Algorithmique — IN102
TD 5

13 janvier 2006

Exercice 1 *Programme complet :*

```
#include <iostream>
using namespace std;

struct graphe{
  int n;
  int **mat;
};

void initGraphe(int nb, graphe &G) {
  G.n = nb;

  // Allocation dynamique d'un tableau à deux dimensions
  G.mat = new int * [nb];
  for(int i=0; i<nb; i++)
    G.mat[i]=new int [nb];
  for(int i=0; i<G.n; i++)
    for(int j=0; j<G.n; j++)
      G.mat[i][j]=0;
}

graphe Fermeture(graphe G) {
  graphe H;

  initGraphe(G.n,H);
  for(int i=0; i<G.n; i++)
    for(int j=0; j<G.n; j++)
      if (i == j)
        H.mat[i][j]=1;
      else
        H.mat[i][j]=G.mat[i][j];
  for(int k=0; k<G.n; k++)
    for(int i=0; i<G.n; i++)
      for(int j=0; j<G.n; j++)
        H.mat[i][j]=H.mat[i][j] || (H.mat[i][k] && H.mat[k][j]);
  return H;
}

void imprime(graphe G) {
  printf("␣␣␣");          // printf effectue des impressions formattées
  for(int i=0;i<G.n;i++) // Impression des indices de colonnes
    printf("%2d␣", i);   // imprimer i comme un entier (%d) sur 2 caractères %2d),
                         // suivi d'un espace
  printf("\n");          // imprimer un retour à la ligne
  for(int i=0; i<G.n; i++) {
```

```
    printf("%2d␣", i); // Indice de ligne
    for(int j=0; j<G.n; j++)
      printf("␣%d␣", G.mat[i][j]);
    printf("\n");
  }
  printf("\n");
}

int main() {
  graphe G,H;

  initGraphe(7,G);
  G.mat[0][1]=1; G.mat[0][3]=1;
  G.mat[1][4]=1;
  G.mat[2][4]=1; G.mat[2][5]=1;
  G.mat[3][1]=1;
  G.mat[4][3]=1;
  G.mat[5][5]=1;

  H=Fermeture(G);

  imprime(G);
  imprime(H);
  exit(0);
}
```

Exercice 2 *Voir solution de l'exercice suivant (fonction AHU).*

Exercice 3 *Programme complet :*

```
#include <iostream>
using namespace std;

// Pour représenter l'absence d'arc
#define infini 10000

int min(int a, int b) {
  if (a==infini) return b;
  if (b==infini) return a;
  if (a<b) return a;
  else return b;
}

int ajoute(int a, int b) {
  if ((a==infini)||(b==infini)) return infini;
  else return a+b;
}

struct graphe {
  int n;
  int **mat;
};
```

```
void initGraphe(int nb, graphe &G) {
  G.n = nb;

  // Allocation dynamique d'un tableau à deux dimensions
  G.mat = new int * [nb];
  for(int i=0; i<nb; i++)
    G.mat[i]=new int [nb];
  for(int i=0; i<G.n; i++)
    for(int j=0; j<G.n; j++)
      G.mat[i][j]=infini;
}


graphe AHU(graphe G) {
  graphe H;

  initGraphe(G.n,H);
  for(int i=0;i<G.n;i++)
    for(int j=0;j<G.n;j++)
      if (i==j)
        H.mat[i][j]=0;
      else
        H.mat[i][j]=G.mat[i][j];

  for(int k=0;k<G.n;k++)
    for(int i=0;i<G.n;i++)
      for(int j=0;j<G.n;j++)
        H.mat[i][j] = min(H.mat[i][j], ajoute(H.mat[i][k], H.mat[k][j]));
  return H;
}

void imprime(graphe G) {
  printf("␣␣␣␣␣");
  for(int i=0;i<G.n;i++)
    printf("%4d␣",i);
  printf("\n");
  for(int i=0;i<G.n;i++) {
    printf("%4d␣",i);
    for(int j=0;j<G.n;j++)
      if (G.mat[i][j]==infini)
        printf("␣␣␣-␣");
      else
        printf("%4d␣",G.mat[i][j]);
    printf("\n");
  }
  printf("\n");
}

int main(){
  graphe G,H;
```

```
  initGraphe(11,G);

#define paris 0
#define chartres 1
#define beauvais 2
#define reims 3
#define evreux 4
#define rouen 5
#define lemans 6
#define nice 7
#define brest 8
#define grenoble 9
#define lyon 10

  G.mat[paris][chartres]=82; G.mat[paris][beauvais]=60; G.mat[paris][reims]=140;
              G.mat[paris][brest]=564; G.mat[paris][grenoble]=576; G.mat[paris][lyon]=472;
  G.mat[chartres][evreux]=77; G.mat[chartres][lemans]=117;
  G.mat[beauvais][rouen]=80;
  G.mat[evreux][rouen]=52;
  G.mat[nice][brest]=1351; G.mat[nice][grenoble]=340; G.mat[nice][lyon]=480;
  G.mat[brest][lyon]=890;
  G.mat[grenoble][lyon]=104;

  for(int i=0;i<11;i++)
    for(int j=0;j<i;j++)
      G.mat[i][j]=G.mat[j][i];

  H=AHU(G);

  imprime(G);
  imprime(H);
  exit(0);
}
```

