, laquelle est souhaitable pour bien des disciplines* ». Plus que l'objet de l'informatique (l'information), c'est sa méthode, son esprit, qui lui valent l'attention qu'on lui porte aujourd'hui. C'est ce que je voudrais essayer de dégager ici.

L'ACTIVITÉ ALGORITHMIQUE

La notion d'algorithme est loin d'être claire. Dans de nombreux ouvrages on en trouve une définition qui ne permet pas de la distinguer de la notion de programme. Elle lui est pourtant de beaucoup antérieure. Prenons ici le mot « algorithme » comme désignant l'ensemble des opérations qui, appliquées à des données acceptables, fourniront la solution cherchée.

On pense aussitôt à des exemples pris en mathématique. Ainsi, dès le plus jeune âge, on fait acquérir aux enfants la pratique de l'algorithme d'addition dans notre système de numération. Ils apprennent à ajouter des nombres d'un chiffre (en comptant sur les doigts, avec un boulier, ou mieux en sachant par cœur une table d'addition). Puis ils apprennent à additionner 2 nombres de 2 chiffres, en utilisant le mécanisme de la retenue. Puis ils apprennent à généraliser pour l'addition d'un nombre quelconque de chiffres. Je ne suis pas très sûr qu'à aucun moment de leur scolarité on leur explicite ce

dernier algorithme. Il faut reconnaître qu'il est particulièrement désagréable à énoncer.

De la même façon, les élèves apprendront à trouver le milieu d'un segment en traçant deux arcs de cercle et leur corde commune. Ont-ils conscience que c'est aussi un algorithme ? Eloignons nous des mathématiques. La conjugaison d'un verbe de la langue française relève d'un algorithme implicite. On donne le modèle de conjugaison de verbes types. Pour conjuguer un verbe quelconque, si ce n'est pas un verbe type, il faut découvrir la transformation qui fait passer de l'infinitif à la forme désirée, et l'appliquer à ce verbe.

Pas plus que pour l'addition, je ne crois pas que l'on explicite jamais un tel algorithme. La difficulté réside en fait dans l'absence de langage pour le dire. Un algorithme, c'est nécessairement un ensemble de règles de manipulation formelle correspondant à un niveau d'abstraction pas toujours facile à exprimer ou à appréhender pour de jeunes esprits.

L'informatique ne peut laisser ses algorithmes dans l'ombre. Les langages de programmation sont là pour les dire et le recours aux ordinateurs exige qu'on les dise.

On peut conserver l'attitude empirique consistant à les mettre en évidence sans les justifier. C'est une pratique beaucoup trop courante en informatique, et je ne peux que me réjouir des efforts entrepris par Dijkstra, Floyd, Wirth pour tenter de convaincre que

l'on ne peut sérieusement continuer dans cette voie. *« Essayer un programme sur des données particulières peut servir à prouver qu'il contient des erreurs, jamais qu'il est correct... »*

Il faut bien voir où réside la difficulté. J'ai dit que l'on ne possède pas de bon langage pour dire les algorithmes. Faute de quoi, les langages de programmation ont joué un rôle de substitut. Mais s'ils sont aptes à dire les algorithmes, ils font peser sur eux les contraintes liées à la technologie des ordinateurs : nature séquentielle des instructions, place limitée en mémoire... Dans cette langue, on n'obtient pas la forme d'un algorithme la plus adaptée à sa démonstration...

Faut-il se décourager, et proclamer que rien ne presse, l'informatique en étant encore à chercher comment dire les algorithmes, et comment en justifier leur valeur. On s'en occupera au lycée plus tard. Je ne pense pas que l'on puisse soutenir raisonnablement ce point de vue. Il y a deux choses dans l'activité algorithmique : la mise en place de l'ensemble de règles efficaces, la preuve qu'elles sont efficaces. Dans toutes les disciplines, l'apprentissage par la pratique précède la preuve par le raisonnement.

