Complexité des algorithmes
Méthode du simplexe

O. Marguin - 4 mai 2011

1. Algorithmes de tri

```
> restart:
> Seed:=randomize():
> tri_insert:=proc(T,n)  # tri de T[1..n] par insertion
     local i,j,aux;
     for i from 2 to n do
        aux:=T[i];
        for j from i-1 by -1 to 1 while T[j]>aux do
           T[j+1]:=T[j]
        end do;
        T[j+1]:=aux;
     end do
  end proc:
> tri_shell:=proc(T,n)  # tri de T[1..n] par la méthode Shell
     local h,i,j,aux;
     h:=iquo(n,2);
     while h>=1 do
        for i from h+1 to n do
          aux:=T[i];
          for j from i-h by -h to 1 while T[j]>aux do
             T[j+h]:=T[j]
          end do;
          T[j+h]:=aux
        od;
        h:=iquo(h,2)
     end do;
  end proc:
> tri_rapide:=proc(T,n)  # tri de T[1..n] par quicksort
     local pivot,s1,s2,T1,T2,i,j,k;
     if n<=1 then return(T) end if;
     pivot:=T[1];
     j:=0;k:=0;
     for i from 2 to n do
         if T[i]<=pivot then
            j:=j+1;T1[j]:=T[i]
         else
            k:=k+1;T2[k]:=T[i]
         end if
     end do;
     tri_rapide(T1,j);
     tri_rapide(T2,k);
```

```
    for i to j do T[i]:=T1[i] end do;
    T[j+1]:=pivot;
    for i to k do T[i+j+1]:=T2[i] end do;
end proc:
> nb:=10000;
```

10000

```
> listedepart:=[seq(rand(),i=1..nb)]:
> for i to nb do T[i]:=listedepart[i]; U[i]:=T[i]; V[i]:=T[i]
end do:
> time(tri_insert(T,nb));
```

56.624

```
> time(tri_shell(U,nb));
```

0.744

```
> time(tri_rapide(V,nb));
```

0.765

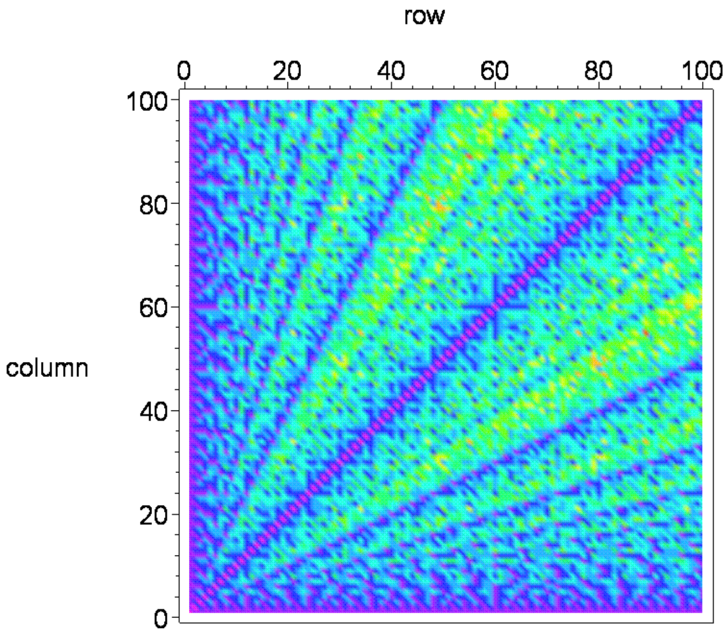
```
> time(sort(listedepart)); # fonction de tri prédéfinie
```

0.010

```
>
```

2. Algorithme d'Euclide

```
> restart:
> dist:=proc(i,j)
    local a,b,r,cpt;
    cpt:=0;
    if i<>j then
      a:=max(i,j);b:=min(i,j);r:=irem(a,b);cpt:=cpt+1;
      while r<>0 do a:=b;b:=r;r:=irem(a,b);cpt:=cpt+1 end do
    end if;
    cpt
end proc:
> n:=100:
> M:=matrix(n,n):
> for i to n do for j to n do M[i,j]:=dist(i,j) end do end do:
> with(plots):
> matrixplot(M,orientation=[-90,0],shading=zhue,style=patchnogr
id,axes=boxed);
```



