Croquefer
Gaëtan
Groupe 2B
Année 2018/2019

Compte rendu du TP n°1 de « Graphes »

I)Problématique

Lorsque que l'on travaille sur des systèmes qui sont amenés à travailler en parallèle mais peuvent interférer les uns avec les autres, il est important de discerner les relations qui les unissent, et ce afin d'utiliser ces derniers de manière optimale.

Dans le but de vérifier quels systèmes peuvent sans crainte être utilisé en parallèle, l'approche théorique consistera à ramener le problème rencontré à un problème de coloration de graphe.

On représentera donc notre système par le biais d'un graphe non-orienté G=(S,A) où :

- chaque sommet s ∈ S représente un des systèmes
- chaque arrête a ∈ A représente une connexion entre deux nœuds

Notre but sera donc désormais de décomposer notre graphe en plusieurs stables qui seront composés des différents systèmes capable de travailler en parallèle sans interférer.

Pour cela, on utilisera l'algorithme de Welsh-Powell.

Afin de résoudre le problème le plus efficacement possible, j'ai choisi d'implémenter ledit algorithme en C++ afin de pouvoir gérer la mémoire de la manière la plus optimisée possible. De surcroît, pour trier les dates de fin après la première étape de l'algorithme, j'ai opté pour un algorithme de tri à bulle, d'une complexité de n^2, et ce par facilité d'implémentation.

II)Réalisation

Comme dans le TP précédent, j'ai opté pour une représentation par matrice. Cette matrice est encore une fois un attribut de ma classe/mon TAD « Graphe », et est complétée par un attribut de type entier correspondant à sa taille.

```
class Graphe
{
  private:
    int _nbSommet;
    std::vector<std::vector<bool> > _matrixLiens;
```

La classe « Graphe », comporte en outre des constructeurs permettant d'initialiser la matrice, une fonction permettant d'ajouter les arcs orienté entre les sommets.

```
Graphe();
Graphe(int nbSommet);
void ajouterLien(int s1, int s2);
```

Enfin, j'ai écris deux fonctions pour l'affichage, la première réalisant un affichage textuel de la matrice :

```
friend std::ostream& operator << (std::ostream& os, const Graphe& g);
```

Et la seconde permettant, uniquement sur linux et via l'outils « Graphviz », d'afficher graphiquement le graphe. Notez que dans le « main » de démonstration fourni avec ce compte rendu, l'appel vers cette fonction est commenté, afin de prévenir d'éventuelles erreurs de compilation/execution.

```
void afficherGraphe();
```

Pour finir, la coloration du graphe est réalisé par la fonction suivante :

```
std::vector<std::vector<int> > getStables();
```

