Tour de magie pour la pause-café

Christian.Kauth@epfl.ch, EPFL - président de PolyProg

At the coffee break.
– How old are you ?
– Just guess my age, I'll answer by younger *or* older*.*
Easy, you rub your hands: a classical dichotomy will do it !
– However I'll lie to you every now and then, but never twice in a row.
How do you tackle this?

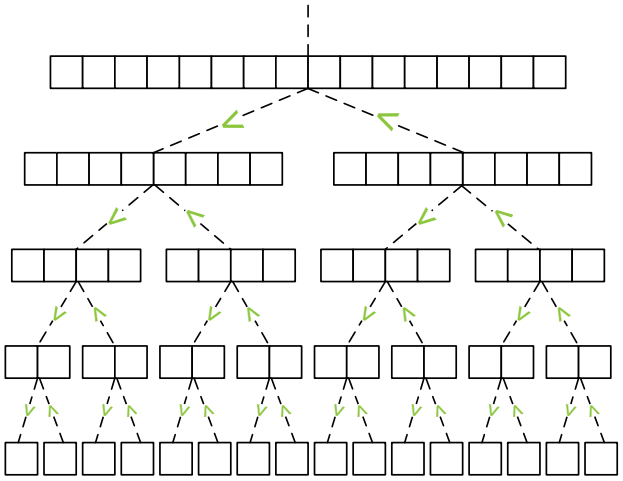
Autour d'une tasse de café.
– Quel âge avez-vous ?
– Devinez, je réponds par *plus jeune* ou *plus âgé*
Facile, vous vous frottez les mains: une petite dichotomie et c'est réglé !
– Par contre je me permets de vous mentir de temps à autre, mais jamais deux fois de suite.
Comment vous y prenez-vous ?

Une pépite de l'Helvetic Coding Contest

Ce problème est extrait de l'**Helvetic Coding Contest 2012** qui s'est déroulé avec grand succès le 17 mars à l'EPFL. Ne cessant de puiser sa motivation dans le renouveau, toujours avec passion et originalité, l'organisateur PolyProg a déclenché un véritable feu d'idées qui se propageait à grande allure. Les 110 jeunes têtes, venant des quatre coins de la Suisse, ont tout donné ! Sponsorisée par OpenSystems et AdNovum, ce fut une journée passée à résoudre des problèmes algorithmiques sans jamais perdre le sourire, que demander de plus ?

Ne soyez pas glouton

Si votre vis-à-vis disait toujours la vérité, une manière efficace pour trouver son âge (un nombre entier entre 1 et 100), serait de couper avec chaque question en deux parties égales l'ensemble des solutions possibles. Si les parties ne sont pas égales, vous risquez de vous retrouver avec la plus grande partie, donc pour minimiser le nombre de questions dans le pire des cas, la coupure au milieu est souhaitable.

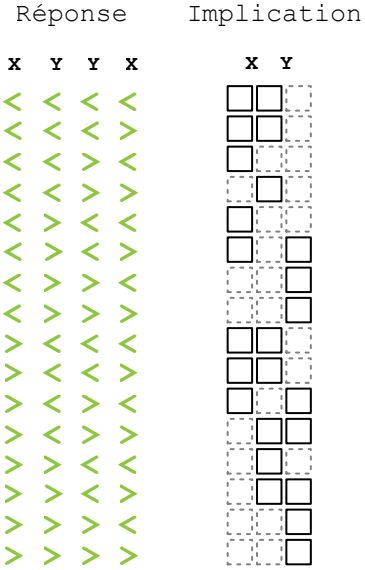


Pour N possibilités initiales (les 100 années dans notre cas, mais N valait 10^6 lors du concours), une telle stratégie nécessite $\lceil log_2(N) \rceil$ questions, soit 20 lors de l'hc^2. Il s'agit d'une recherche binaire, connue sous le nom de **dichotomie**.
Dès que votre vis-à-vis recourt au mensonge, vous avez droit à 200 questions, soit dix fois plus, et à 2 âges-solutions pour compenser l'incertitude. Un premier réflexe pourrait être de répéter chaque question de la dichotomie jusqu'à ce que vous receviez deux fois la même réponse de suite. Là vous savez que votre vis-à-vis ne ment pas (par les règles du jeu, il n'a pas le droit). Par contre s'il est rusé, il peut alterner entre vérité et mensonge !
Une stratégie un peu plus raffinée serait donc de forcer la vérité par un mensonge antécédent. Ceci peut se faire en demandant une valeur extrême (par exemple les 100 années à votre jeune vis-à-vis). S'il répond par *plus âgé*, il ment et au tour suivant il devra être honnête. Malheureusement vous pouvez partir du principe que votre vis-à-vis est assez rusé pour ne pas mentir dans une situation pareille.