On pourrait démontrer que les deux raies de complexité maximale ont pour pentes phi et 1/phi, où phi est le nombre d'Or.

```
>
```

3. Méthode du simplexe

3.1 Problème de l'artisan : solution graphique

```
> restart:
> with(plots):
> grille:=implicitplot({seq(X1=i,i=-0..20),seq(X2=j,j=0..15)
  },X1=-1..20,X2=-1..15,grid=[2,2],color=green):
> D1:=implicitplot(X1+3*X2=18,X1=-1..20,X2=-1..15,grid=[2,2]
  ,color=blue,thickness=2):
> D2:=implicitplot(X1+X2=8,X1=-1..20,X2=-1..15,grid=[2,2],co
  lor=tan,thickness=2):
> D3:=implicitplot(2*X1+X2=14,X1=-1..20,X2=-1..15,grid=[2,2]
  ,color=brown,thickness=2):
> Delta:=Z->implicitplot(20*X1+30*X2=Z,X1=-1..20,X2=-1..15,g
  rid=[2,2],color=red,thickness=4):
> G:=seq(display(grille,D1,D2,D3,Delta(2*k),scaling=constrai
```

```
ned),k=0..105):
> display(G,scaling=constrained,insequence=true);
```


```
>
```

3.2 Résolution algébrique

```
> restart:
> eqs1:={x[3]=18-x[1]-3*x[2],x[4]=8-x[1]-x[2],x[5]=14-2*x[1]
-x[2]};
```

$$\{x_3 = 18 - x_1 - 3\,x_2,\ x_4 = 8 - x_1 - x_2,\ x_5 = 14 - 2\,x_1 - x_2\}$$

```
> z1:=20*x[1]+30*x[2];
```

$$20\,x_1 + 30\,x_2$$

```
> eqs2:=solve(eqs1,{x[2],x[4],x[5]});
```

$$\{x_2 = -\frac{1}{3}x_3 + 6 - \frac{1}{3}x_1,\ x_4 = 2 - \frac{2}{3}x_1 + \frac{1}{3}x_3,\ x_5 = 8 - \frac{5}{3}x_1 + \frac{1}{3}x_3\}$$

```
> z2:=simplify(z1,eqs2);
```

$$10\,x_1 + 180 - 10\,x_3$$

```
> eqs3:=solve(eqs2,{x[1],x[2],x[5]});
```

$$\{x_1 = -\frac{3}{2}x_4 + 3 + \frac{1}{2}x_3,\ x_2 = -\frac{1}{2}x_3 + 5 + \frac{1}{2}x_4,\ x_5 = 3 + \frac{5}{2}x_4 - \frac{1}{2}x_3\}$$

```
> z3:=simplify(z2,eqs3);
```

$$210 - 5\,x_3 - 15\,x_4$$

```
>
```

3.3 Méthode des tableaux

```
> restart:
> with(LinearAlgebra):
```

```
> pivotage:=proc(T,r,s)  # pivotage en r,s
      local n,i,pivot,U;
      U:=copy(T);
      n:=RowDimension(U);pivot:=U[r,s];
      for i to n do
        if i<>r then
           U:=RowOperation(U,[i,r],-T[i,s]/pivot)
        end if
      end do;
      U:=RowOperation(U,r,1/pivot);
      U
  end:
> T1:=Matrix([[18,1,3,1,0,0],[8,1,1,0,1,0],[14,2,1,0,0,1],[z
  ,20,30,0,0,0]]);
```

$$T1 := \begin{bmatrix} 18 & 1 & 3 & 1 & 0 & 0 \\ 8 & 1 & 1 & 0 & 1 & 0 \\ 14 & 2 & 1 & 0 & 0 & 1 \\ z & 20 & 30 & 0 & 0 & 0 \end{bmatrix}$$

```
> T2:=pivotage(T1,1,3);
```

$$T2 := \begin{bmatrix} 6 & \frac{1}{3} & 1 & \frac{1}{3} & 0 & 0 \\ 2 & \frac{2}{3} & 0 & \frac{-1}{3} & 1 & 0 \\ 8 & \frac{5}{3} & 0 & \frac{-1}{3} & 0 & 1 \\ z-180 & 10 & 0 & -10 & 0 & 0 \end{bmatrix}$$

```
> T3:=pivotage(T2,2,2);
```

$$T3 := \begin{bmatrix} 5 & 0 & 1 & \frac{1}{2} & \frac{-1}{2} & 0 \\ 3 & 1 & 0 & \frac{-1}{2} & \frac{3}{2} & 0 \\ 3 & 0 & 0 & \frac{1}{2} & \frac{-5}{2} & 1 \\ z-210 & 0 & 0 & -5 & -15 & 0 \end{bmatrix}$$

```
>
```