L'ACTIVITÉ OPÉRATIONNELLE

Par certains côtés, les algorithmes revêtent un caractère statique. Ils décrivent un certain nombre de relations liant les données aux résultats, par l'intermédiaire d'autres éléments intervenant

éventuellement pour simplifier la présentation, Ainsi, si je décris un algorithme de multiplication, je ferai apparaître des produits partiels dont la somme forme le résultat cherché. Mais je n'ai aucune raison de préciser l'ordre dans lequel je les calcule. Celui-ci n'a aucune influence sur la valeur du résultat. De la même manière, je peux décrire la façon dont on construit une phrase avec un groupe nominal sujet et un groupe verbal.

C'est tout autre chose que de mettre en jeu effectivement cet algorithme pour lui faire produire les résultats qu'il décrit. L'informatique développe essentiellement des schémas dynamiques, temporels, où l'ordre des opérations joue un rôle essentiel. C'est pour en rendre compte que l'on a inventé l'organigramme. Les langages de programmation comportent tous cette faculté de préciser l'enchaînement des opérations, mettant en évidence les différentes circonstances ainsi que leur traitement. La façon dont l'informatique lie le nom et la valeur d'un objet est typique de cette activité opérationnelle. Dans un schéma algébrique, les objets sont désignés (s'il y a lieu) par des noms, auxquels sont associées des valeurs. Le nom n'est qu'un mécanisme de désignation, et l'on ne change rien dans le schéma si l'on remplace un nom par sa valeur. Les physiciens ont une bonne habitude de cette façon de faire, et leurs formules correspondent très exactement à ceci. Cette façon d'opérer permet de recourir au mécanisme de substitution qui est à la base de toutes les manipulations de formule chères à nos élèves : « *je*

remplace V par sa valeur R I... ».

En informatique, il en va autrement. Il n'y a pas de lien définitif entre le nom et la valeur d'une chose. L'instruction d'affectation que vous retrouvez dans tous les langages a pour but de définir la nouvelle valeur d'un élément. Au moment où on l'exécute, la valeur associée à un nom change.

Si à un certain moment `J` a la valeur 5, et si j'exécute l'instruction L.S.E. `J <- J + 1` alors `J` change de valeur et vaut dorénavant 6, jusqu'à une nouvelle affectation. Ceci a été très directement inspiré par la structure des ordinateurs. Ils comportent une mémoire formée de cases. Chaque case est désignée par un numéro, son adresse, qui est vraiment le nom associé à la case. Chaque case peut contenir une valeur. Exécuter un programme, c'est modifier le contenu de la mémoire, donc changer certaines valeurs associées à certains noms.

Dès lors, on ne peut plus parler de la valeur de la variable `F`. Il faut parler de la suite des valeurs que prend cette variable, et l'on ne peut en rendre compte correctement qu'en reprenant le mécanisme exact d'exécution du programme.

Cette façon de dissocier le nom et la valeur des choses est caractéristique de l'informatique. Mais elle déborde notablement de cette science. Il ne m'appartient pas de rappeler comment la pensée humaine a réussi à reconnaître que le nom n'est qu'une étiquette attachée aux êtres, *« Sous un tout*

autre nom, la rose sentirait aussi bon »... dit Juliette.

L'ACTIVITÉ ORGANISATRICE

En 1970, au colloque de Sèvres, la mentalité organisatrice nous était apparue comme très clairement favorisée par l'informatique. Aussi n'ai-je pas été stupéfait en relisant ce passage des conclusions de ce colloque :

« *Sur le plan de l'organisation, nous pouvons traiter maintenant brièvement ce point. En fait, organisation est presque synonyme de construction d'algorithme.* »

Cette confusion entre organisation et algorithme est troublante, mais peut assez bien s'expliquer par l'état d'avancement des réflexions sur la programmation à cette époque. On trouve un autre signe de ceci dans la méfiance dont ce colloque faisait preuve à l'égard de la programmation : il recommandait de ne pas la confondre avec l'informatique, et surtout de ne pas croire que l'on aurait avancé en mettant en place un cours FORTRAN ou L.S.E. !

En fait, le problème est au niveau même de la définition de la programmation.

