Introduction à l'algorithmique **Hiver 2024**

IFT2125-6001 Évaluation (7.5%)

Devoir 4 - Le dernier effort

Consignes générales : À faire en équipe de 1 à 2 personnes. Remettre seulement les fichiers de code python. Indiquez vos noms et matricules sur tous les fichiers.

Votre code doit obligatoirement passer les tests fournis avec le devoir, sinon une note de 0 sera attribuée aux exercices dont les tests de base ne sont pas réussis. D'autres tests seront rajoutés lors de la correction. Seulement les librairies standards de Python sont permises, sauf si mentionné autrement.

Il va de soit que le code doit être clair, bien indenté et bien commenté.

La date de remise est le 23 avril à 22h30 La structure de la remise dans studium doit être la suivante :

```
StudiUM
├── well_placement.py
├── school_group_separation.py
└── prime_friend_groups.py
```

1 Progrès technologique ?= progrès social - Discussion (0 points)

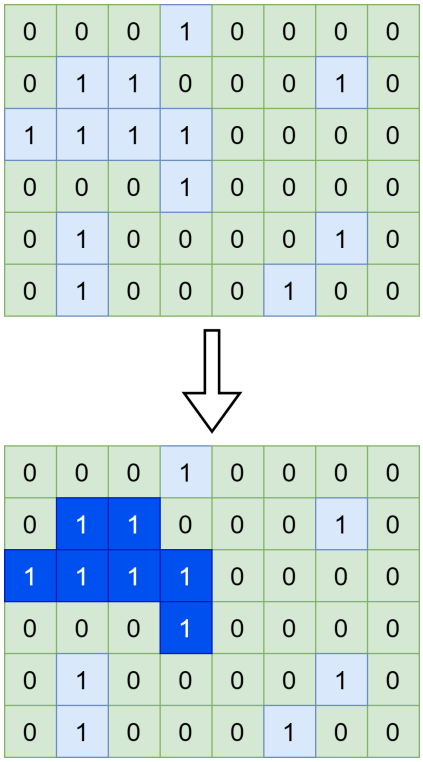
Nous sommes bien fiers et fières quand nous découvrons des algorithmes efficaces pour résoudres des problèmes, ou lorsque notre implémentation est grandement utilisée. Par contre, est-ce que le progrès technologique signifie tout le temps un progrès social?

Discutez de vive voix pendant quelques petites minutes avec un.e collègue de classe à propos de cas où des algorithmes (avec les logiciels qui les incluent) ont eu des effets négatifs sur certains aspects de notre société :

- Économie
- Environnement
- Égalité

- Justice
- Santé physique
- Santé mentale
- Etc.

2 Où puis-je placer mon puit? - Code Python (30 points)



Vous avez le mandat de trouver le meilleur emplacement pour un puit d'eau sur un terrain. Vous disposez de données géographiques qui indiquent la présence d'aquifer (réserve d'eau) sous le sol. Ces données séparent le terrain dans une grille où chaque case a une valeur de 1 si un aquifer se trouve sous le sol et 0 s'il n'y a pas de trace d'aquifer à cette case là.

On veut mettre le puit là où la réserve d'eau sera la plus grande donc on cherche à identifier le plus grand aquifer. Dans ce problème, on assume qu'un aquifer est un ensemble de case qui sont connecté par verticalement (haut et bas) et horizontalement

(gauche et droite) sur la grille. Les cases en diagonales ne se connectent pas. On vous demande de retourner seulement la taille du plus grand aquifer. S'il n'y a pas de trace d'eau, alors on retourne 0.

Données :

```
<hauteur> <largeur>
<1ère rangée 0/1>
<2e rangée 0/1>
...
<dernière rangée 0/1>
```

Exemples :

```
Exemple 1 :
3 4
0001
1100
1011
--> doit donner 3

Exemple 2 :
3 4
0000
0000
0000
--> doit donner 0

Exemple 3 :
10 20
00000000000000000000
00110000001000000000
00110000000000000000
00000000000001000000
00000000000000000000
00000001000000000010
00000000000000000001
00000000000000000111
00000000000000010011
00000000000000001111
--> doit donner 10
```

Code

Votre code doit lire un fichier spécifié en premier argument qui contient un problème et écrire la réponse dans un fichier spécifié en second argument.

