Algorithmique:
Programmation Dynamique 1

Bonus :

Exercice : Seul l'exercice 1 de cette série (Voyage en train) fait l'objet du bonus.

Délai : mardi 12 novembre à 23h59 au plus tard

Fichiers : (avec votre nom et votre prénom à la place de 'NomPrenom')

- Un fichier NomPrenom.pdf répondant à toutes les questions de l'exercice.
- Un fichier python (NomPrenom.py) de votre implémentation de l'algorithme. N'envoyez pas le bytecode (fichier .pyc).

Rendu : Les fichiers doivent être soumis sur Moodle dans le devoir 'Bonus 4'.

Consignes : Le nom respect des consignes entrainera la non-evaluation de votre travail.

- L'implémentation et le rapport sont des travaux **individuels**.
- Respectez les spécifications de l'input et de l'output pour les fonctions demandées.
- Utiliser une version de Python > 3.0
- Rendez un rapport dactylographié au format pdf
- Mettez votre nom sur le rapport ainsi qu'un titre.

Conseil : Tester votre code avant de le rendre dans un nouvel environnement.

1 Voyage en train

Dans ce problème on veut optimiser le coût d'un voyage en train d'une ville à une autre, au sein d'une compagnie ferroviaire ayant une étrange politique de prix des billets. On peut résumer les coûts sous la forme d'une matrice M. Voici un exemple :

	Genève	Nyon	Morges	Renens	Lausanne
Genève	0				
Nyon	5	0			
Morges	10	4	0		
Renens	14	8	4	0	
Lausanne	17	12	7	4	0

On remarque que, pour certains trajets, la somme du prix des étapes 'atomiques' entre deux villes peut être plus grande, égale ou plus petite que le prix pour le train direct.

Dans le cas où l'on veut faire le trajet Genève-Lausanne, la somme du prix des étapes vaut 17, le trajet direct vaut 17 également, mais le trajet optimal, qui consiste à faire Genève-Nyon-Morges-Lausanne, coûte seulement 16.

Nous allons pour cet exercice considérer le cas général où il y a n villes, soit un nombre maximal de $n-2$ étapes intermédiaires. Lors d'un trajet, on ne revient jamais en arrière (d'où le fait que M soit à moitié vide). On est libre de prendre n'importe quel nombre de sous-trajets. Le but est d'analyser et d'implémenter une stratégie de programmation dynamique pour trouver le coût optimal pour aller de la ville numéro 0 à la ville numéro $n-1$. Pour ce faire, nous proposons de remplir un tableau T des coûts optimaux, c'est-à-dire un tableau similaire à la matrice Mci-dessus, ne comprenant non-pas les coûts 'officiels' de la compagnie ferroviaire, mais les coûts optimaux pour chaque sous-trajet. Par exemple, la case correspondant au trajet Genève-Lausanne aurait une valeur de 16.

1.1 Fonction de récurrence (3 points)

1. Nous avons défini ci-dessus en quoi consiste la matrice T des coûts optimaux. Donnez une relation qui permette de trouver la valeur de T_{ij} (=coût optimal pour aller de la ville numéro i à la ville numéro j). N'oubliez pas que le coût optimal (minimal) d'un trajet est le minimum des coûts de l'ensemble des trajets possibles. Justifiez !
2. Quelle est la conséquence de ce modèle sur la manière dont on doit 'remplir' T ? (1 point)

1.2 Complexité en temps (3 points)

1. Quelle est selon vous la complexité en temps (en fonction de n) de l'algorithme de programmation dynamique pour résoudre le problème ? Argumentez. (2 points)

2. Si l'on tentait de résoudre le problème avec une stratégie de Backtracking, quelle serait cette complexité ? (1 point)

1.3 Implémentation (5 points)

Implémentez un solver pour ce problème avec une stratégie de programmation dynamique utilisant la fonction de récursion que vous donnez à la question 1.1.

Dans le code python que vous devez rendre, il doit y avoir une fonction `get_solution` qui a comme :

input la matrice des coûts de la compagnie ferroviaire M, sous forme de liste de liste. Par exemple :

```
M = [
 [0, None, None, None, None],
 [5, 0, None, None, None],
 [10, 4, 0, None, None],
 [14, 8, 4, 0, None],
 [17, 12, 7, 4, 0]]
```

output la matrice T des coûts correspondants (sous forme de liste de liste), le chemin optimal (sous forme de liste) et le coût du chemin optimal entre la ville 0 et la ville $n-1$. Dans l'exemple :

```
T = [
 [0, None, None, None, None],
 [5, 0, None, None, None],
 [9, 4, 0, None, None],
 [13, 8, 4, 0, None],
 [16, 11, 7, 4, 0]]

path = (0, 1, 2, 4)

cout = 16
```

Pour vous aider à la production et à l'affichage des matrices de coûts, vous trouverez sur Moodle un module (fichier) python nommé *aidevoyage.py*.

Voici un exemple d'utilisation :

```
import aidevoyage
n = 5
M = aidevoyage.gen_matrix(n) #génération aléatoire de M, matrice des coûts de taille n
```

```
aidevoyage.print_matrix(M) # imprime M à l'écran
```

NB : Les matrices ainsi générées sont accédées de la manière suivante : $M[i][j]$ correspond à la valeur de la $i$ème *ligne* et de la $j$ème *colonne*, et non pas l'inverse. De plus, les éléments au-dessus de la diagonale ont la valeur *None* si vous essayez d'y accéder, et sont visualisés par un 'x' lorsque vous imprimez la matrice à l'écran.

Notez que suivant l'implémentation, reconstruire le chemin n'est pas trivial et doit faire l'objet d'une fonction spécifique.

2 Rendu de la monnaie

Implémentez en python la stratégie de programmation dynamique vue en cours pour le problème du rendu de la monnaie.