Pour certains, la programmation se réduit à l'écriture d'un algorithme, par ailleurs totalement précisé, dans un langage approprié. La question essentielle est « *mettrai-je un point ou une virgule à tel endroit...* ». Dans cet esprit, un cours de

programmation se réduit à l'exposé de la liste des instructions existant dans un langage, avec la façon de les écrire, et leur interprétation. C'est le petit côté des choses, et il serait grave qu'on le confonde avec l'informatique. Non que ce soit sans importance. C'est plus grave que l'orthographe : en langage FORTRAN, par exemple :
`D05I = 1.10` est une instruction affectant à la variable `D05I` la valeur 1.10
`D05 I = 1,10` est l'instruction provoquant la répétition des lignes qui suivent jusque et y compris la ligne 5, en donnant successivement à `I` les valeurs entières de 1 à 10.

Cette précision des notations a ses mérites, et c'est une école de rigueur à laquelle les élèves ne sont pas toujours habitués. Et l'ordinateur met comme une certaine méchanceté à relever toutes ces erreurs !

Mais on apprend très vite à circuler à travers ces règles tatillonnes des langages de programmation, dès que l'on s'est un peu familiarisé avec leurs universaux. Car ils sont tous construits sur le même moule, descendant du même FORTRAN. Je dois avouer que c'est à travers l'apprentissage des langages de programmation que j'ai fini par pénétrer cette idée qu'une grammaire a un double but :

- de façon normative, préciser qu'elles sont les constructions valides, auxquelles on pourra donner un sens. Ainsi, en L.S.E., `X <- 'BIEN'!1` n'est pas une construction

valide, car on ne peut concaténer une chaîne et une constante numérique entière.

- de façon constructive, fournir un éventail de formes possibles, chacune comportant sa nuance sémantique.

Ainsi la langue française autorise deux positions de L'adjectif épithète. Il peut être prénominal (comme dans « une belle robe ») ou postnominal (« une journée ensoleillée »). Il n'a pas la même signification dans les deux positions : un homme grand n'est pas nécessairement un grand homme, il y a loin d'un certain retard à un retard certain... L'adjectif prénominal ne peut se mettre au comparatif : on peut parler d'une belle robe, d'une très belle robe, non d'une plus belle robe...

Dépassons cette conception de la programmation. Pour beaucoup, c'est la création et la rédaction simultanée d'un algorithme dans un langage de programmation. Dans cette conception, toute l'informatique est programmation. Cette vue est manifestement trop large, mais s'explique assez bien par ce que j'ai dit au début. Nous ne sommes pas assez habitués à expliciter les algorithmes. Les langages de programmation apportant une solution au problème, on l'a acceptée telle quelle, sans bien voir ce qu'elle avait d'inutilement contraignant.

On trouve dans les livres de nombreux exemples de programmation illustrant cette façon de procéder. Le problème étant posé (pas toujours très

explicitement), on « évoque » un algorithme de résolution, mettant en évidence avec précision certains de ses détails, donnant des indications sur sa structure d'ensemble.

Soit par exemple à ordonner une suite de nombres, ou de chaînes de caractères (problème du tri). Je cherche la plus grande valeur de la suite. J'échange cette valeur avec celle qui est en dernière position dans la suite. Ceci fait, je suis ramené au problème du tri d'une suite plus courte de 1 élément. Je recommence sur celle-ci jusqu'à épuisement.

Ceci n'est pas un algorithme. Tout est dans le vague. Je n'ai rien dit de la façon de trouver le plus grand élément. Aussi la méthode proposée ne tiendra-t-elle aucun compte d'informations supplémentaires que je pourrais récolter dans la recherche du plus grand élément.

Si je me contente d'une formulation aussi imprécise de la méthode pour en rédiger le programme, je n'aurai aucune façon de justifier celui-ci. Je serai bien obligé de me contenter d'essayer quelques cas particuliers pour rechercher les erreurs possibles (celles auxquelles je pense). Cette façon de faire est remarquablement illustrée par les exemples donnés par Knuth dans son ouvrage *The art of computer programming*.