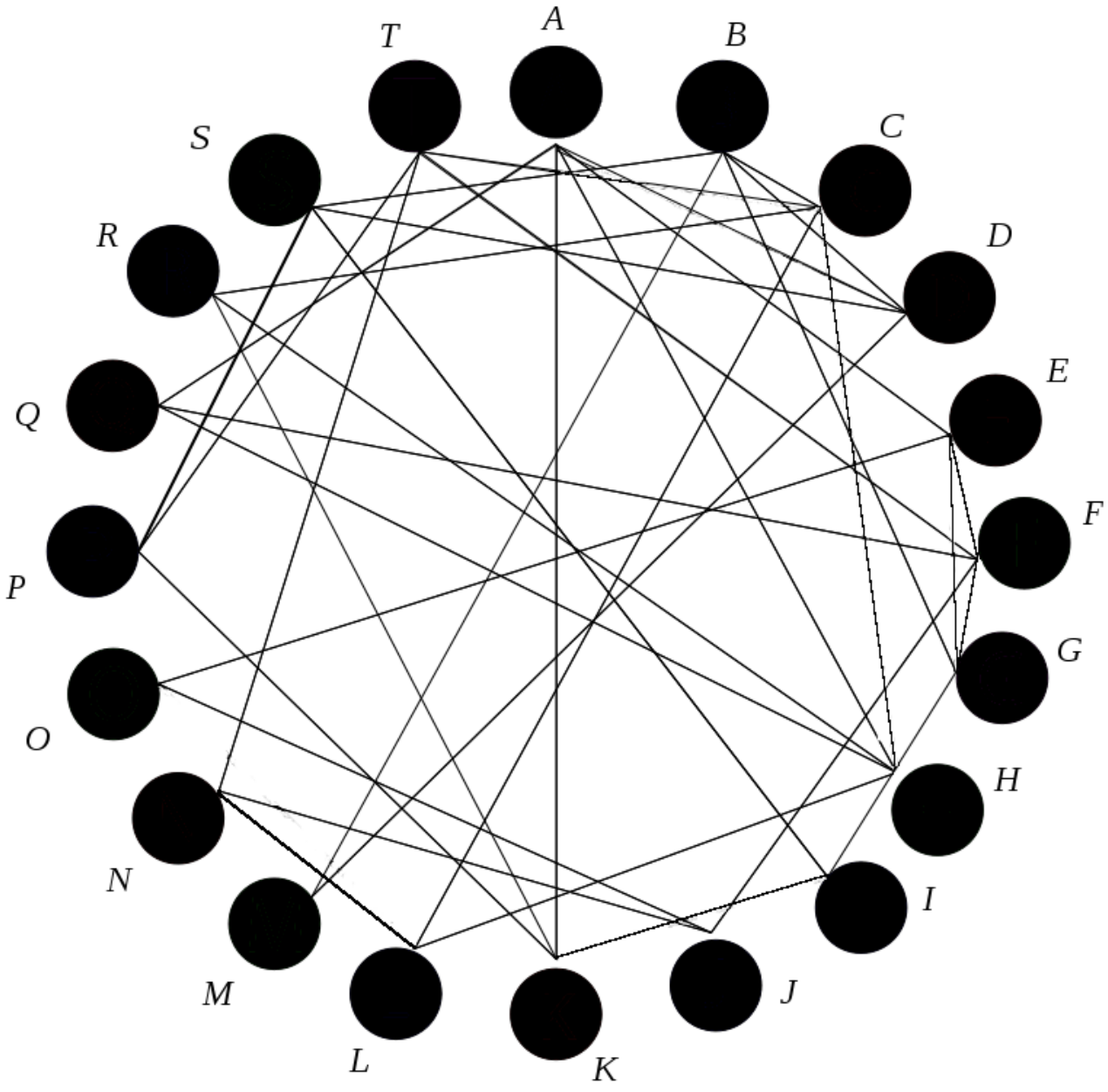
Qui utilise l'utilitaire suivant, afin de modulariser le code :

```
bool contains(std::vector<int> tab, int e);
```

Cet utilitaire permet de vérifier si un élément donné appartient à un vecteur précis.

Pour plus de détail sur l'algorithme, vous trouverez en annexe le code des fonctions.

Voici le graphe G1 correspondant au fréquences d’un réseau mobile, obtenu à partir du tableau donné en exercice :



Voici le graphe G2 correspondant au cycle d'assemblage d'un smartphone, obtenu à partir du tableau donné en exercice :

```
===NOUVEAU STABLE===


j :1    k :1
1 n'est pas dans le stable

j :2    k :1
2 est dans le stable

j :3    k :1    k :2
3 n'est pas dans le stable

j :4    k :1    k :2
4 n'est pas dans le stable

j :5    k :1    k :2
5 est dans le stable

j :6    k :1    k :2    k :5
6 n'est pas dans le stable

j :7    k :1    k :2    k :5
7 n'est pas dans le stable

j :8    k :1    k :2    k :5
8 n'est pas dans le stable

j :9    k :1    k :2    k :5
9 est dans le stable

j :10   k :1    k :2    k :5    k :9
10 est dans le stable

j :11   k :1    k :2    k :5    k :9    k :10
11 n'est pas dans le stable

j :12   k :1    k :2    k :5    k :9    k :10
12 est dans le stable

j :13   k :1    k :2    k :5    k :9    k :10   k :12
13 n'est pas dans le stable

j :14   k :1    k :2    k :5    k :9    k :10   k :12
14 n'est pas dans le stable

j :15   k :1    k :2    k :5    k :9    k :10   k :12
15 n'est pas dans le stable

j :16   k :1    k :2    k :5    k :9    k :10   k :12
16 est dans le stable

j :17   k :1    k :2    k :5    k :9    k :10   k :12   k :16
17 n'est pas dans le stable

j :18   k :1    k :2    k :5    k :9    k :10   k :12   k :16
18 est dans le stable

j :19   k :1    k :2    k :5    k :9    k :10   k :12   k :16   k :18
19 n'est pas dans le stable

j :20   k :1    k :2    k :5    k :9    k :10   k :12   k :16   k :18
20 n'est pas dans le stable


stable: 1 2 5 9 10 12 16 18
```

```
===NOUVEAU STABLE===


j :1    k :17
1 n'est pas dans le stable

j :2    k :17
2 n'est pas dans le stable

j :3    k :17
3 n'est pas dans le stable

j :4    k :17
4 n'est pas dans le stable

j :5    k :17
5 n'est pas dans le stable

j :6    k :17
6 n'est pas dans le stable

j :7    k :17
7 n'est pas dans le stable

j :8    k :17
8 n'est pas dans le stable

j :9    k :17
9 n'est pas dans le stable

j :10   k :17
10 n'est pas dans le stable

j :11   k :17
11 n'est pas dans le stable

j :12   k :17
12 n'est pas dans le stable

j :13   k :17
13 n'est pas dans le stable

j :14   k :17
14 n'est pas dans le stable

j :15   k :17
15 n'est pas dans le stable

j :16   k :17
16 n'est pas dans le stable

j :17   k :17
17 n'est pas dans le stable

j :18   k :17
18 n'est pas dans le stable

j :19   k :17
19 n'est pas dans le stable

j :20   k :17
20 n'est pas dans le stable


stable: 17
```

