Algorithmes pour les graphes - TD sur plateforme
Exercice 2 : Calcul des longueurs de tous les circuits

Il s'agit maintenant de compléter la procédure `permut` de l'exercice 1 pour afficher la longueur de chaque circuit. Les distances entre sommets sont données par un tableau à deux dimensions `cout` qui est déclaré en variable globale : pour tout couple de sommets `(i,j)`, `cout[i][j]` est égal à la distance pour aller du sommet `i` au sommet `j`.

La longueur de chaque circuit sera calculée incrémentalement. Pour cela, vous ajouterez à la procédure `permut` un paramètre contenant la longueur du chemin correspondant à `vus[0..nbVus]`.

Votre travail : Vous devez implémenter la procédure `permut` :

```
void permut(int vus[], int nbVus, int nonVus[], int nbNonVus, int longueur)
```

telle que :
- le tableau `vus[0..nbVus-1]` contient les sommets de la liste vus,
- le tableau `nonVus[0..nbNonVus-1]` contient les sommets de l'ensemble $nonVus$,
- la variable `longueur` contient la longueur du chemin correspondant à `vus[0..nbVus-1]`.

Cette procédure affiche la longueur de chaque séquence de sommets commençant par `vus[0.. nbVus-1]`, et se terminant par n'importe quelle permutation de `nonVus[0..nbNonVus-1]`.

Code fourni (téléchargeable sur `http://liris.cnrs.fr/csolnon/TPTSP/code2.c`) : Procédure `main`, qui appelle votre procédure `permut`, et procédure `creeCout`, qui initialise la variable globale `cout` définissant les coûts des arcs.

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

int** cout;

void creeCout(int nbSommets){
    int coutMin = 10;
    int coutMax = 40;
    int i, j, iseed, it;
    iseed = 1;
    cout = (int**) malloc(nbSommets*sizeof(int*));
    for (i=0; i<nbSommets; i++){
        cout[i] = (int*) malloc(nbSommets*sizeof(int));
        for (j=0; j<nbSommets; j++){
            if (i == j) cout[i][j] = coutMax+1;
            else {
                it = 16807 * (iseed % 127773) - 2836 * (iseed / 127773);
                if (it > 0) iseed = it;
                else iseed = 2147483647 + it;
                cout[i][j] = coutMin + iseed % (coutMax-coutMin+1);
            }
        }
    }
}

void permut(int vus[], int nbVus, int nonVus[], int nbNonVus, int longueur){
    /*
    Variable globale :
     - pour tout couple de sommets (i,j), cout[i][j] = cout pour aller de i vers j
    Entree :
     - nonVus[0..nbNonVus-1] = sommets non visites
     - vus[0..nbVus-1] = sommets visites
    Precondition :
     - nbVus > 0 et Vus[0] = 0 (on commence toujours par le sommet 0)
     - longueur = somme des couts des arcs du chemin <vus[0], vus[1], ..., vus[nbVus-1]>
    Postcondition : affiche les longueurs de tous les circuits commençant par vus[0..nbVus-1]
                    et se terminant par les sommets de nonVus[0..nbNonVus-1]
```

```
                    (dans tous les ordres possibles), suivis de 0
     */
    // INSEREZ VOTRE CODE ICI !
}

int main(){
    int nbSommets, i;
    scanf("%d",&nbSommets);
    int vus[nbSommets];
    int nonVus[nbSommets-1];
    creeCout(nbSommets);
    for (i=0; i<nbSommets-1; i++)
        nonVus[i] = i+1;
    vus[0] = 0;
    permut(vus,1,nonVus,nbSommets-1,0);
    return 0;
}
```

Exemple d'exécution : Si l'entier $nbSommets$ saisi en entrée est 4, alors le programme affichera les longueurs des 6 permutations possibles (l'ordre d'affichage des longueurs peut varier) :

```
72
93
119
104
111
95
```