Ma conviction profonde est que la programmation est entre les deux extrêmes présentés jusqu'à maintenant. L'algorithme existe avant qu'on le

programme. On n'a pas attendu 1950 pour créer des algorithmes ! Mais en lui-même, un algorithme n'a aucune raison d'être adapté à la structure de nos ordinateurs. L'ordre des opérations n'y est pas totalement défini. Les structures d'information qu'il emploie sont celles commodes pour sa description, et non pas le minimum qu'il faut essayer d'implanter dans la mémoire de l'ordinateur.

Dès lors, programmer, c'est organiser un algorithme pour permettre son exécution sur un ordinateur. C'est décider d'un mode de représentation des données et d'un ordre d'exécution des opérations. Les deux choses ne sont pas indépendantes : pour économiser le stockage de variables intermédiaires, il faut faire très attention au moment où elles sont utilisées et bien savoir à partir de quand elles ne servent plus. Ainsi, dans l'exemple du tri évoqué plus haut, on peut décider de trier l'ensemble de valeurs « sur place », par permutations entre eux d'éléments de la suite. On peut aussi utiliser des suites d'auxiliaires, dans lesquelles on range des sous-suites triées que l'on extrait de la suite de départ et que l'on fusionne ensuite. Suivant la place que l'on se donne pour faim le tri, on obtient des programmes aux performances très différentes.

Ainsi se trouvent bien précisées les phases de la résolution informatique d'un problème, ce que l'on appelle la méthode informatique.

LA MÉTHODE INFORMATIQUE

Le point de départ, c'est en général une situation concrète décrite de façon formelle dans le langage courant. Ainsi les bons vieux problèmes d'arithmétique du temps jadis. Mais de façon très semblable, les problèmes de physique au lycée.

La première chose à faire, c'est d'en extraire un problème posé en terme de relations strictement formelles entre les données et les résultats. C'est l'analyse du problème. C'est une phase essentielle, propre à l'esprit humain. Elle représente en effet un passage de ce qui est conçu de façon informelle par l'esprit, ce qui est sémantique, à une représentation formelle traitable au plan de relations syntaxiques.

Le langage est le lieu de cette rencontre de la forme et du sens. Il joue là-dedans un rôle essentiel. C'est bien pourquoi l'informatique a tant de relations avec la linguistique. Il n'est pas facile de trouver une représentation formelle qui implique de façon unique la sémantique du problème. Tout langage n'y est pas apte. C'est pourquoi, là encore, chaque discipline s'est constitué son propre vocabulaire, particulièrement précis où la notion du synonyme n'existe pas. En physique, le mot « force » a un sens parfaitement défini qui ne se prête à aucune interprétation. Nous sommes loin de Lewis Caroll :
« Quand j'utilise un mot, dit HUMPTY DUMPTY d'un ton plutôt méprisant, il signifie juste ce que j'ai choisi pour lui, ni plus ni moins...
La question, répondit Alice, est de savoir si vous pouvez faire que les mots signifient tant de choses

différentes ?
La question, dit HUMPTY DUMPTY, est de savoir qui commande, c'est tout... »

Cette analyse, que l'on recommandait tant aux jeunes élèves du primaire (quelle est la question posée dans le problème, quelles sont les données...), combien aujourd'hui en perçoivent l'importance ? Combien, en physique, pêchent dans l'arsenal des formules quelque chose qui puisse accepter les données de l'énoncé et fournisse un résultat, sans avoir le moins du monde tenté de dégager le phénomène physique en jeu, les conditions dans lesquelles il se produit, et en conséquence, les lois physiques qu'il convient d'utiliser. Dans sa thèse de 3e cycle (Université de PARIS VI), Mademoiselle Martine Rousseau a réalisé un programme d'intelligence artificielle qui simule ce travail des mauvais élèves. Le plus surprenant est le succès considérable de la méthode appliquée aux problèmes d'électricité de la classe de première C. Ceci pose de très sérieuses questions sur la valeur pédagogique de ces problèmes...

