Algortihmique d'Aide à la Décision

Par Clément Guilbert

Dossier n°1 : Le Compte Est Bon

Introduction

L'une des épreuves de la fameuse émission "Des chiffres et des lettres" est celle dite du "Compte Est Bon". Au cours de celle-ci, les candidats doivent atteindre un certain résultat a partir de termes tirés aléatoirement, en effectuant le moins de calcul possibles. Ils peuvent additionner, soustraire, multiplier ou diviser les nombres fournis et utiliser les résultats d'opérations intermédiaires pour arriver au nombre final.

C'est cet algorithme que j'ai décidé d'implémenter en C, en permettant de choisir le nombre de termes initialement proposés.

Principe

L'idée principale d'un tel algorithme est d'effectuer récursivement les opérations possibles tant qu'elles améliorent l'écart au résultat. En parcourant plusieurs arbres récursif, on peut ainsi favoriser des branches par rapport à d'autres et conserver la solution optimale.

Code

Main.c

```
 #include  "numgen.h"
#include  "compte.h"

long resultat=0;
long meilleurecart=0;
int meilleurcalcul=0;

int  main(int  argc, char *argv[])
{
    int i;
    int params = 6;
    termeini=genNumbers(params);
    resultat=genRes();
    printf("Nombre a atteindre : %ld\n", resultat);
    printf("Liste de nombres : \n");

    for(i=0;i<params;i++)
        printf("%ld\n", termeini[i]);
    for(i=0;i<params;i++)
        terme[6][i]=termeini[i];
    for(i=0;i<params;i++) // initialise terme[][]
        terme[6][i]=termeini[i];

    meilleurecart=LONG_MAX;
    meilleurcalcul=INT_MAX;
    compte(params);
    affichesolution(meilleurcalcul);
    free(termeini);
    return(0);
}
```

compte.h

```
#pragma once

#include <stdlib.h>
#include <stdio.h>

#ifndef GLOBALS
#define GLOBALS
extern long resultat;
extern long meilleurecart;
extern int meilleurcalcul;
#endif

void affichesolution(int l);
void compte(int l);

typedef struct
{
  long valeur1[16];
  char operation[16];
  long valeur2[16];
  long resultat[16];
} solution;

long * termeini;

long terme[16][16];

solution savesolution;
solution bestsolution;
```

compte.c

```
#include  "compte.h"

void  affichesolution(int  calcul)
{
    int i;
    printf("******************\n");
        for(i=6;i>calcul;i--)
            printf("%lu %c %lu = %lu\n",bestsolution.valeur1[i],bestsolution.operation[i],bestsolution.valeur2[i],bestsolution.resul
    printf("******************\n");
}

void  compte(int  calcul)
{
    int i,j,k,n;
    long terme1=0,terme2=0;
    long ecart, r;
    ecart=resultat-terme[calcul][0];

    if(ecart<0)
        ecart=-ecart;
    if(ecart<=meilleurecart)
    {
        if(ecart<meilleurecart)
        {

            meilleurecart=ecart;
            bestsolution=savesolution;
            meilleurcalcul=calcul;
        }
        else  // (ecart==meilleurecart)
        {
```

```
            if(calcul>meilleurcalcul)
            {
                bestsolution=savesolution;
                meilleurcalcul=calcul;
            }
        }
    }
    if(calcul==1)
        return;
    for(i=0;i<calcul-1;i++)
    {
        for(j=i+1;j<calcul;j++)
        {
            terme1=terme[calcul][i];
            terme2=terme[calcul][j];
            n=1;
            for(k=0;k<calcul;k++)
            {
                if(k!=i && k!=j)
                {
                    terme[calcul-1][n]=terme[calcul][k];
                    n++;
                }
            }
            terme[calcul-1][0]=terme1+terme2;
            savesolution.valeur1[calcul]=terme1;
            savesolution.operation[calcul]='+';
            savesolution.valeur2[calcul]=terme2;
            savesolution.resultat[calcul]=terme[calcul-1][0];
            compte(calcul-1);
            _Bool check = (terme1!=1 && terme2!=1);
            if(check)
            {
                terme[calcul-1][0]=terme1*terme2;
                savesolution.operation[calcul]='*';
                savesolution.resultat[calcul]=terme[calcul-1][0];
                compte(calcul-1);
                if(terme1>=terme2)
                {
                    terme[calcul-1][0]=terme1-terme2;
                    if(terme[calcul-1][0])
                    {
                        savesolution.operation[calcul]='-';
                        savesolution.resultat[calcul]=terme[calcul-1][0];
                        compte(calcul-1);
                    }
                    r=terme1%terme2;
                    if(!r)
                    {
                        terme[calcul-1][0]=terme1/terme2;
                        savesolution.operation[calcul]='/';
                        savesolution.resultat[calcul]=terme[calcul-1][0];
                        compte(calcul-1);
                    }
                }
                else
                {
                    terme[calcul-1][0]=terme2-terme1;
                    savesolution.valeur1[calcul]=terme2;
                    savesolution.operation[calcul]='-';
                    savesolution.valeur2[calcul]=terme1;
                    savesolution.resultat[calcul]=terme[calcul-1][0];
                    compte(calcul-1);
                    r=terme2%terme1;
                    if(!r)
                    {
```

```
                    {
                        terme[calcul-1][0]=terme2/terme1;
                        savesolution.operation[calcul]='/';
                        savesolution.resultat[calcul]=terme[calcul-1][0];
                        compte(calcul-1);
                    }
                }
            }
            else  if(terme1>=terme2)
            {
                terme[calcul-1][0]=terme1-terme2;
                if(terme[calcul-1][0])
                {
                    savesolution.operation[calcul]='-';
                    savesolution.resultat[calcul]=terme[calcul-1][0];
                    compte(calcul-1);
                }
            }
            else
            {
                terme[calcul-1][0]=terme2-terme1;
                savesolution.valeur1[calcul]=terme2;
                savesolution.operation[calcul]='-';
                savesolution.valeur2[calcul]=terme1;
                savesolution.resultat[calcul]=terme[calcul-1][0];
                compte(calcul-1);
            }
        }
    }
}
```