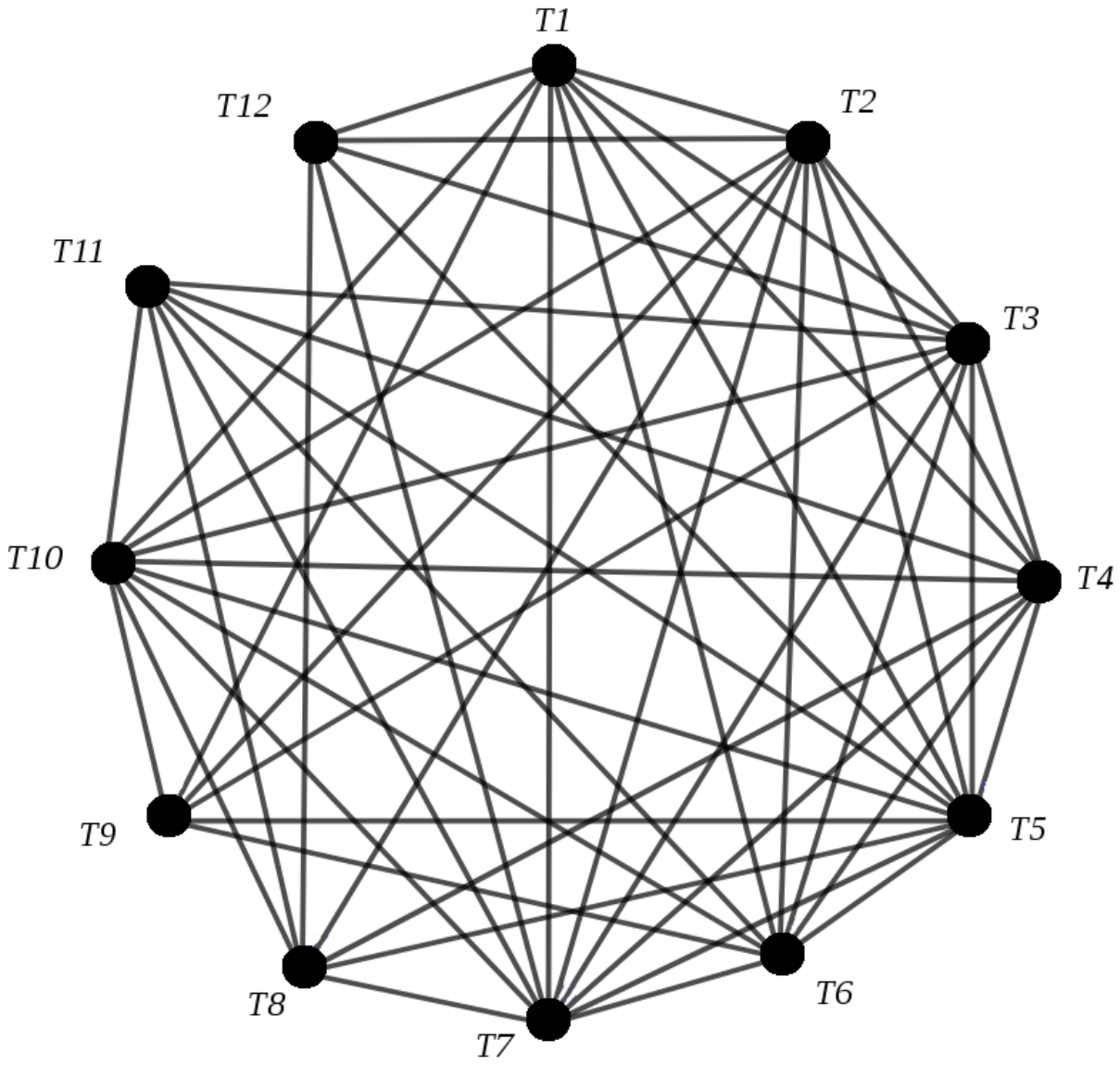
```
===NOUVEAU STABLE===


j :1    k :3
1 n'est pas dans le stable

j :2    k :3
2 n'est pas dans le stable

j :3    k :3
3 n'est pas dans le stable

j :4    k :3
4 est dans le stable

j :5    k :3    k :4
5 n'est pas dans le stable

j :6    k :3    k :4
6 est dans le stable

j :7    k :3    k :4    k :6
7 n'est pas dans le stable

j :8    k :3    k :4    k :6
8 n'est pas dans le stable

j :9    k :3    k :4    k :6
9 n'est pas dans le stable

j :10   k :3    k :4    k :6
10 n'est pas dans le stable

j :11   k :3    k :4    k :6
11 est dans le stable

j :12   k :3    k :4    k :6    k :11
12 n'est pas dans le stable

j :13   k :3    k :4    k :6    k :11
13 n'est pas dans le stable

j :14   k :3    k :4    k :6    k :11
14 est dans le stable

j :15   k :3    k :4    k :6    k :11   k :14
15 est dans le stable

j :16   k :3    k :4    k :6    k :11   k :14   k :15
16 n'est pas dans le stable

j :17   k :3    k :4    k :6    k :11   k :14   k :15
17 n'est pas dans le stable

j :18   k :3    k :4    k :6    k :11   k :14   k :15
18 n'est pas dans le stable

j :19   k :3    k :4    k :6    k :11   k :14   k :15
19 n'est pas dans le stable

j :20   k :3    k :4    k :6    k :11   k :14   k :15
20 n'est pas dans le stable


stable: 3 4 6 11 14 15
```

```
===NOUVEAU STABLE===


j :1    k :7
1 n'est pas dans le stable

j :2    k :7
2 n'est pas dans le stable

j :3    k :7
3 n'est pas dans le stable

j :4    k :7
4 n'est pas dans le stable

j :5    k :7
5 n'est pas dans le stable

j :6    k :7
6 n'est pas dans le stable

j :7    k :7
7 n'est pas dans le stable

j :8    k :7
8 est dans le stable

j :9    k :7    k :8
9 n'est pas dans le stable

j :10   k :7    k :8
10 n'est pas dans le stable

j :11   k :7    k :8
11 n'est pas dans le stable

j :12   k :7    k :8
12 n'est pas dans le stable

j :13   k :7    k :8
13 est dans le stable

j :14   k :7    k :8    k :13
14 n'est pas dans le stable

j :15   k :7    k :8    k :13
15 n'est pas dans le stable

j :16   k :7    k :8    k :13
16 n'est pas dans le stable

j :17   k :7    k :8    k :13
17 n'est pas dans le stable

j :18   k :7    k :8    k :13
18 n'est pas dans le stable

j :19   k :7    k :8    k :13
19 est dans le stable

j :20   k :7    k :8    k :13   k :19
20 est dans le stable


stable: 7 8 13 19 20
```

