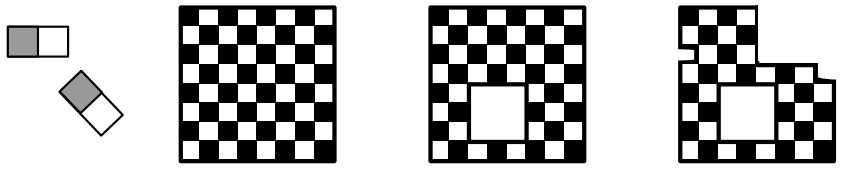
Master 1 – Modélisation, Graphes et Algorithmes
Projet : calcul d'un couplage dans un graphe biparti

1 Description du projet

Nous disposons d'un échiquier, dont certaines cases ont été supprimées. Nous aimerions le paver avec des dominos, si cela est possible.



L'échiquier mutilé peut-être représenté par un graphe : à chaque case restante nous lui associons un sommet ; et deux sommets sont adjacents si et seulement si les cases correspondante de l'échiquier sont voisines. Étant donné une case, les cases voisines sont celles se trouvant à sa gauche, à sa droite, au-dessus et en-dessous (si elles existent).

Un graphe $G = (V, E)$ est *biparti* si et seulement si ses sommets peuvent être partitionnés en deux ensembles N et B (noté $V = N \uplus B$) tels que les arêtes de E relient des sommets de N à des sommets de B. Autrement dit, il n'existe aucune arête entre deux sommets de N ou entre deux sommets de B.

Question 1 *Expliquez pourquoi le graphe représentant l'échiquier mutilé est un* graphe biparti.

Question 2 *Montrez que ce graphe contient au plus* $2n$ *arêtes, où* n *est le nombre de cases de l'échiquier mutilé.*

Étant donné un graphe $G = (V, E)$, un *couplage* $M \subseteq E$ est un sous-ensemble d'arêtes ne partageant aucun sommet commun. Ce couplage est *parfait* si chaque sommet de V appartient à une arête de M.

Question 3 *Expliquez en quoi un* couplage parfait*, dans le graphe représentant l'échiquier mutilé, est utile pour le paver des dominos.*

2 Couplage parfait dans un graphe biparti

L'algorithme de Hopcroft-Karp construit un *couplage parfait* dans un *graphe biparti* en temps $\mathcal{O}(m\sqrt{n})$. Puisque $m \leq 2n$ dans le graphe de l'échiquier, cela implique un temps d'exécution en $\mathcal{O}(n^{3/2})$.

Supposons que $M \subseteq E$ soit un couplage du graphe biparti $G = (V, E)$. Un sommet n'appartenant pas à une arête de M est dit *libre*. Un *chemin augmentant* est un chemin qui commence d'un sommet libre et se termine sur un autre sommet libre, en alternant des arêtes hors du couplage M avec des arêtes du couplage M. Ce chemin augmentant peut être réduit à une seule arête n'appartenant pas à M. Un chemin augmentant a donc une arête de plus hors du couplage M que dans le couplage, d'où le qualificatif « augmentant ».

Étant donné un couplage M et un chemin augmentant P, leur *différence symétrique* (notée $M \oplus P$) constitue un couplage de taille $|M| + 1$. La différence symétrique de deux ensembles X et Y est définie par :