Ce ne serait pas un des moindres mérites de l'introduction de l'informatique au Lycée que de contraindre à une analyse poussée avant toute résolution de problème. Certes, elle n'est pas seule à exiger cette analyse, mais peut-être le jugement de l'ordinateur est-il plus sévère (et mieux accepté) que celui du professeur !

L'analyse faite, le problème correctement exprimé, il faut trouver, ou construire, la méthode qui permettra de passer des données aux résultats, conformément aux relations existant entre eux au terme de l'analyse. Ce travail d'algorithmique serait très utilement guidé par un enseignement approprié en mathématique. Car il faudrait s'efforcer de prouver que l'algorithme proposé fournit effectivement la solution du problème. Il faudrait pouvoir le faire aussi bien d'un algorithme de calcul (les mathématiciens en ont l'habitude) que d'un algorithme d'analyse syntaxique (le demandera-t-on au mathématicien ou au grammairien), ou d'un algorithme de tri...

L'algorithme dégagé, et fondé (soit sur de bonnes raisons, soit sur une vraie démonstration), alors on attaque la programmation. On étudie les structures de données possibles et les formes de programme qui en résultent. Dans l'idéal, on en compare les mérites et on choisit la meilleure solution.

Bien souvent, malheureusement, on se contentera d'une solution possible, faute de pouvoir en exhiber d'autres, ou de pouvoir utilement les comparer. En cela, on se rapprochera des médecins de Molière ! Dans cette phase du travail, il faut savoir penser en référence au déroulement dans le temps du programme, aux propriétés opératoires des instructions du langage utilisé. Il faut savoir organiser les éléments sur lesquels on travaille, mettre de l'ordre dans leur multiplicité. Ce sont les vertus opérationnelles et organisatrices soulignées à Sèvres

en 1970...

L'ORDINATEUR

C'est là qu'on le retrouve. Car il est impossible de faire comprendre pourquoi une telle organisation est nécessaire sans donner un minimum de renseignements sur la structure des ordinateurs. La programmation, telle que nous la pratiquons, n'est pas née de la nécessité de dire des algorithmes mais de celle de les exécuter sur des machines munies d'une seule unité de traitement séquentielle et d'une mémoire finie. Il faut quelque connaissance des mécanismes d'adressage pour comprendre que les seules structures d'information aisément manipulables sont les vecteurs et les chaînes de caractères. Les autres ne le sont qu'au prix de systèmes interprétatifs, lents à l'exécution.

Si les langages sont un moyen de communication entre interlocuteurs dont il importe de connaître quelque peu la psychologie (c'est du moins, je crois, ce que disent les professeurs de langues vivantes) alors il est bon de connaître un peu les réactions de l'ordinateur avant de lui parler.

L'ENSEIGNEMENT DE LA PROGRAMATION

Le colloque de Sèvres avait attiré l'attention sur la nécessaire distinction entre informatique et programmation. Il avait insisté sur le fait qu'il fallait surtout ne pas tomber dans le travers d'instituer au

lycée des cours sur un langage particulier, FORTRAN ou autre (L.S.E. par exemple). Certes, ce serait une grave erreur que de se Lancer dans ces cours fastidieux où l'on présente l'un après l'autre les ordres du langage, la façon de les écrire, leurs diverses variantes. Il y a des brochures où ceci est précisé. On ne peut rien faire sans un langage, mais on ne sait rien faire quand on ne connaît que le langage.

La difficulté vient de ce qu'il faudrait enseigner la programmation, indépendamment de ses langages. Dans l'état actuel des choses, on ne sait guère le faire. Sans doute est-ce aussi difficile que d'enseigner la démonstration en mathématique. Mais le savoir faire naîtra de l'expérience. De toute façon, on ne peut enseigner la programmation sans l'ancrer dans un langage particulier, faute de quoi on ne ferait que du verbiage. Mais à travers l'accident d'un langage particulier, il faut viser à mettre en évidence ce qui est universel (par exemple la notion d'affectation, de branchement, de procédure, de test, celle de structure d'information et d'opérations spécifiques de chaque structure, etc.).

