Loi norma

TELECOM Nancy

E2 : Examen Machine 2

CS54 : Computer Science 54
Première année

UNIVERSITÉ DE LORRAINE

Lundi 8 novembre 2021
durée: 4 heures.

Propos liminaires

Vous disposez de 4 heures pour réaliser tout ou partie de l'examen. Prenez le temps de lire les problèmes, de les comprendre, de réfléchir à des solutions. Il existe de multiples manières de résoudre les problèmes posés. Tentez, explorez. Aucune solution, si elle donne le bon résultat, n'est mauvaise !

Veillez à respecter scrupuleusement les noms demandés pour chaque fichier python et chaque fonction. Assurez vous de bien déposer vos productions sur le dépôt git qui vous a été attribué pour l'examen, les évaluations étant automatiques.

Les seuls documents autorisés (en ligne ou sur papier) sont : les supports de cours, une page web sur le projet git de l'examen, une page web sur la documentation officielle de python. Toute autre consultation sera considérée comme une fraude.

Ne copiez pas, ne trichez pas, ne récupérez pas de code d'internet. Les conséquences (conseil de discipline, exclusion, ...) sont trop lourdes pour s'y risquer.

Préparation de l'environnement de travail

Récupérez (par clonage) une version locale du projet git qui vous a été affecté pour cet examen.

À chaque étape de l'examen, nous vous demandons de déposer vos réalisations python sur le git du projet que vous venez de récupérer. Sur chaque exercice, nous vous imposons le nom du fichier python associé et parfois le nom de quelques fonctions clés.

Au besoin, référez-vous à la page Arche de CS54 pour les bases de Git et GitLab :
`https://arche.univ-lorraine.fr/mod/page/view.php?id=1252876`.

★ **Exercice 1.** À cheval ! (8 points)

Dans cet exercice, vous devrez écrire un certain nombre de fonctions python qui suivent chacune une signature imposée (nom de fonction et types des paramètres). **Toutes vos fonctions devront être implémentées dans un même fichier python dénommé : `P1.py`.** Vous êtes autorisés (et il est même conseillé) d'écrire d'autres fonctions annexes.

Sur un échiquier, le cavalier se déplace de façon singulière en L. Pour cela il se déplace de façon horizontale ou verticale de deux cases puis d'une case (ou d'une case puis de deux) de façon perpendiculaire.
On suppose que l'échiquier est modélisé par une liste dénommée `echiquier` de taille $n \times n$ d'entiers. `echiquier[i]==0` si le cavalier n'est pas passé (au sens ne n'est jamais posé après un déplacement) dans la ligne l et la colonne c qui correspondent à cet indice, sinon la cellule vaut 1.

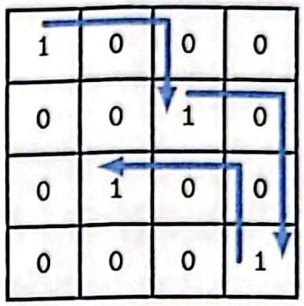

Figure 1: Exemple de progression d'un cavalier sur l'échiquier et de son incidence sur les valeurs des cases

▷ **Question 1.** (0,5 pt) Écrivez une fonction qui renvoie l'indice associé à la case dans la liste `echiquier` :

```
indice_de_position(position: Tuple[int, int], taille_cote: int) -> int
```

Le numéro de la première ligne et de la première colonne est 1 et le contenu associé est à l'indice 0 de la liste `echiquier`. Le paramètre `taille_cote` représente la longueur d'un côté de l'échiquier.

▷ **Question 2.** (1 pt) Écrivez une fonction :

```
suivants(position: Tuple[int, int], taille_cote: int) -> List[Tuple[int, int]]
```

qui, étant donnée une position courante du cavalier (en paramètre de la fonction) et la longueur du côté de l'échiquier (`taille_cote`), calcule et retourne toutes les positions atteignables sur l'échiquier par celui-ci en un déplacement.

▷ **Question 3.** (4 pts) Étant donné un échiquier de longueur de côté `taille_cote` donné et une position initiale du cavalier sous forme d'indice de ligne et d'indice de colonne, concevez une fonction récursive qui recherche un parcours pour le cavalier à l'issue duquel il est passé dans toutes les cases de l'échiquier sans passer deux fois dans la même case.