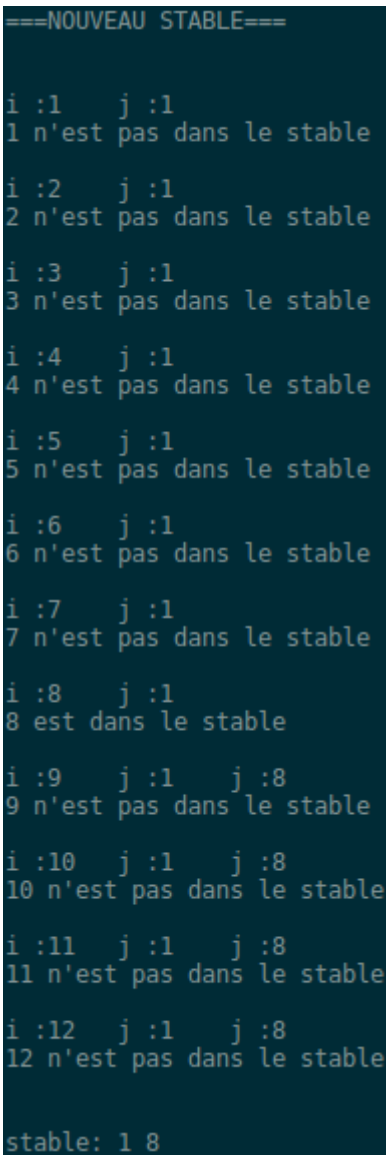
Enfin, voici le résultat retourné :