Exercice 4 *Programme complet :*

```
#include <iostream>
using namespace std;

#define vide 0
#define epsilon 1
#define infini 10000

double max(double a, double b) {
  if ((a==infini) || (b==infini)) return infini;
  if (a>b) return a;
  else return b;
}

double fois(double a, double b) {
  if ((a==infini) || (b==infini)) return infini;
```

```
  else return a*b;
}

double etoile(double a) {
  if (a <= 1) return 1;
  else return infini;
}

struct graphe{
  int n;
  double **mat;
};

void initGraphe(int nb, graphe &G) {
  G.n = nb;

  // Allocation dynamique d'un tableau à deux dimensions
  G.mat = new double * [nb];
  for(int i=0; i<nb; i++)
    G.mat[i]=new double [nb];
  for(int i=0; i<G.n; i++)
    for(int j=0; j<G.n; j++)
      G.mat[i][j]=vide;
}

graphe AHU(graphe G) {
  graphe H;

  initGraphe(G.n,H);
  for(int i=0;i<G.n;i++)
    for(int j=0;j<G.n;j++) {
      H.mat[i][j]=G.mat[i][j];
      if (i==j)
        H.mat[i][j]=max(epsilon,H.mat[i][j]);
    }
  for(int k=0;k<G.n;k++)
    for(int i=0;i<G.n;i++)
      for(int j=0;j<G.n;j++)
        H.mat[i][j]=max(H.mat[i][j], fois(fois(H.mat[i][k],etoile(H.mat[k][k])), H.mat[k][j]));
  return H;
}

void imprime(graphe G) {
  printf("␣␣");
  for(int i=0;i<G.n;i++)
    printf("␣%2d␣",i);
  printf("\n");
  for(int i=0;i<G.n;i++) {
    printf("%2d␣",i);
    for(int j=0;j<G.n;j++)
      if (G.mat[i][j]==infini)
```

```
        printf("*****␣");
      else
        printf("%5.2f␣",G.mat[i][j]); // Imprime un décimal (%f):
                                      //      partie entière sur 5 chiffres , et 2 décimales
    printf("\n");
  }
  printf("\n");
}


int main() {
  graphe G,H;

  initGraphe(4,G);

#define euro 0
#define dollar 1
#define livre 2
#define suisse 3

  G.mat[euro][euro]=1.0; G.mat[dollar][dollar]=1.0; G.mat[livre][livre]=1.0; G.mat[suisse][suisse]=1.0;

  G.mat[euro][dollar]=1.28; G.mat[euro][livre]=0.68;    G.mat[euro][suisse]=1.55;
  G.mat[dollar][euro]=0.78; G.mat[dollar][livre]=0.53; G.mat[dollar][suisse]=1.21;
  G.mat[livre][euro]=1.46;  G.mat[livre][dollar]=1.87; G.mat[livre][suisse]=2.27;
  G.mat[suisse][euro]=0.64; G.mat[suisse][dollar]=0.82; G.mat[suisse][livre]=0.43;

  for(int i=0;i<4;i++)
    for(int j=0;j<4;j++)
      printf("%f␣",G.mat[i][j]*G.mat[j][i]); // Imprime un décimal (%f)
  printf("\n");

  H=AHU(G);

  imprime(G);
  imprime(H);

  // On introduit volontairement une erreur
  G.mat[livre][suisse]=2.30;

  H=AHU(G);

  imprime(H);
  exit(0);
}
```