Le profil attendu de la fonction qui permet de réaliser cela est le suivant :

```
cavalier(echiquier: List[int], position: Tuple[int, int], taille_cote: int) -> bool
```

Ses paramètres sont :

- l'échiquier (liste de taille n^2 d'entiers dont toutes les valeurs sont initialement à 0),
- la position du cavalier (tuple indiquant la ligne et la colonne) où il s'est posé (l'échiquier passé en paramètre a donc une valeur non nulle à la position correspondante),
- la longueur d'un côté de l'échiquier.

Le résultat renvoyé par la fonction est :

- une valeur booléenne qui est égale à `True` si une solution au problème a été trouvée, à `False` sinon.

▷ **Question 4.** (3 pts) Étendez votre fonction précédente pour qu'elle renvoie, lorsqu'une solution a été trouvée, le chemin parcouru par le cavalier depuis sa position de départ. Votre nouvelle fonction aura le profil suivant :

```
chemin_cavalier(echiquier: List[int], position: Tuple[int, int], taille_cote: int)
    -> Tuple[bool, List[Tuple[int, int]]]
```

Ses paramètres sont:

- l'échiquier (liste de taille n^2 d'entiers),
- la position du cavalier (tuple indiquant la ligne et la colonne) où il s'est posé (l'échiquier passé en paramètre a donc une valeur non nulle à la position correspondante),
- la longueur d'un côté de l'échiquier.

Le résultat renvoyé par la fonction est un tuple composé de :

- une valeur booléenne qui est égale à `True` si une solution au problème a été trouvée, à `False` sinon ;
- la liste des positions successives prises par le cavalier pour atteindre la position finale. La liste vide si aucune solution n'a été trouvée.

★ **Exercice 2.** *Les voyages forment la jeunesse (6 points)*

Les systèmes de divertissement à bord des avions offrent aujourd'hui des catalogues de contenus absolument gigantesques dont plusieurs centaines voire milliers de clips vidéo. Afin d'améliorer l'expérience des usagers, la compagnie Air Kifance, vous demande d'ajouter un service qui, étant donnée une durée de vol d connue, compose, pour chaque passager, une liste de lecture optimale. La liste de lecture optimale démarre exactement 8 minutes après le début du vol puis enchaîne un ensemble de clips sans interruption et s'arrête au plus tard, 15 minutes avant la fin du vol. L'arrêt se fait impérativement en fin d'un clip (pas d'interruption possible en milieu de clip, trop frustrant !). Une liste de lecture optimale va mobiliser l'attention de l'usager sur la plus grande période possible, idéalement sur l'intégralité de la période d'activité du système de divertissement durant le vol.

Votre code source doit se trouver dans le fichier `P2.py`.

▷ **Question 5.** (1 pt) Écrire une fonction qui lit un fichier texte (dont le nom est donné en paramètre) contenant les informations sur l'ensemble des clips vidéos disponibles :

```
load_clips_data(clips_database_filename: str) -> Tuple[int, List[Tuple[int, str]]]
```

Son paramètre est le nom du fichier de données à lire dont le contenu est illustré à la figure 2.

```
40
190, AC/DC - You shook me all night long
220, MUSE - Plug in Baby
416, Iron Maiden - Afraid to shoot strangers
235, Shaka Ponk - I'm Picky
221, Foo Fighters - Saint Cecila
232, Imagine Dragons - Walking the wire
161, Greenday - Boulevard of broken dreams
253, Rise Against - Hero of war
```

Figure 2: Dans l'exemple ci-dessus, le vol dure 40 minutes; le titre d'AC/DC dure 190 secondes, celui de MUSE, 220 secondes et ainsi de suite.

Le résultat renvoyé par la fonction est un tuple composé de :

- une valeur entière correspondant à la durée total du vol en minutes ;
- la liste des données des clips. Chaque clip est un tuple composé d'une valeur entière correspondant à la durée en secondes du clip et d'une chaîne de caractères précisant le titre du clip.

▷ **Question 6.** (5 pt) Écrire une fonction qui calcule une séquence optimale de clips pour un vol d'une durée d minutes donnée :

```
compute_optimal_playlist(flight_duration_in_minutes: int, clips: List[Tuple[int, str
    -> Tuple[int, List[Tuple[int, str]]]
```

Ses paramètres sont:

- la durée total du vol en minutes
- la liste des données des clips. Chaque clip est un tuple composé d'une valeur entière correspondant à la durée en secondes du clip et d'une chaîne de caractères précisant le titre du clip.