Une stratégie élégante

Les solutions possibles à ce problème sont légion, mais elles ont toutes en dénominateur commun de combiner répétition et relation entre les questions. Toutes demandent de l'ingéniosité et remplissent leur inventeur de grande satisfaction. Je vous propose de vous creuser un peu les méninges avant de poursuivre votre lecture.
La suite des 4 questions *XYYX* permet de résoudre le problème. Prenons X inférieur à Y et coupons ainsi l'ensemble des solutions en 3 parties. Pour chacune des 16 réponses possibles (*plus jeune* ou *plus âgé* pour chacune des 4 questions), vous pouvez écarter au moins une des 3 parties. Dans 8 cas, les réponses sur les 2 questions Y sont identiques et correspondent à la vérité. Si au contraire, elles sont opposées, une des deux réponses sur les questions X doit être honnête. Comme ces deux réponses sont

identiques dans 4 cas supplémentaires, elles doivent être franches ! L'analyse des 4 cas restants est laissée à la discrétion du lecteur, mais le tableau suivant résume la situation.



Comme vous ne pouvez pas savoir à l'avance quelle partie sera éliminée, l'optimisation du pire cas réclame de couper l'ensemble des solutions en 3 parties égales. Choisissant *X* et *Y* selon les critères d'une recherche ternaire, cette stratégie ne nécessite pas plus de 136 questions, $4\log_{\frac{3}{2}}(N)$, dans le pire des cas.

```
vector<int> a(1000002,-1);   // possible ages
int X, Y;                    // third cuts
bool r1, r2, r3, r4;         // responses to queries 1 to 4
bool l, m, r;                // left, middle & right feasibility

// [1] initialize candidate ages
for (unsigned int i=1; i<a.size(); i++)
  a[i] = a[i-1] + 1;

while (a.size()>2)
{
  // [2] define ternary-search cuts
  X = (a.size()-2)/3+1;
  Y = a.size()-1-(a.size()-2)/3-1;

  // [3] ask oracle
  r1 = ask_age((a[X]+a[X-1])/2.0);
  r2 = ask_age((a[Y]+a[Y+1])/2.0);
  r3 = ask_age((a[Y]+a[Y+1])/2.0);
  r4 = ask_age((a[X]+a[X-1])/2.0);

  // [4] evaluate feasibilities (0: truth, 1: lie)
  l = false; m = false; r = false;
  for (int c1=0; c1<2; c1++)
    for (int c2=0; c2<2-c1; c2++)
      for (int c3=0; c3<2-c2; c3++)
        for (int c4=0; c4<2-c3; c4++)
        {
          l |= !(( r1^c1) || ( r2^c2) || ( r3^c3) || ( r4^c4));
          m |= !((!r1^c1) || ( r2^c2) || ( r3^c3) || (!r4^c4));
          r |= !((!r1^c1) || (!r2^c2) || (!r3^c3) || (!r4^c4));
        }

  // [5] filter ages
  if (!l && !m)      a = vector<int>(a.begin()+Y+1, a.end());
  else if (!l && !r) a = vector<int>(a.begin()+X, a.begin()+Y+1);
  else if (!m && !r) a = vector<int>(a.begin(), a.begin()+X);
  else if (!l)       a = vector<int>(a.begin()+X, a.end());
  else if (!r)       a = vector<int>(a.begin(), a.begin()+Y+1);
  else if (!m)       a.erase(a.begin()+X, a.begin()+Y+1);
}

printf("Your age is %d or %d\n", a[0], a[a.size()-1]);
```

La solution en C++

Chaque idée, chaque algorithme se laissent implémenter avec plus ou moins d'élégance, souvent synonyme de brièveté.
Suite à une question, vous devez interpréter la réponse obtenue (< – *plus jeune* ou > – *plus âgé*) en fonction de la sincérité de l'interlocuteur (vérité ou mensonge). Le tableau ci-dessous montre les interprétations.

	vérité	mensonge
<	<	>
>	>	<

En substituant **<** par 0, **>** par 1, vérité par 0 et mensonge par 1, l'opération logique de cette table de vérité correspond à celle d'une porte XOR, le symbole ^ en C++.

	0	1
0	0	1
1	1	0

La suite coule de source.

❶ Définissez un vecteur `a` contenant tous les âges possibles en ordre croissant sur lequel vous allez faire une recherche ternaire. Celle-ci consiste à

❷ définir les coupures (au un et deux tiers des éléments),

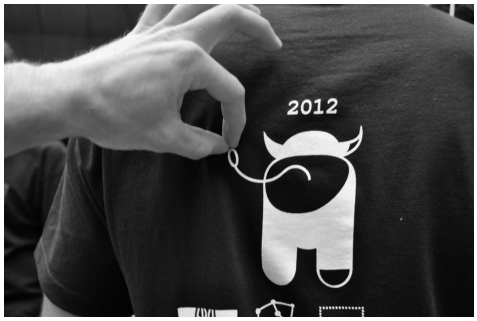
❸ poser les questions pour *XYYX* et

❹ évaluer la faisabilité de l'ensemble solution selon l'astuce XOR et en simulant chaque combinaison de vérité et mensonge. Cette étape peut se faire alternativement par *look-up* dans la table **réponse–implication** ci-dessus. Reste à

❺ écarter l'ensemble des solutions à rejeter. Le code ci-dessus implémente l'algorithme en C++.

La prochaine pause-café

Vous avez aimé cette petite énigme et vous êtes d'humeur à en croquer d'autres ? Alors, ne vous retenez pas et visitez l'ensemble des problèmes du concours sur hc2.ch. Au plaisir de vous accueillir à la prochaine édition de l'Helvetic Coding Contest ! ■



ISSN 1420-7192