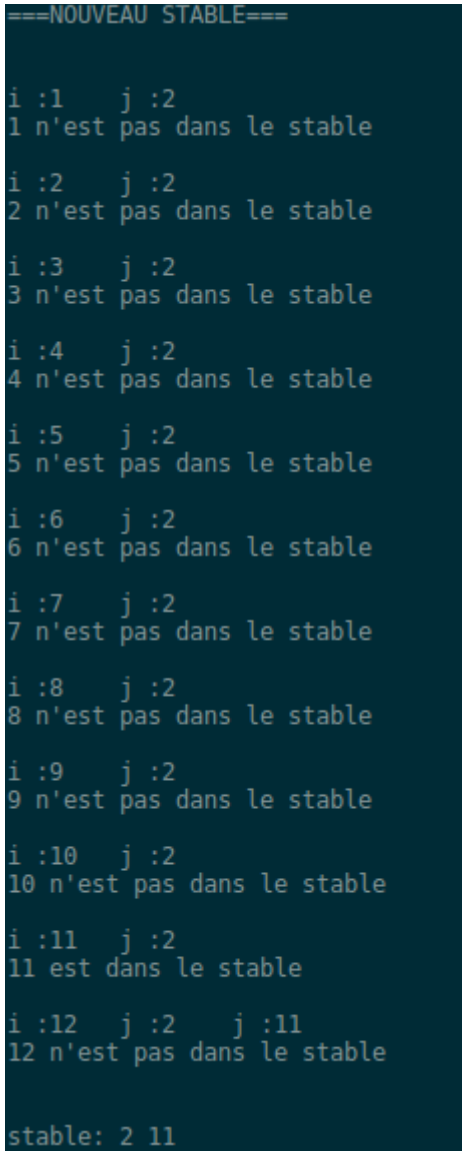
```
Stables
1 2 5 9 10 12 16 18
3 4 6 11 14 15
7 8 13 19 20
17
```

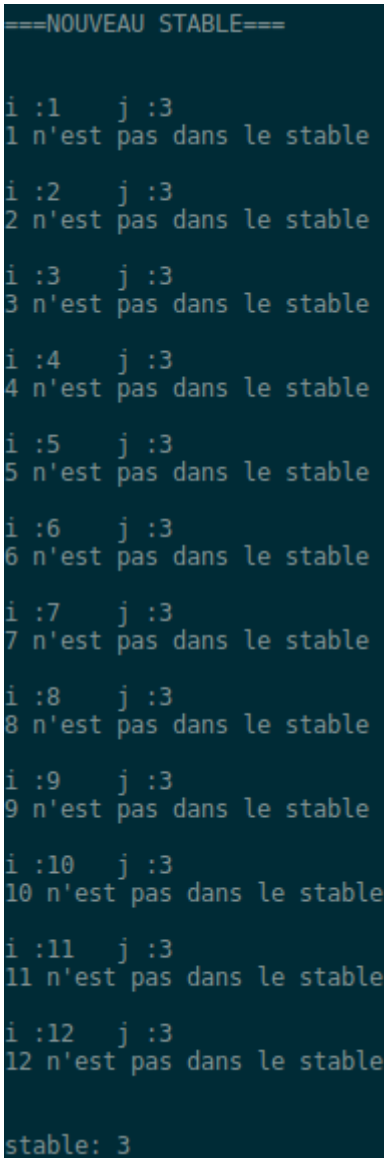
Voici le graphe G2 correspondant au cycle d'assemblage d'un smartphone, obtenu à partir du tableau donné en exercice :



