Algorithmes pour les graphes - TD sur plateforme
Exercice 3 : Calcul de la longueur du plus court circuit

Il s'agit de reprendre le code de la procédure `permut` de l'exercice 2, pour calculer la longueur du plus court circuit. Pour cela, vous ajouterez à `permut` un paramètre `pcc` contenant la longueur du plus court circuit trouvé depuis le début de la recherche, et `permut` retournera en sortie la longueur du plus court circuit trouvé à la fin de son exécution.

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

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

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]`,
- la variable `pcc` contient la longueur du plus court circuit trouvé depuis le début.

Soit `C` l'ensemble des circuits commençant par `Vus[0..nbVus-1]` et visitant ensuite chaque sommet de `nonVus[0..nbNonVus-1]` puis retournant en 0. S'il existe un circuit de `C` de longueur inférieure à `pcc`, alors la fonction `permut` retourne la longueur du plus petit circuit de `C`, sinon elle retourne `pcc`.

Code fourni (téléchargeable sur `http://liris.cnrs.fr/csolnon/TPTSP/code3.c`) : Procédure `main` appelant votre procédure `permut`, et procédure `creeCout` initialisant la variable globale `cout` définissant les coûts des arcs.

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.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);
            }
        }
    }
}

int permut(int vus[], int nbVus, int nonVus[], int nbNonVus, int longueur, int pcc){
    /*
     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]>
```

```
     - pcc = longueur du plus court circuit trouve depuis le premier appel à permut
     Postcondition : Soit C l'ensemble des circuits commencant par Vus[0..nbVus-1] et visitant
                     ensuite chaque sommet de nonVus[0..nbNonVus-1] puis retournant en 0.
                     S'il existe un circuit de C de longueur inferieure a borne, alors retourne la
                     longueur du plus petit circuit de C, sinon retourne borne
     */
    // INSEREZ VOTRE CODE ICI !
}

int main(){
    int nbSommets, i, pcc;
    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;
    pcc = permut(vus,1,nonVus,nbSommets-1,0,INT_MAX);
    printf("%d\n",pcc);
    return 0;
}
```

Exemple d'exécution : Les temps CPU (en secondes) sont donnés à titre indicatif, pour un processeur 2,6 GHz Intel Core i5.

Entrée	Sortie	Temps CPU
4	72	0.00
6	91	0.00
8	123	0.00
10	134	0.01
12	161	0.82
14	182	128.30