Exemple d'appel :

```
python3 well_placement.py input.txt output.txt
```

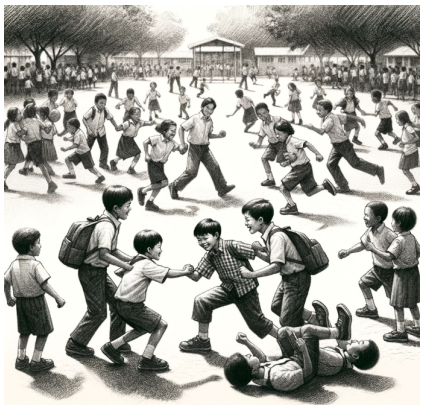
Un fichier *test_well_placement.py* et des fichiers d'entrées vous sont fournis pour vous aider à tester votre code.

Exemple d'appel :

```
python3 test_well_placement.py
```

Remise
Compléter le fichier *well_placement.py* fournis, et ne remettre **uniquement** que ce fichier. Ne **pas** remettre le fichier *test_well_placement.py*, ou n'importe quel autre fichier de test.

3 Les écoliers turbulents - Code Python (30 points)



Après vos succès auprès des jeunes avec la création de jeux de cartes et des labyrinthes, une école primaire à Montréal vous consulte à propos d'un problème avec des écoliers turbulents.

Les enseignants et enseignantes ont noté que certaines paires d'écolières et d'écoliers sont dangeureuses : lorsqu'on met les deux enfants d'une paire dangeureuse dans un même groupe pour une activité, on peut s'attendre presqu'avec certitude que des problèmes comportementaux vont surgir à un moment donné.

Avec une liste d'écolières et d'écoliers et une liste des paires dangeureuses, déterminez si les enfants peuvent être séparé en deux groupes tels que chaque paire dangeureuse est séparée entre les deux groupes. Si c'est possible, retournez une combinaisons d'enfants en deux groupes qui fonctionne, sinon, retourner "impossible".

Données :

```
<n le nombre d'enfants>
<nom 1>
<nom 2>
...
<nom n>
<m le nombre de paires dangereuses>
<nom A> <nom B>
<nom C> <nom D>
...
```

Exemples :

```
Exemple 1 :
5
Isabella
Jacob
Kaitlyn
Liam
Mohammed
2
Kaitlyn Mohammed
Liam Isabella

--> doit donner (parmi d'autres réponses valides)
Isabella Jacob Kaitlyn
Liam Mohammed


Exemple 2 :
3
Noah
Olivia
Parker
3
Noah Olivia
Olivia Parker
Parker Noah

--> doit donner
```

```
impossible
```

Code
Votre code doit lire un fichier spécifié en premier argument qui contient un problème et écrire la réponse dans un fichier spécifié en second argument.

Exemple d'appel :

```
python3 school_group_separation.py input.txt output.txt
```

Un fichier *test_school_group_separation.py* et des fichiers d'entrées vous sont fournis pour vous aider à tester votre code.

Exemple d'appel :

```
python3 test_school_group_separation.py
```

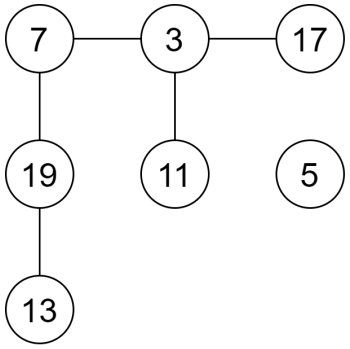
Remise
Compléter le fichier *school_group_separation.py* fournis, et ne remettre **uniquement** que ce fichier. Ne **pas** remettre le fichier *test_school_group_separation.py*, ou n'importe quel autre fichier de test.