Et la trace de mon algorithme sur ce graphe :



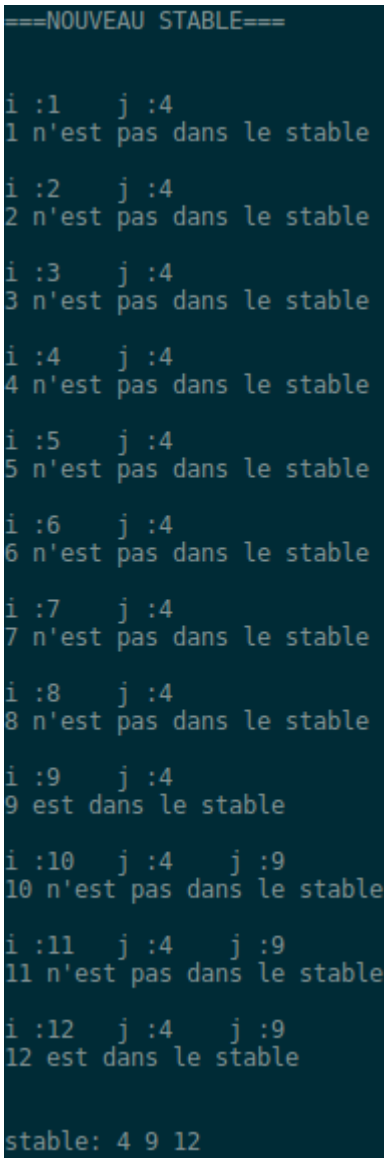


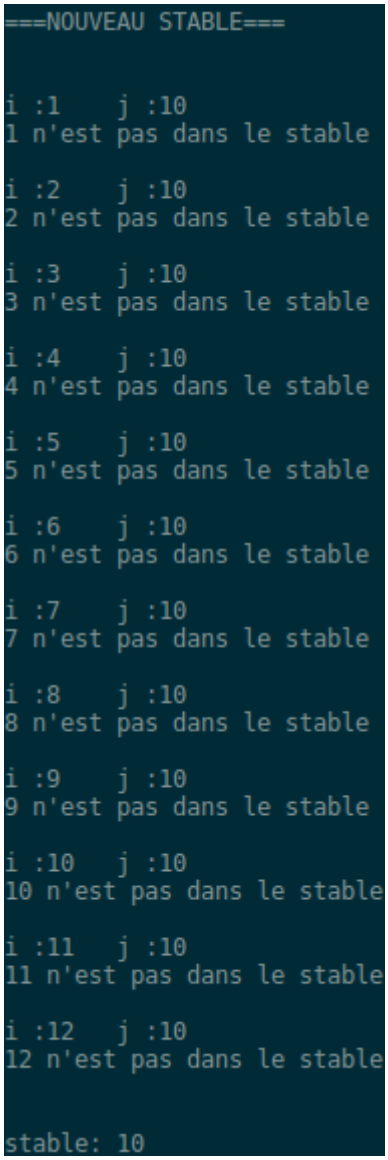


```
===NOUVEAU STABLE===


i :1    j :7
1 n'est pas dans le stable

i :2    j :7
2 n'est pas dans le stable

i :3    j :7
3 n'est pas dans le stable

i :4    j :7
4 n'est pas dans le stable

i :5    j :7
5 n'est pas dans le stable

i :6    j :7
6 n'est pas dans le stable

i :7    j :7
7 n'est pas dans le stable

i :8    j :7
8 n'est pas dans le stable

i :9    j :7
9 n'est pas dans le stable

i :10   j :7
10 n'est pas dans le stable

i :11   j :7
11 n'est pas dans le stable

i :12   j :7
12 n'est pas dans le stable


stable: 7
```

