La machine à loisirs

Christian.Kauth@epfl.ch, EPFL - STI, coach de PolyProg et doctorant en microsystèmes et microélectronique

PolyProg welcomes you onto the EPFL campus for a new academic year that will be full of projects, fascinating subjects and student life. To foster long-term success without having the latter fall by the wayside, PolyProg invented a leisure time creator. It is, as is any real system, causal, meaning that an initial investment is required of you in order to reap its benefits.

PolyProg vous souhaite la bienvenue sur le campus de l'EPFL pour une nouvelle année académique riche en projets, matières passionnantes et vie estudiantine. Afin de favoriser votre succès à long terme sans que cette dernière ne reste sur le carreau, PolyProg a inventé une machine à loisirs ! Elle est, comme tout système réel, causale, c'est-à-dire qu'un investissement initial vous est réclamé avant que vous n'en puissiez récolter les fruits.

Bénéfices de la programmation & algorithmique

Si l'année commence en rose avec tous les apéros de bienvenue, les cours et projets dicteront bientôt votre quotidien et vous laisseront en fin de semestre peu de temps pour souffler. L'expérience montre que de bonnes connaissances en programmation et algorithmique permettent souvent aux étudiants d'être plus efficaces dans leurs projets.

Programmation: avoir de la facilité à écrire le simulateur d'un prototype permet très vite de poser des jalons au stade initial d'un projet.

Algorithmique: un esprit analytique entraîné à la pensée algorithmique est susceptible de trouver rapidement des approches structurées à des problèmes.

Soyez futés et apprenez l'algorithmique dès vos premiers jours à l'EPFL avec PolyProg !

PolyProg c'est ...

L'association des étudiants qui aiment les algorithmes et la programmation. Nous organisons des séminaires et des concours à ce sujet, et nous offrons la possibilité de participer à des événements nationaux et internationaux comme l'**ACM ICPC** ou l'**Helvetic Coding Contest** hc2.ch. Si la syntaxe d'un langage de programmation s'apprend en cours, dans un bouquin ou sur le net, le mindset d'un programmeur se forge avec le temps et la pratique !

Venez nous voir à notre stand lors de la semaine de bienvenue, notre assemblée générale ou participez aux séminaires ! Vous pouvez suivre notre actualité sur polyprog.epfl.ch.

Aspirez à l'efficacité

Voici à présent un petit exercice pour tester vos capacités à penser *out of the box*. Imaginez que votre *prof* de génétique vous mette à disposition la base de données avec l'ADN de toute la population suisse, soit quelques dizaines de terabytes d'information compressée! Supposons toutefois que cette base de données ne soit pas compressée, mais bel et bien un énorme tableau a de N valeurs entières $a[i]$. Votre mission consiste à en identifier la plus petite et la plus grande, et cela au plus vite ! Top chronos.
Quelques minutes vous suffiraient pour écrire ce code en C++ :

```
int min = a[0];
int max = a[0];
for ( int i=1 ; i<N ; ++i )
   if      ( a[i] < min )  min = a[i];
   else if ( a[i] > max )  max = a[i];
```

Et laisser faire le travail à l'ordinateur, qui vous sortira le minimum et maximum après quelques heures de calcul.
Mais est-ce que vous ne pouvez pas faire mieux ? Investissez un peu plus de temps dans la réflexion, pour trouver un algorithme qui vous donne le résultat, disons en 75% du temps de calcul nécessaire à l'approche ci-dessus, qui fait $2N$-2 comparaisons. Votre résultat doit être correct et vous ne pouvez évidemment pas le deviner. Comment est-ce possible ? Une discussion autour d'un café avec vos collègues, pourrait-elle vous inspirer éventuellement ?

Indication: qu'est-ce que vous pouvez faire avec deux nombres que vous ne pouvez pas faire avec un seul ?

Solution: comparez les deux prochains nombres entre eux. Le plus petit ne peut être un candidat pour le maximum tandis que le plus grand ne sera jamais le minimum. Cette observation, rétrospectivement évidente, réduit le nombre de comparaisons nécessaires de $2N$-2 à $3/2N$. Vous aurez gagné quelques heures de calcul, moins le temps d'un café sympa, tout comme la satisfaction d'avoir été créatif ☺

```
int min = a[0];
int max = a[0];
   for ( int i=N%2 ; i<N ; i+=2 )
      if ( a[i] < a[i+1] ){
         if ( a[i]   < min ) min = a[i];
         if ( a[i+1] > max ) max = a[i+1];
      } else {
         if ( a[i]   > max ) max = a[i];
         if ( a[i+1] < min ) min = a[i+1];
      }
```

Au plaisir de vous rencontrer ! ■