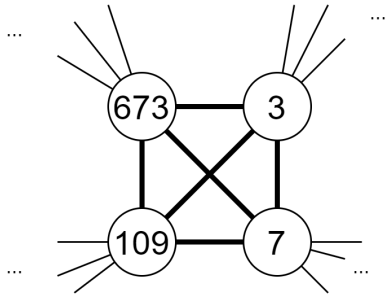
4 Les ensembles de nombres premiers spéciaux - Code Python (40 points)

Soit p et q deux nombres premiers. On va dire que cette paire de nombres est spéciale lorsque la concaténation des deux nombres dans n'importe quel ordre donne aussi un nombre premier. Par exemple, 3 et 7 forment une paire spéciale car 37 et 73 sont premiers. Par contre, 2 et 3 ne forment pas une paire spéciale car 23 est premier mais 32 ne l'est pas. Notez qu'on ne peut pas considérer 307 comme une concaténation de 3 et 7 à cause du 0 inséré en surplus.

Un ensemble de nombres premiers est spécial lorsque toutes les paires de nombres dans cet ensemble forment des paires spéciales. En théorie des graphes, on appelle cela une clique.

Pour caractériser un tel ensemble, on peut regarder la somme de tous ces éléments. Par exemple, l'ensemble spécial de $(67, 3, 37)$ a une somme de 107. Cette caractérisation est n'est pas nécessairement unique.





Trouvez l'ensemble de 4 nombres premiers qui est spécial et dont la somme est la n-ième parmi tous les ensembles spéciaux de de 4 nombres premiers. $1 \leq n \leq 7$. Vous devez retourner cette somme du n-ième plus petit ensemble spécial dans le fichier donné en argument. Votre code doit trouver et calculer ces valeurs et pas garder dans le code les réponses pré-calculées.

Pour cet exercice, on vous demande d'implémenter la fonction de test primalité de Miller-Rabin qui pourra grandement accélérer certains calculs (les tests de primalité vont sûrement prendre la majorité du temps de calcul). Cet algorithme qui permet de vérifier la primalité d'un nombre est un algorithme probabilite Monte Carlo. Par contre, pour des valeurs limitées ($< 10^{24}$), il a déjà été déterminé qu'il suffit de tester avec un ensemble spécifique de nombres pour en obtenir un algorithme déterministe!

Ne soyez pas surpris si le code prends quelques dizaines de secondes à s'exécuter.

Par exemple, voici les 3 premiers ensembles spéciaux de 4 nombres premiers en croissant de somme totale :

```
(3, 7, 109, 673)
(3, 37, 67, 2377)
(7, 19, 97, 3727)
...
```

Voici leurs sommes :

```
792
2484
3850
...
```

Données : aucune (seulement 2 paramètres passés en arguments)

Code
Votre code doit lire un entier spécifié en premier argument et écrire la réponse dans un fichier spécifié en second argument.

Exemple d'appel :

```
python3 prime_special_groups.py 2 output.txt
```

Réponse attendue dans le fichier :

```
2484
```

Un fichier *test_prime_special_groups.py* et des fichiers d'entrées vous sont fournis pour vous aider à tester votre code.

Exemple d'appel :

```
python3 test_prime_special_groups.py
```

Important : Ces tests sont fait avec $n = 1, 2, 3, 6$. On vous demande ne pas partager publiquement vos solutions pour $n = 4, 5, 7$. Merci!

Remise
Compléter le fichier *test_prime_special_groups.py* fournis, et ne remettre **uniquement** que ce fichier. Ne **pas** remettre le fichier *test_prime_special_groups.py*, ou n'importe quel autre fichier de test.

Bonus 10%
Nous allons accorder un bonus maximal de 10% (pour le devoir). Ce bonus sera annoncé

seulement après la remise officielle du devoir. Le bonus sera conditionnel à la réussite de(s) l'exercice(s) ciblé(s) du présent devoir et dévoilé(s) seulement après la remise. Le bonus sera aussi condionnel à la remise sans retard du devoir.