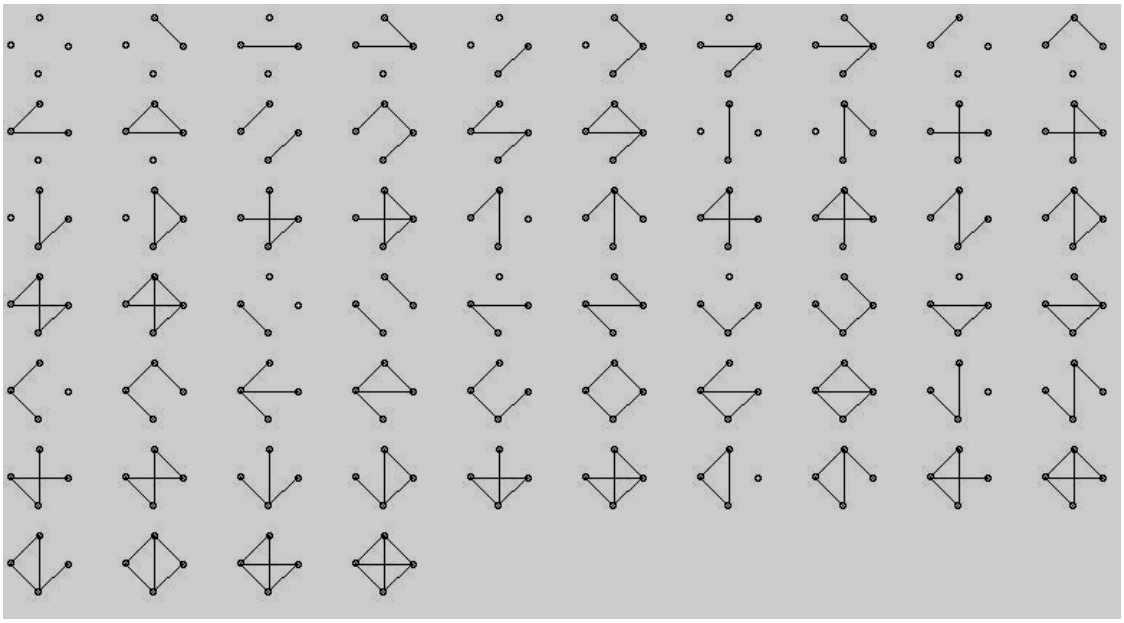
Tous les graphes avec *N* sommets

On se donne *N* points numérotés de 0 à *N* – 1. On veut construire tous les graphes ayant comme sommets ces points. Pour cela il s'agit de placer des arêtes de jonction de toutes les façons possibles.

Le nombre d'arêtes maximal est le nombre de façons de choisir deux points parmi les *N*, soit $C_N^2 = N(N-1)/2$. Par exemple 6 arêtes pour $N = 4$. Chaque graphe est obtenu en prenant une partie de cet ensemble d'arêtes. Le nombre de parties de cet ensemble est $2^{N(N-1)/2}$. C'est aussi le nombre des graphes, par exemple 64 pour $N=4$. Nous allons maintenant les dessiner, comme ci-dessous pour $N = 4$.

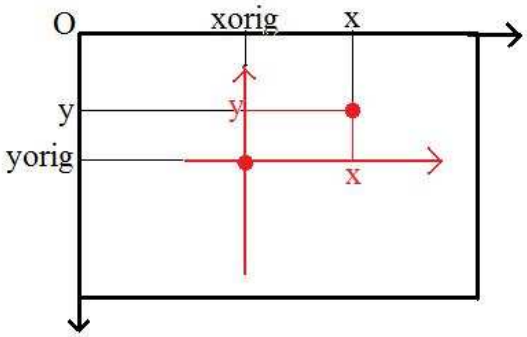


Préparatifs

- dessin des *N* sommets : Ces points vont être placés sur un cercle de centre *xorig*, *yorig*, et de rayon *rayon*, de façon à former un polygone régulier. Le sommet numéro *i* fait un angle de $2\pi\, i / N$ avec l'horizontale. Dans un repère classique d'origine le centre du cercle, les coordonnées du sommet *i* sont :

$x_i = rayon\ \cos(2\pi\, i / N)$ et $y_i = rayon\ \sin(2\pi\, i / N)$. Mais dans le repère écran, cela donne :

$x_i = xorig + rayon\ \cos(2\pi\, i / N)$ et $y_i = yorig - rayon\ \sin(2\pi\, i / N)$.



