TP D'OPTION INFORMATIQUE 5
Couplage maximum dans un graphe biparti

L'objet de ce TP est d'implémenter la recherche d'un couplage maximum dans un graphe biparti à l'aide de chemins augmentants.

1. Rappeler la différence entre un couplage maximal et un couplage maximum, et donner un exemple illustrant cette différence.
2. On définit le type suivant pour représenter un graphe biparti :

```
type biparti = {
    sep : int;
    adj : int list array
};;
```

   où l'ensemble des n sommets est partitionné entre $[\![0, \texttt{sep}]\!]$ et $[\![\texttt{sep}+1, n]\!]$, et où `adj` contient le tableau des listes d'adjacences du graphe.
   Représenter au papier le graphe g_1 défini comme :

```
let g1 = {
  sep = 3;
  adj = [|
    [4; 5];
    [4];
    [5; 6];
    [6; 7];
    [0; 1];
    [0; 2];
    [2; 3];
    [3]
  |] };;
```

3. On rappelle l'algorithme de calcul d'un couplage maximum :

```
Fonction chemin_augmentant(G, C):
    soient U,V les deux parties du graphe biparti G
    les pères sont indéfinis
    Fonction visiter(u):
        on marque u
        pour chaque voisin non marqué v de u
            on marque v
            le père de v devient u
            si v n'est pas couvert par C
                on arrête chemin_augmentant
                en renvoyant le chemin décrit par les pères depuis v
            sinon
                soit w couplé à v dans C
                si w n'est pas marqué
                    le père de w devient v
                    visiter(w)
    pour chaque sommet u de U
        si u n'est pas couvert par C
            on efface les marques
            visiter(u)
    Il n'existe pas de chemin augmentant
```

```
Fonction couplage_max(G):
    soit C le couplage vide
    tant que C a un chemin augmentant dans G :
        on augmente C le long de ce chemin
    on renvoie C
```

Appliquer au papier cet algorithme sur le graphe g_1, en listant chacun des couplages successivement obtenus.

4. On choisit de représenter le tableau des pères, ou prédécesseurs par un `int option array`, `None` correspondant à un père indéfini.
   Écrire une fonction `pere_vers_chemin : int option array -> int -> int list` prenant en entrée un tel tableau, et un sommet v, et renvoyant le chemin allant de v à la racine de son arbre encodé dans ce tableau. Par exemple :
   `pere_vers_chemin [|Some 4; None; None; None; Some 1; Some 0; None|] 5` doit renvoyer `[5; 0; 4; 1]`.
5. On représente également un couplage par un `int option array` indiquant pour chaque sommet à quel sommet il est couplé.
   (a) Rappeler la définition d'un chemin augmentant pour un couplage.
   (b) Quelle est la parité du nombre de sommets dans un chemin augmentant ?
   (c) Écrire une fonction `augmenter_couplage : int list -> int option array -> unit` augmentant un couplage le long d'un chemin augmentant.
6. Écrire une fonction `chemin_augmentant : biparti -> int option array -> int list option` prenant en entrée un graphe biparti et un couplage, et renvoyant `Some c` où `c` est un chemin augmentant, s'il en existe un, ou `None` sinon. On pourra s'inspirer de la méthode vue pour interrompre une recherche dans le cadre d'un retour sur trace.
7. Écrire une fonction `couplage_max : biparti -> int option array` prenant en entrée un graphe biparti et en renvoyant un couplage maximum.
8. Déterminer la complexité de la fonction précédente.
9. Soit G un graphe biparti de parties U et V. On dit qu'un ensemble de sommets T est un **transversal** de G si toute arête de G a une extrémité dans T. Un **transversal minimum** de G est un transversal de cardinal minimum.
   Le **théorème de König** énonce que pour un graphe biparti, les cardinaux d'un transversal minimum et d'un couplage maximum sont égaux. On peut le montrer avec la construction suivante, calculant un transversal minimum à partir d'un couplage maximum C :
   - On appelle **chemin alternant** un chemin dont les arêtes alternent entre $\overline{C}$ et C.
   - Soit Z l'ensemble des sommets de G accessibles par un chemin alternant depuis un sommet non couplé de U.
   - Un transversal minimum de G est alors $T = (U \setminus Z) \cup (V \cap Z)$.

   (a) Appliquer cette construction pour obtenir un transversal minimum de g_2 :

```
let g2 = {
  sep = 4;
  adj = [|[6; 7; 8]; [6]; [7]; [8]; [5; 9]; [4]; [0; 1]; [0; 2]; [0; 3]; [4]|]
};;
```

   (b) Implémenter une fonction `transversal_min : biparti -> int option array -> int list` implémentant cette construction.
   (c) Démontrer le théorème de König à l'aide de cette construction.
   Indications :
   - Montrer qu'aucun sommet non couplé de V n'appartient à Z.
   - Montrer que pour toute arête $\{u, v\} \in C$ avec $u \in U$ et $v \in V$, on a $u \in Z \Leftrightarrow v \in Z$.
   - Montrer que T est un transversal de G.
   - Montrer que T et C ont même cardinal.
   - Montrer que T est minimum.