3.4 Automatisation

```
> simplexe:=proc(tab,n,m)  # n variables, m contraintes
      local h,i,j,r,s,T,sols;
      T:=Copy(tab);
      while true do
         # recherche d'une colonne pivot
         h:=0;
         for j from 2 to n+m+1 do
            if T[m+1,j]>h then h:=T[m+1,j];s:=j fi
         od;
```

```
        if h=0 then  # affichage du résultat
            sols:=NULL;
            for j from 1 to n do
                 if T[m+1,j+1]=0 then

sols:=sols,x[j]=add(T[i,j+1]*T[i,1],i=1..m)
                 else
                     sols:=sols,x[j]=0
                 end if
            end do;
           return sols,`zmax`=-coeff(T[m+1,1],z,0)
        end if;
        # recherche d'une ligne pivot
        h:=infinity;
        for i from 1 to m do
           if T[i,s]>0 and T[i,1]/T[i,s]<h then
h:=T[i,1]/T[i,s];r:=i end if
        end do;
        if h=infinity then return `zmax`=infinity end if;
        T:=pivotage(T,r,s);
     end do
end proc:
> # exemple du cours :
> T:=Matrix([[18,1,3,1,0,0],[8,1,1,0,1,0],[14,2,1,0,0,1],[z,
20,30,0,0,0]]);
```

$$T := \begin{bmatrix} 18 & 1 & 3 & 1 & 0 & 0 \\ 8 & 1 & 1 & 0 & 1 & 0 \\ 14 & 2 & 1 & 0 & 0 & 1 \\ z & 20 & 30 & 0 & 0 & 0 \end{bmatrix}$$

```
> simplexe(T,2,3);
```

$$x_1 = 3, x_2 = 5, zmax = 210$$

```
> # autre exemple :
> tab_alea:=proc(n,m)    # tableau aléatoire pour n
variables, m contraintes
     local M1,M2,M3,M4,M5,M6;
     M1:=Matrix(m,1,rand(50..100));
     M2:=Matrix(m,n,rand(-4..10));
     M3:=DiagonalMatrix([1$m]);
     M4:=<<z>>;
     M5:=Matrix(1,n,rand(-2..8));
     M6:=Matrix(1,m,0);
     Matrix([[M1,M2,M3],[M4,M5,M6]])
end:
> n:=4;m:=7;
```

$$4$$
$$7$$

```
> T:=tab_alea(n,m):
> simplexe(T,n,m);
```

$$x_1 = \frac{2719}{497}, x_2 = 0, x_3 = \frac{1074}{497}, x_4 = \frac{1093}{71}, zmax = \frac{57415}{497}$$

```
> # vérification avec la fonction prédéfinie de Maple :
> with(simplex):
> eqs:=seq(add(T[i,j+1]*x[j],j=1..n)<=T[i,1],i=1..m);
```

$$-4 x_1 + 5 x_2 - 3 x_3 + 6 x_4 \leq 64, -x_1 + 2 x_2 + 8 x_3 + 3 x_4 \leq 58,$$
$$-4 x_1 + x_2 - x_3 - 4 x_4 \leq 61, 2 x_1 - 3 x_2 + 4 x_3 \leq 92, 4 x_1 + 4 x_2 - x_3 - 2 x_4 \leq 74,$$
$$8 x_1 + 10 x_2 - x_3 + x_4 \leq 57, -x_1 - x_2 + 5 x_3 + 3 x_4 \leq 90$$

```
> gain:=add(T[m+1,j+1]*x[j],j=1..n);
```

$$-x_1 + 4 x_2 - x_3 + 8 x_4$$

```
> sol:=maximize(gain,{eqs},NONNEGATIVE);
```

$$\{ x_4 = \frac{1093}{71}, x_3 = \frac{1074}{497}, x_1 = \frac{2719}{497}, x_2 = 0 \}$$

```
> `zmax`=subs(sol,gain);
```

$$zmax = \frac{57415}{497}$$

```
>
```