Ensuite lorsque l'on dessinera les graphes les uns après les autres, il conviendra de changer les coordonnées de l'origine *xorig* et *yorig*.

- Numérotation des arêtes : Une double boucle permet d'avoir toutes les paires de sommets, c'est-à-dire les arêtes, qui sont numérotées au fur et à mesure de 0 à $N(N-1)/2$. Il est aussi indispensable de pouvoir retrouver les extrémités *i* et *j* d'une arête à partir de son numéro, d'où le tableau *arete*[] où l'on place le nombre $N*i + j$. Pour retrouver *i* on fait *arete*[] / *N* et pour *j* on fait *arete*[]%*N*.

- Parties de l'ensemble des arêtes : Chaque graphe prend une partie de l'ensemble des $N(N-1)/2$ arêtes. Les parties sont en bijection avec les nombres en binaire de longueur $N(N-1)/2$. Il suffit de fabriquer ces nombres en binaire grâce aux restes successifs des divisions par 2, et à chaque fois on trouve un graphe. Rappelons que cette méthode des restes successifs écrit les nombres en binaire suivant les poids croissants. Cela commence ainsi :

000000 d'où aucune arête, puis 100000, et l'on dessine le graphe avec la seule arête numéro 0, puis 010000 avec l'arête numéro 1, ensuite 110000 avec les deux arêtes numéro 0 et 1, etc.