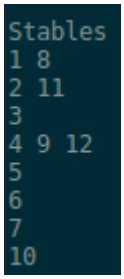
```
===NOUVEAU STABLE===


i :1    j :5
1 n'est pas dans le stable

i :2    j :5
2 n'est pas dans le stable

i :3    j :5
3 n'est pas dans le stable

i :4    j :5
4 n'est pas dans le stable

i :5    j :5
5 n'est pas dans le stable

i :6    j :5
6 n'est pas dans le stable

i :7    j :5
7 n'est pas dans le stable

i :8    j :5
8 n'est pas dans le stable

i :9    j :5
9 n'est pas dans le stable

i :10   j :5
10 n'est pas dans le stable

i :11   j :5
11 n'est pas dans le stable

i :12   j :5
12 n'est pas dans le stable


stable: 5
```

```
===NOUVEAU STABLE===


i :1    j :6
1 n'est pas dans le stable

i :2    j :6
2 n'est pas dans le stable

i :3    j :6
3 n'est pas dans le stable

i :4    j :6
4 n'est pas dans le stable

i :5    j :6
5 n'est pas dans le stable

i :6    j :6
6 n'est pas dans le stable

i :7    j :6
7 n'est pas dans le stable

i :8    j :6
8 n'est pas dans le stable

i :9    j :6
9 n'est pas dans le stable

i :10   j :6
10 n'est pas dans le stable

i :11   j :6
11 n'est pas dans le stable

i :12   j :6
12 n'est pas dans le stable


stable: 6
```





Enfin, voici le résultat retourné :



III)Bilan

Lors de ce TP sur la coloration des graphes, j'ai pu appliquer à des cas concrets l'algorithme de Welsh-Powell que j'avais déjà appris à utiliser dans des cadres théoriques.

Je ne pensais pas que ce simple problème de graphe avait une utilité si tangible, et notamment dans le champ de l'informatique. En effet, l'exercice 2 sur le cycle d'assemblage fait directement écho à la programmation en parallèle que Mr. Cariou avait rapidement évoqué lors des cours d'architecture des ordinateurs : de la même manière que les robots de travail sont partagés pour certaines tâches, les ressources de l'ordinateur sont partagées entre tous les threads en cours exécution. J'espère que les applications de la théorie des graphes qui me reste à découvrir seront tous aussi intéressantes.

Annexe

```
vector<vector<int> > Graphe::getStables()
{
  vector<vector<int> > result;
  vector<int> sommetsParcourus;
  vector<int> stable;

  while((int)sommetsParcourus.size()!=this->_nbSommet)
  {
    //DEBUG cout<<"\n===NOUVEAU STABLE==="<<endl<<endl;
    for(int i=0; i<this->_nbSommet; i++)
    {
      if(!contains(sommetsParcourus, i))
      {
        sommetsParcourus.push_back(i);
        stable.push_back(i);
        break;
      }
    }

    for(int i=0; i<this->_nbSommet; i++)
    {
      int count=0;
      //DEBUG cout<<"\ni :"<<i+1<<"\t";

      for(int j=0; j<(int)stable.size(); j++)
      {
        //DEBUG cout<<"j :"<<stable[j]+1<<"\t";
        if(!this->_matrixLiens[stable[j]][i] && stable[j]!=i)
        {
          if(!this->_matrixLiens[i][stable[j]])
          {
            count++;
          }
        }
      }
      //DEBUG cout<<endl;
      if(count==(int)stable.size())
      {
        if(!contains(sommetsParcourus, i))
        {
          //DEBUG cout<<i+1<<" est dans le stable"<<endl;
          sommetsParcourus.push_back(i);
          stable.push_back(i);
        }
```

```
        /*DEBUG else
        {
          cout<<i+1<<" n'est pas dans le stable"<<endl;
        }*/
      }
      /*DEBUG else
      {
        cout<<i+1<<" n'est pas dans le stable"<<endl;
      }*/

    }
    result.push_back(stable);
    /*DEBUG cout<<"\n\nstable: ";
    for(int i=0; i<stable.size(); i++)
    {
      cout<<stable[i]+1<<" ";
    }
    cout<<endl<<endl;*/
    stable.clear();
  }

  return result;
}
```