numgen.h

```
#pragma once

/*
   A C-program for MT19937, with initialization improved 2002/1/26.
   Coded by Takuji Nishimura and Makoto Matsumoto.

   Before using, initialize the state by using init_genrand(seed)
   or init_by_array(init_key, key_length).

   Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura,
   All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

     1. Redistributions of source code must retain the above copyright
        notice, this list of conditions and the following disclaimer.

     2. Redistributions in binary form must reproduce the above copyright
        notice, this list of conditions and the following disclaimer in the
        documentation and/or other materials provided with the distribution.

     3. The names of its contributors may not be used to endorse or promote
        products derived from this software without specific prior written
        permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
   A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
```

```
   CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
   PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
   PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
   LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
   NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
   SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


   Any feedback is very welcome.
   http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html
   email: m-mat @ math.sci.hiroshima-u.ac.jp (remove space)
*/
#include <time.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
/* Period parameters */
#define N 624
#define M 397
#define MATRIX_A 0x9908b0dfUL   /* constant vector a */
#define UPPER_MASK 0x80000000UL /* most significant w-r bits */
#define LOWER_MASK 0x7fffffffUL /* least significant r bits */

static unsigned long mt[N]; /* the array for the state vector  */
static int mti=N+1; /* mti==N+1 means mt[N] is not initialized */

/* initializes mt[N] with a seed */
void init_genrand(unsigned long s);

/* initialize by an array with array-length */
/* init_key is the array for initializing keys */
/* key_length is its length */
/* slight change for C++, 2004/2/26 */
void init_by_array(unsigned long init_key[], int key_length);

/* generates a random number on [0,0xffffffff]-interval */
unsigned long genrand_int32(void);

/* generates a random POSITIVE number on [0,0xffffffff]-interval */
unsigned long genrand_int32_pos(void);

/* generates a random number on [0,0x7fffffff]-interval */
long genrand_int31(void);
/* generates a random number on [0,1]-real-interval */
double genrand_real1(void);

/* generates a random number on [0,1)-real-interval */
double genrand_real2(void);

/* generates a random number on (0,1)-real-interval */
double genrand_real3(void);

/* generates a random number on [0,1) with 53-bit resolution*/
double genrand_res53(void) ;
/* These real versions are due to Isaku Wada, 2002/01/09 added */

long * genNumbers(int n);

long genRes();
```

numgen.h

```
#include "numgen.h"

long * genNumbers(int n)
{
    init_genrand(7);
    long * numbers = (long *) malloc (n+1);
    if(numbers==NULL) exit(1);
    for(int i=0;i<n;i++) {

        numbers[i] = genrand_int32_pos()%100;
    }
    return numbers;
}

long genRes()
{
    return genrand_int32_pos()%900+100;
}
```