Programme C-SDL

```
#include <SDL/SDL.h>
#include <math.h>
#define deuxpi 6.28318
#define N 4
#define rayon 20
void pause(void);
void putpixel(int xe, int ye, Uint32 couleur);
Uint32 getpixel(int xe, int ye);
void cercle( int xo, int yo, int R, Uint32 couleur);
void ligne(int x0,int y0, int x1,int y1, Uint32 c);
SDL_Surface * ecran;
Uint32 blanc,noir;
int arete[N*(N-1)/2],r,xorig,yorig,x[N],y[N];

int main(int argc, char ** argv)
{   int i, j,k,q,x1,x2,y1,y2;
    SDL_Init(SDL_INIT_VIDEO);
    ecran=SDL_SetVideoMode(800,600,32, SDL_HWSURFACE|SDL_DOUBLEBUF);
    blanc=SDL_MapRGB(ecran->format,255,255,255);
```

```
   noir=SDL_MapRGB(ecran->format,0,0,0);
   SDL_FillRect(ecran,0,blanc);
   xorig=rayon+10; yorig=rayon+10;
   k=0;   for(i=0; i<N-1; i++) for(j=i+1;j<N;j++)  arete[k++]=N*i+j;

   for(i=0;i<pow(2,N*(N-1)/2);i++)
    {
     for(j=0;j<N;j++)  /* dessin des noeuds par de petits cercles */
      {  x[j]=xorig+rayon*cos(deuxpi*j/(float)N);
         y[j]=yorig-rayon*sin(deuxpi*j/(float)N);     cercle(x[j],y[j],2,noir);
      }
     q=i;
     for(j=0;j<N*(N-1)/2;j++)
      {  r=q%2;
         if (r==1)   {  x1=x[arete[j]/N]; y1=y[arete[j]/N];  /* dessin d'une arête */
                        x2=x[arete[j]%N];   y2=y[arete[j]%N];
                        ligne(x1,y1,x2,y2,noir);
                     }
        q=q/2;
       }
     xorig+= 2*rayon+40; if (xorig>750) {yorig+=2*rayon+20; xorig=rayon+10;}

     if (yorig>550) {SDL_Flip(ecran); pause();
                     SDL_FillRect(ecran,0,blanc);
                     xorig=10+rayon; yorig=10+rayon;
                     }
  }
  SDL_Flip(ecran);pause(); return 0;
}

void pause(void)
{
   SDL_Event evenement;
   do   SDL_WaitEvent(&evenement);
   while(evenement.type != SDL_QUIT && evenement.type != SDL_KEYDOWN);
}


void putpixel(int xe, int ye, Uint32 couleur)
{ Uint32 * numerocase;
numerocase= (Uint32 *)(ecran->pixels)+xe+ye*ecran->w;    *numerocase=couleur;
}

Uint32 getpixel(int xe, int ye)
{ Uint32 * numerocase;
  numerocase= (Uint32 *)(ecran->pixels)+xe+ye*ecran->w;   return (*numerocase);
}

void cercle( int xo, int yo, int R, Uint32 couleur)
 {
    int x, y, F, F1, F2,newx,newy;
    x=xo; y=yo+R; F=0;
    if (x<800 && x>=0 && y>=0 && y<600) putpixel(x,y,couleur);
    if (x<800 && x>=0 && 2*yo-y>=0 && 2*yo-y<600) putpixel (x,2*yo-y, couleur);
    while( y>yo)
```

```
        {
          F1=F+2*(x-xo)+1; F2=F-2*(y-yo)+1;
          if ( abs(F1)<abs(F2))  { x+=1; F=F1;}
          else {y-=1; F=F2;}
          if (x<800 && x>=0 && y>=0 && y<600) putpixel(x,y,couleur);
           newx=2*xo-x ; newy=2*yo-y ;
          if (x<800 && x>=0 && newy>=0 && newy<600) putpixel(x, newy,couleur);
          if (newx<800 && newx>=0 && y>=0 && y<600) putpixel( newx,y,couleur);
          if (newx<800 && newx>=0 && newy>=0 && newy<600) putpixel(newx,
          newy, couleur);
        }
      if (xo+R<800 && xo+R>=0) putpixel(xo+R,yo,couleur);
      if (xo-R<800 && xo-R>=0) putpixel(xo-R,yo, couleur);
  }

  void ligne(int x0,int y0, int x1,int y1, Uint32 c)
  {
  int dx,dy,x,y,residu,absdx,absdy,pasx,pasy,i;
  dx=x1-x0; dy=y1-y0;  residu=0;       x=x0;y=y0;  putpixel(x,y,c);
  if (dx>0) pasx=1;else pasx=-1; if (dy>0) pasy=1; else pasy=-1;
  absdx=abs(dx);absdy=abs(dy);
  if (dx==0) for(i=0;i<absdy;i++) { y+=pasy;
                                    putpixel(x,y,c); }
  else if(dy==0) for(i=0;i<absdx;i++){ x+=pasx;
                                       putpixel(x,y,c);  }
  else if (absdx==absdy)
    for(i=0;i<absdx;i++) {x+=pasx; y+=pasy;
                          putpixel(x,y,c);
                          }
  else if (absdx>absdy)
         for(i=0;i<absdx;i++)
                          { x+=pasx; residu+=absdy;
                            if(residu >= absdx) {residu -=absdx; y+=pasy;}
                            putpixel(x,y,c);
                          }
  else for(i=0;i<absdy;i++)
            {y+=pasy; residu +=absdx;
             if (residu>=absdy)   {residu -= absdy;x +=pasx;}
             putpixel(x,y,c);
            }
  }
```

Parmi les graphes obtenus, certains sont connexes et d'autres pas. Nous allons nous intéresser aux graphes connexes.

Nombre de graphes non orientés connexes à *N* sommets numérotés

Formule

On a vu que le nombre de graphes non orientés à *N* nœuds (numérotés de 1 à *N*) est : $2^{\binom{N}{2}}$.