$$X \oplus Y = (X \cup Y) - (X \cap Y).$$

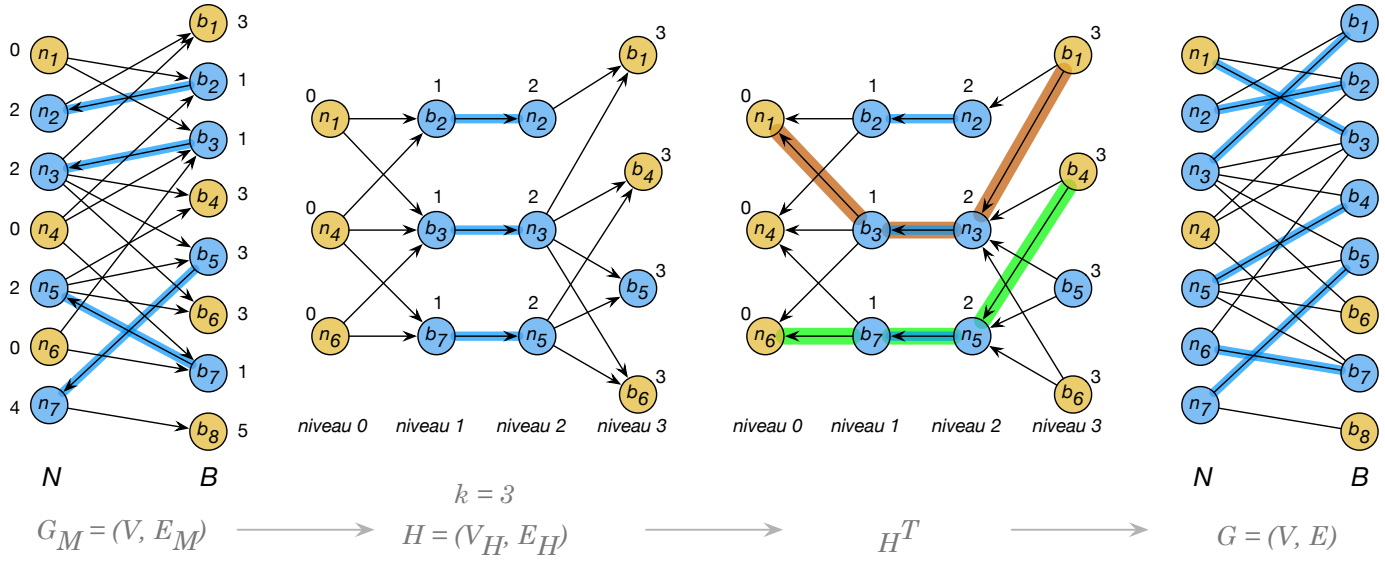
Ainsi, dans le cas d'un couplage M et d'un chemin augmentant P, $M \oplus P$ désignera les arêtes de P qui n'appartenaient pas à M.

L'algorithme de Hopcroft-Karp est incrémental : à chaque itération il recherche des chemins d'augmentations, puis les utilise pour augmenter la taille du couplage M.

Algorithme de Hopcroft-Karp($G = (N \uplus B, E)$)
Entrée : un graphe biparti $G = (V, E)$ où $V = N \uplus B$.
Sortie : un couplage $M \subseteq E$.

$M \leftarrow \emptyset$
répéter
  $\mathcal{P} \leftarrow \{P_1, P_2, \ldots, P_k\}$ un ensemble maximal de chemins augmentant sans sommets communs et de longueur minimale *(voir* $(\star)$ *ci-après)*
  $M \leftarrow M \oplus (P_1 \cup P_2 \cup \cdots \cup P_k)$
jusqu'à *ce que* $\mathcal{P}$ *soit vide*
retourner M

$(\star)$ Trouver les chemins augmentants disjoints de longueur minimale.



Nous commençons par créer un graphe orienté $G_M = (V, E_M)$ à partir de $G = (N \uplus B, E)$ et du couplage M. Les arcs de E_M sont obtenus :

— en orientant les arêtes $\{x, y\} \in E$ dans le sens (x, y) si $x \in N$, $y \in B$ et $\{x, y\} \notin M$;
— en orientant les arêtes $\{x, y\} \in E$ dans le sens (y, x) si $x \in N$, $y \in B$ et $\{x, y\} \in M$;

Question 4 *Écrire une fonction* `construire_GM` *qui attend en paramètre un graphe biparti G et un couplage M. Cette fonction construit le graphe G_M.*

À partir de G_M, nous construisons un graphe orienté sans cycle $H = (V_H, E_H)$. Ce graphe contient des *niveaux de sommets*, chaque niveau ne contenant (en alternance) que des sommets de N ou que des sommets de B. Le premier niveau (niveau 0) est celui contenant les sommets libres de N dans G_M. Les niveaux successifs sont obtenus par un *parcours en largeur* depuis chacun des sommets libres de N dans G_M.

On note k la plus petite distance dans G_M d'un sommet libre de B, lors du parcours depuis un sommet libre de N. Le dernier niveau dans H est le niveau k. Les sommets situés à une distance supérieure à k dans G_M n'appartiennent pas à H. On ne conserve dans E_H que les arcs de E_M allant d'un niveau au niveau immédiatement consécutif.

