16,23 et 30 janvier 2002

Simulation d'expériences probabilistes (II)

Laure Danthony
http://www.ens-lyon.fr/~ldanthon/

Fonction auxilliaire

Sans commentaire :

```
function tirage_p(p : integer;n:integer):boolean;
{renvoie true avec une proba p/n}
var res : integer;
begin
   res := random(n)+1;
   if (res <= p) then tirage_p := true else tirage_p:=false
end;
```

1 Oeufs en chocolat

1. Il est facile de voir que si $P(0 \to 1)$ désigne la probabilité de passer à 1 figurine ("différente") si on en possède 0 en achetant un œuf, alors on a : $P(0 \to 1) = 1$, $P(1 \to 2) = \frac{3}{4}$, $P(2 \to 3) = \frac{1}{2}$ et $P(3 \to 4) = \frac{1}{4}$. Donc $E(T) = 1 + \frac{4}{3} + 2 + 4 \simeq 8$.

2. La fonction oeufs :

```
function oeufs(nb_coups : integer) : real;
var f1,f2,f3,f4 : boolean;
   alea,T,i     : integer;
   res          : real;
begin
   res:=0;
   for i:=1 to nb_coups do begin
      T:=0;
      f1:=false;f2:=false;
      f3:=false;f4:=false;
      while not (f1 AND (f2 AND (f3 AND f4))) do
      begin
         alea := random(4)+1;
         if alea=1 then f1:=true
         else if alea=2 then f2:=true
         else if alea=3 then f3:=true
         else f4:=true;
         T:=T+1
      end;
      res:=res+T
   end;
```

```
    oeufs:= res / (nb_coups)
end;
```

3. On teste : `writeln(oeufs(10000):3:3)` et on obtient par exemple $8,291$.

2 Le problème des menteurs

1. On a la relation de récurrence $p_{n+1} = (1-p) + p_n(2p-1)$ avec $p_1 = p$, qui se résout (facilement !) en $p_n = (2p-1)^{n-1}\left(p - \frac{1}{2}\right) + \frac{1}{2}$

2. Lorsque $p = 1$, $p_n \underset{n\to\infty}{\longrightarrow} 1$. Si $p = 0$, la suite diverge. Si $0 < p < 1$, alors $p_n \underset{n\to\infty}{\longrightarrow} \frac{1}{2}$

3. Fonction menteurs :

```
function menteurs(param :integer; nb_gens : integer;nb_coups:integer):real;
{k=1/p}
var i,k,moy,tot : integer;
var aux,save,tmp : boolean;
begin
   tot:=0;
   for k:=1 to nb_coups do
   begin
      if (random(2)=1) then tmp:=true else tmp:=false;{valeur ini}
      save:=tmp;
      for i:=2 to nb_gens do
      begin
         aux:=tirage_p(1,param);
         if not aux then tmp:=(not tmp)
      end;
      if (save=tmp) then tot:=tot+1
   end;
   menteurs:=tot/nb_coups;
end;
```

4. On teste : `writeln(menteurs(3,1000,1000):3:3)` et on obtient par exemple $0,489$.

3 Points fixes et chapeaux

1. Déclaration des types :

```
 CONST long = 34;
type table = array[1..long] of integer;
```

2. Procédure permutation :

```
procedure permutation(var t : table);
var i,rand : integer;
begin
   for i:=1 to long do t[i]:=i;
   for i:=long downto 2 do
```

```
   begin
      rand:=random(i)+1; {tire dans 1..i}
      swap(rand,i,t)
   end;
end;
```

3. Procédure points-fixes :

```
procedure points_fixes(nb_perm : integer);
var i,k,nb,S : integer;
var t,aux : table;
begin
   for i:=1 to long do aux[i]:=0;
   S:=0;
   for k:=1 to nb_perm do
      begin
         permutation(t); {t contient une nouvelle permutation}
         nb:=0;
         for i:=1 to long do
         begin
            if t[i]=i then inc(nb)
         end;
         S:=S+nb; {pour l'espérance}
         nb:=min(nb,long-1); {pour éviter les effets de bord}
         aux[nb+1]:=aux[nb+1]+1;
      end;
   write('[');
   for i:=1 to long do
      if aux[i]<>0 then write('(',i-1,',',aux[i],');');
   writeln(']');
   writeln('proba 0 pt fixe',aux[1]/nb_perm:4:4);
   writeln('esperance : ',S/nb_perm:4:4);
end;
```

4. Expérimentalement, on trouve en lançant `points_fixes(100000)` une probabilité d'aucun point fixe environ égale à $0,367 \simeq \frac{1}{e}$, le nombre moyen de points fixes étant 1 !