Le résultat renvoyé par la fonction est un tuple composé de :

- une valeur entière indiquant à la durée totale en seconde de la liste de lecture proposée ;
- la liste des clips sélectionnés. Chaque clip est un tuple composé d'une valeur entière correspondant à la durée en secondes du clip et d'une chaîne de caractères précisant le titre du clip.

```
190, AC/DC - You shook me all night long
416, Iron Maiden - Afraid to shoot strangers
161, Greenday - Boulevard of broken dreams
253, Rise Against - Hero of war
```

Figure 3: Le total musical est de 1020 secondes (17 minutes) et correspond à la liste de lecture la plus longue possible compte-tenu des contraintes de temps de vol, des temps d'activation et de désactivation du système de divertissement à bord.

★ **Exercice 3.** Déverrouillez votre mobile Android (6 points)

La méthode la plus utilisée pour déverrouiller un téléphone Android consiste à "dessiner" un schéma sur le pavé numérique en glissant de façon ininterrompue son doigt sur l'écran. Les touches de l'écran de verrouillage sont numérotées de 1 à 9 et organisées comme illustrées dans la figure 4.

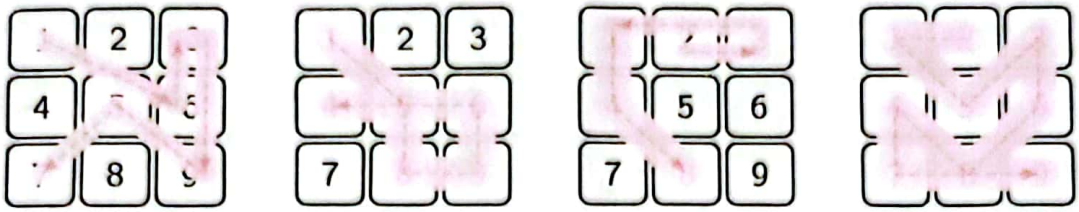

Figure 4: Clavier Android et exemples de schémas (codes) valides

Un schéma est considéré valide si et seulement si :

- Un digit ne peut faire partie qu'une seule fois du schéma ; ainsi, un schéma connecte au plus les 9 digits ;
- un schéma connectera toujours le premier digit non connecté sur son trajet. Puis, il pourra continuer pour connecter d'autres digits non connectés ;
- un schéma peut traverser sur son trajet un digit précédemment connecté pour aller connecter un digit non connecté.

Par exemple, `1-6-3-9-5-7`, `1-5-8-9-6-4`, `2-3-1-4-8` et `2-1-5-3-6-8-4-7-9` sont des schémas valides; `2-1-7` n'en n'est pas un ! Vous noterez que dans l'exemple `1-5-8-9-6-4`, le digit 5 n'est retenu qu'une fois (lors de sa première occurence) dans le code associé au chemin. Il en est de même pour le digit 2 dans l'exemple `2-3-1-4-8`. Il est aussi important de noter que le schéma `9-8-7-4-6-3-5-1-2` qui correspondrait au parcours dans le sens inverse du dernier schéma illustré (`2-1-5-3-6-8-4-7-9`) n'est pas valide. En effet, il n'est pas autorisé de repasser par le digit 8 déjà sélectionné pour aller du digit 4 au digit 6 car ces trois digits ne sont pas "alignés".

Votre code source devra se trouver dans un fichier dénommé `P3.py`.

▷ **Question 7.** (4 pt) Écrivez une fonction qui calcule le nombre de schémas valides de longueur inférieure ou égale à n (avec $1 \leq n \leq 9$) :

```
compute_unlock_patterns_count(max_length: int) -> int
```

Pour vos besoins de vérification, il y a 7152 schémas valides de longueur 5 (constitués de 5 digits) soit 9161 schémas de longueur inférieure ou égale à 5. C'est cette dernière valeur (9161) qui est le résultat attendu pour l'appel de votre fonction avec 5 comme valeur pour le paramètre `max_length`.

▷ **Question 8.** (2 pt) Écrivez une nouvelle fonction qui calcule tous les schémas valides de longueur inférieure ou égale à n (avec $1 \leq n \leq 9$) :

```
compute_unlock_patterns(max_length: int) -> List[str]
```

Le résultat renvoyé par la fonction est une liste contenant chaque schéma valide sous forme d'une chaîne de caractères (de format `2-3-1-4`, chaque digit est donc séparé du suivant par un tiret).