Question 5 *Écrire une fonction* `construire_niveaux` *qui attend en paramètre un graphe G_M et qui construit le graphe des niveaux ainsi que la valeur k.*

Les arêtes E_M sont obtenus des *parcours en largeur* initiés depuis chacun des sommets libres de N. Au premier niveau de ce parcours ne se trouvent que des arêtes n'appartenant pas à M, puisque les sommets de départ étaient libres. Puis les arêtes traversées alternent entre celles du couplage M et celles n'appartenant pas à M. On observe que lors du parcours en largeur, les recherches des successeurs depuis un sommet de N n'utilisent que des arêtes n'appartenant pas à M, alors que les recherches des successeurs à partir d'un sommet de B utilisent uniquement des arêtes de M. Le parcours se termine dès qu'on atteint un niveau k où un sommet libre de B est atteint.

Le graphe H est ensuite transposé pour obtenir H^T. Cette étape renverse simplement le sens de l'orientation des arcs de H. Les niveaux restent numérotés comme dans H.

Question 6 *Écrire une fonction* `renverser` *qui attend en paramètre un graphe orienté H et qui calcule H^T.*

À partir de chacun des sommets libres du niveau k, un *parcours en profondeur* est initié. L'objectif est, pour chacun de ces sommets, de construire un chemin allant jusqu'à l'un des sommets libres du niveau 0. Une fois obtenu un *chemin augmentant*, ses sommets ne peuvent plus servir à la construction d'un autre chemin. Il est donc possible que certains sommets libres du niveau k ne parviennent pas à construire leur chemin. L'ensemble $\mathcal{P}$ est la collection des chemins augmentants qui ont été construits.

Question 7 *Écrire une fonction* `chemins_augmentants` *qui se charge de calculer l'ensemble $\mathcal{P}$ de chemins utilisés par l'algorithme de* `HopcroftKarp`*.*

Question 8 *Implémentez l'algorithme de* `HopcroftKarp`*.*

3 Contraintes d'implémentation

Pour l'implémentation, vous utiliserez le langage de votre choix parmi Python, Caml, C++, Java. Les graphes utilisés dans votre implémentation devront être représentés par des listes d'adjacences.

3.1 Entrée

Afin de tester votre algorithme, votre programme devra lire un échiquier mutilé stocké dans un fichier. L'échiquier est donné sous le format suivant qu'il est **impératif de respecter**.

```
8
B N B N B N X X
N B N B N B X X
B N B N B N B X
X X N B N B N X
B N B N B N B N
N B X X N B N B
B N X X B N B N
N B N B N B N B
```

où :

- la première ligne est un entier n qui indique la taille $n \times n$ de l'échiquier ;
- `N` indique que la case est noire, `B` que la case est blanche et `X` que la case n'existe pas.

3.2 Sortie

Votre programme affichera sur la sortie standard un booléen (`True` / `False`) qui indiquera si l'échiquier mutilé peut être pavé par des dominos. S'il est possible de le paver avec t dominos, il produira un fichier avec les dominos numérotés de 1 à t. Voici un exemple de fichier :

```
 1  1  2  2  3  3  X  X
 4  5  6  6  7  7  X  X
 4  5  8  8  9  9 10  X
 X  X 11 11 12 12 10  X
13 14 15 15 16 16 17 17
13 14  X  X 18 18 19 19
20 20  X  X 21 21 22 22
23 23 24 24 25 25 26 26
```

4 Rapport de projet

Vous devez rendre un rapport d'au **maximum 4 pages**. Ce document devra :

- indiquer vos noms et prénoms ;
- répondre aux questions posées ;
- indiquer comment exécuter ou compiler votre code (en quelques lignes) ;
- expliquer vos choix d'implémentation, difficultés rencontrées, solutions proposées, ... ;
- si nécessaire, vous pourrez également y présenter le diagramme de classes et un pseudo-code simplifié des méthodes plus complexes.

5 Modalités

Vous réaliserez ce projet par groupe de 2 étudiants. Une soutenance de projet pourra éventuellement avoir lieu après le rendu du projet.

Vous devez rendre une archive (au format *.zip* ou similaire) contenant le code source de votre programme et votre rapport au format PDF. L'archive doit porter le nom des auteurs. Vous devrez déposer cette archive sur la page Celene du cours.

Le projet est à rendre avant le 13 décembre 2025, minuit.