Pour savoir combien sont connexes parmi eux, on va soustraire à ce nombre total des graphes ceux qui ne sont pas connexes, en appelant A_N le nombre de graphes connexes provenant de cette différence. Privilégions un nœud, par exemple le nœud 1. Dans un graphe non connexe, il appartient à une composante connexe ayant k nœuds, avec k compris entre 1 et $N - 1$. Comptons le nombre de ces graphes non connexes ayant cette composante connexe de k nœuds, dont le nœud 1. On commence par choisir les $k - 1$ nœuds autres que 1 formant cette composante, parmi les $N - 1$ nœuds autres que 1, soit $\binom{N-1}{k-1}$ cas.

A chaque fois, il existe A_k façons de constituer cette composante connexe. On vient de trouver $\binom{N-1}{k-1} A_k$ façons d'avoir cette composante connexe.

Il reste les $N - k$ nœuds restants, formant un graphe connexe ou non, soit $2^{\binom{N-k}{2}}$ possibilités. Finalement, en répétant cela pour chaque valeur de k, le nombre de graphes connexes est :

$$A_N = 2^{\binom{N}{2}} - \sum_{k=1}^{N-1} \binom{N-1}{k-1} A_k \, 2^{\binom{N-k}{2}}$$

En appliquant la formule classique : $\frac{k}{N}\binom{N}{k} = \binom{N-1}{k-1}$, la formule devient :

$$A_N = 2^{\binom{N}{2}} - \frac{1}{N} \sum_{k=1}^{N-1} k \binom{N}{k} 2^{\binom{N-k}{2}} A_k$$

En ajoutant la condition initiale $A_1 = 1$ à cette relation de récurrence, on peut calculer les premiers termes de la suite (A_N) :