On s'efforcera d'insister sur la part de création de l'algorithme, et, toutes les fois que ce sera possible, on en donnera des justifications, sinon des preuves. On essaiera de montrer ce qu'il y a d'arbitraire dans l'écriture du programme. Ce faisant, il faudra proposer des alternatives à la solution adoptée, et fonder sur quelque argumentation les choix qui sont faits.

Nous n'en sommes pas encore à la « programmation systématique » prônée par Niklaus Wirth. Mais du moins sortira-t-on de cette « cuisine » que l'on reproche trop souvent aux informaticiens. Encore que, selon Claude Bernard, *« La science est comparable à un salon brillamment illuminé dans lequel on ne parvient qu'après avoir traversé une longue et affreuse cuisine... »*.

CONCLUSION

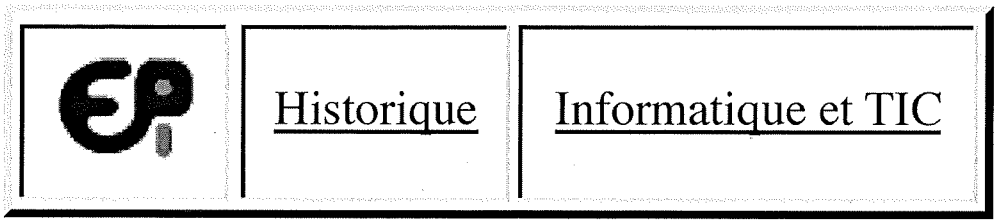
J'aurais bien mal atteint mon but si l'on ne retenait de ces quelques réflexions qu'un aspect négatif. J'ai mis en vedette des points difficiles de l'informatique. Je ne suis pas pessimiste on viendra à bout de ces difficultés, les recherches sont en bonne voie.

Je crois beaucoup à la valeur et à l'importance de l'expérience entreprise depuis 5 ans : il faut que l'introduction de l'informatique dans l'enseignement du second degré soit un succès, parce que c'est souhaitable, parce qu'un échec serait très difficile à surmonter.

Pour cela, il faut viser juste. Il faut reconnaître les écueils possibles et les éviter. Agir autrement, c'est faire la politique de l'autruche. Je souhaite de tout cœur que les professeurs de lycée réussissent dans l'œuvre qu'ils ont entreprise, et les assure du soutien total des informaticiens universitaires.

Jacques Arsacc
Directeur de l'Institut de Programmation de Paris

Association EPI



NOUS AVONS LU
sur le Net

Depuis 10 ans TIC et informatique sont enseignées dès le primaire, en Inde

Sur *binaire*, le blog de la SIF (extrait).

Après la Bavière et Israël, François Tort nous entraine en Inde avec un **Entretien avec** Pradeep Kumar Misra, professeur associé au département B.Ed/-M.Ed. à la faculté de sciences de l'éducation de l'université M. J. P. Rohilkhand, à Bareilly, dans l'état de l'Uttar Pradesh. Il forme de futurs enseignants. Ses travaux de recherche portent sur les technologies éducatives, la formation des

enseignants et la formation tout au long de la vie.

...

40 % des indiens, soit 430 millions de personnes, ont moins de 18 ans. Cela fait de l'Inde le pays abritant la plus grande population d'enfants au monde. La promesse faite par le gouvernement que « chaque village devra être connecté par une bande passante, et l'éducation pourra être offerte aux enfants des écoles même les plus « reculées » montre que dans un avenir proche les cours d'informatique à l'école seront nécessaires et souhaités. En résumé, l'informatique est apparue comme une composante vitale de l'enseignement scolaire indien mais il nous reste encore à faire de plus gros efforts pour définir et mettre en oeuvre un programme et des contenus d'enseignement détaillés, offrant un bon équilibre entre connaissances théoriques et pratiques, et développant de nouvelle méthodes pédagogiques, mais par dessus tout aidant les enfants *« to grow virtual but remain socially grounded »*.

À lire à : http://binaire.blog.lemonde.fr/

Association EPI
novembre 2014