$A_2 = 1$, $A_3 = 4$, $A_4 = 38$, $A_5 = 728$, $A_6 = 26704$, $A_7 = 1\,866\,256$, …

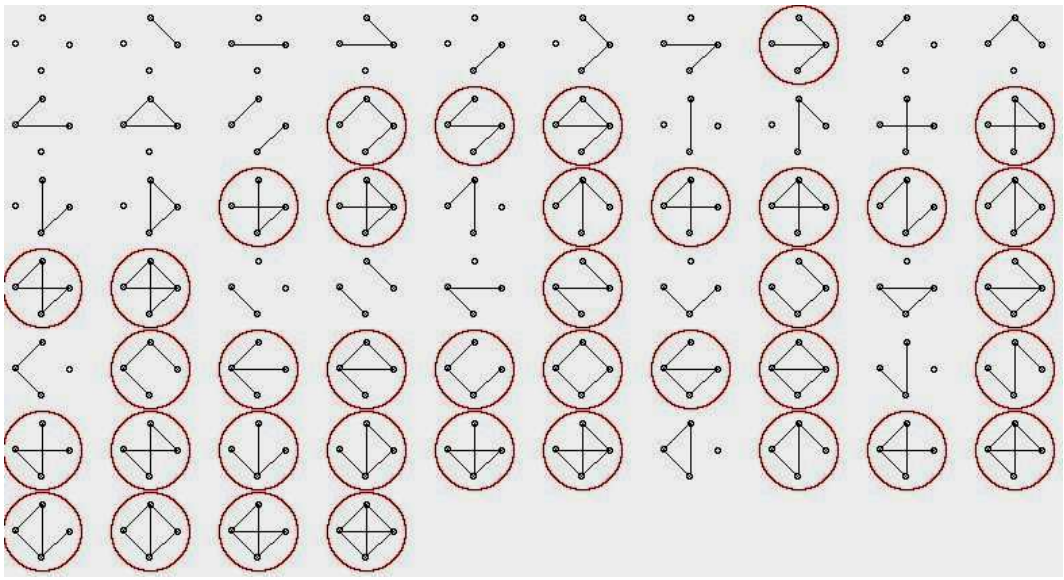
Programme

Le programme suivant trace tous les graphes possédant N nœuds et fait ressortir ceux qui parmi eux sont connexes (ils sont encerclés en rouge). Pour reconnaître ces graphes connexes, on lance une exploration en profondeur de chaque graphe. Si les N nœuds sont atteints, c'est que le graphe est connexe.

```
#include <SDL/SDL.h>
#include <math.h>
#include <stdio.h>
#define deuxpi 6.28318
#define N 6
#define rayon 20
void dessinnoeuds(void);
void explorer(int j);
void pause(void);
void putpixel(int xe, int ye, Uint32 couleur);
```

```
Uint32 getpixel(int xe, int ye);
void cercle( int xo, int yo, int R, Uint32 couleur);
void ligne(int x0,int y0, int x1,int y1, Uint32 c);
SDL_Surface * ecran; Uint32 blanc,noir,rouge;
int arete[N*(N-1)/2],r[N*(N-1)/2],xorig,yorig,x[N],y[N],NA,v[N][N],nbv[N];
int debut[N*(N-1)/2],fin[N*(N-1)/2],dejavu[N],nombrenoeuds,nombreconnexes;

int main(int argc, char ** argv)
{   int i, j,k,q,xdebut,xfin,ydebut,yfin;
    SDL_Init(SDL_INIT_VIDEO);
    ecran=SDL_SetVideoMode(800,600,32, SDL_HWSURFACE|SDL_DOUBLEBUF);
    blanc=SDL_MapRGB(ecran->format,255,255,255);
    noir=SDL_MapRGB(ecran->format,0,0,0);
    rouge=SDL_MapRGB(ecran->format,255,0,0);
    SDL_FillRect(ecran,0,blanc);
    xorig=rayon+10; yorig=rayon+10;
    k=0;   for(i=0;i<N-1;i++) for(j=i+1;j<N;j++)  arete[k++]=N*i+j;
    NA=N*(N-1)/2;
    for(i=0;i<pow(2,NA);i++)
     {
      dessinnoeuds();
      q=i;
      for(j=0;j<NA;j++)
        {  r[j]=q%2;
           if (r[j]==1)
             {  debut[j]=arete[j]/N; fin[j]=arete[j]%N;
                xdebut=x[debut[j]]; ydebut=y[debut[j]]; xfin=x[fin[j]]; yfin=y[fin[j]];
                ligne(xdebut,ydebut,xfin,yfin,noir);
             }
          q=q/2;
        }
      for(k=0;k<N;k++) nbv[k]=0;
      for(k=0;k<NA;k++) if (r[k]==1)
        { v[debut[k]] [nbv[debut[k]]++] = fin[k];
          v[fin[k]][nbv[fin[k]]++] = debut[k];
        }
      nombrenoeuds=0; for(k=0;k<N;k++) dejavu[k]=0;
      explorer(0);
      if (nombrenoeuds==N)
          {cercle(xorig, yorig,rayon+9,rouge);
            nombreconnexes++;
          }

     xorig+= 2*rayon+40; if (xorig>750) {yorig+=2*rayon+20; xorig=rayon+10;}

     if (yorig>550) {SDL_Flip(ecran); pause();
                     SDL_FillRect(ecran,0,blanc);
                     xorig=10+rayon; yorig=10+rayon;
                     }
  }
  SDL_Flip(ecran);pause();
  printf("%d ",nombreconnexes); pause();
  return 0;
}
```

```
void dessinnoeuds(void)
{ int j;
   for(j=0;j<N;j++)
      {  x[j]=xorig+rayon*cos(deuxpi*j/(float)N);
         y[j]=yorig-rayon*sin(deuxpi*j/(float)N);     cercle(x[j],y[j],2,noir);
      }
}

void explorer(int i)
{ int j,voisin,;
  nombrenoeuds++;
  dejavu[i]=1;
  for(j=0;j<nbv[i];j++)
    { voisin=v[i][j];
      if (dejavu[voisin]==0) explorer(voisin);
    }
}
```



Entourés d'un cercle, les graphes connexes pour $N = 4$