A lire

L'algorithme que vous avez utilisé pour générer des sujets d'examen utilise une approche basée sur le problème du sac à dos (ou knapsack problem) pour tenter de respecter au mieux les contraintes définies. Ces contraintes incluent le nombre maximal de points, la durée maximale, et la distribution des niveaux de difficulté des questions.

### Objectif de l'algorithme

L'objectif principal de cet algorithme est de générer des sujets d'examen qui respectent au mieux les contraintes spécifiées. Cela signifie qu'il essaie de trouver un ensemble de questions pour chaque sujet d'examen qui :

1. \*\*Totalise un nombre de points inférieur ou égal au nombre de points maximal\*\*.

2. \*\*Totalise une durée de réponse inférieure ou égale à la durée maximale\*\*.

3. \*\*Respecte la distribution des niveaux de difficulté définie\*\* (par exemple, un certain pourcentage de questions faciles, moyennes et difficiles).

### Respect des contraintes

Il est important de noter que l'algorithme ne garantit pas que toutes les contraintes seront toujours respectées à 100%. Voici pourquoi :

- \*\*Points et durée\*\* : L'algorithme essaie de ne pas dépasser le nombre de points maximal et la durée maximale, mais il pourrait ne pas atteindre exactement ces valeurs en raison de la combinaison spécifique des questions disponibles.

- \*\*Distribution des difficultés\*\* : L'algorithme tente de respecter la distribution des niveaux de difficulté, mais encore une fois, cela dépend des questions disponibles. S'il n'y a pas suffisamment de questions pour respecter parfaitement cette distribution, l'algorithme fera de son mieux pour s'en approcher.

### Importance de la deuxième phase (Validation par arbre de décision)

Puisque l'algorithme essaie de respecter les contraintes au mieux mais ne peut pas garantir qu'elles seront toujours respectées, la deuxième phase de validation par un arbre de décision est cruciale. Cette phase permettra de vérifier si les sujets générés respectent bien les contraintes définies. Voici comment cela fonctionne :

1. \*\*Génération des sujets\*\* : L'algorithme génère plusieurs sujets d'examen en respectant au mieux les contraintes.

2. \*\*Validation des sujets\*\* : Chaque sujet généré est passé dans un arbre de décision (ou un autre modèle de machine learning) qui évalue s'il respecte suffisamment les contraintes pour être considéré comme valide.

3. \*\*Evaluation de la validité\*\* : L'arbre de décision renvoie une décision (valide ou non valide) ou un score de validité, indiquant à quel point le sujet respecte les contraintes.

### Exemple de sortie avec la mise à jour demandée

Voici une version mise à jour de votre code qui inclut les points après le nombre de points et la difficulté après la durée dans les fichiers texte :

```python

import json

import random

# Charger le jeu de données

with open('questions.json', 'r') as file:

data = json.load(file)

questions = data['questions']

# Fonction pour vérifier la validité d'un sujet

def est\_valide(sujet, max\_points, max\_duration, difficulty\_distribution):

points = sum([q['marks'] for q in sujet])

duration = sum([int(q['duree\_reponse'].split()[0]) for q in sujet])

difficulty\_counts = [0, 0, 0]

for q in sujet:

difficulty\_counts[q['difficulte'] - 1] += 1

return (points <= max\_points and

duration <= max\_duration and

all(difficulty\_counts[i] <= difficulty\_distribution[i] for i in range(3)))

# Fonction pour générer des sujets en utilisant une approche de sac à dos

def generer\_sujets(questions, n\_sujets, max\_points, max\_duration, difficulty\_distribution):

sujets = []

for \_ in range(n\_sujets):

sujet = []

tentative = 0

while (not est\_valide(sujet, max\_points, max\_duration, difficulty\_distribution) or len(sujet) == 0) and tentative < 1000:

random.shuffle(questions)

sujet = []

total\_points = 0

total\_duration = 0

difficulty\_counts = [0, 0, 0]

for q in questions:

points = total\_points + q['marks']

duration = total\_duration + int(q['duree\_reponse'].split()[0])

difficulty\_counts[q['difficulte'] - 1] += 1

if points <= max\_points and duration <= max\_duration and all(difficulty\_counts[i] <= difficulty\_distribution[i] for i in range(3)):

sujet.append(q)

total\_points = points

total\_duration = duration

if total\_points == max\_points and total\_duration == max\_duration:

break

tentative += 1

sujets.append(sujet)

return sujets

# Spécifier les paramètres

n\_sujets = 4

max\_points = 20

max\_duration = 45

difficulty\_distribution = [30, 50, 20]

# Générer les sujets

sujets\_generes = generer\_sujets(questions, n\_sujets, max\_points, max\_duration, difficulty\_distribution)

# Sauvegarder les sujets générés dans des fichiers texte

for i, sujet in enumerate(sujets\_generes):

with open(f'sujet\_{i+1}.txt', 'w') as file:

for question in sujet:

options = ' '.join(question['options'])

file.write(f"{question['question']}, {options}, {question['marks']} points, {question['duree\_reponse']} minutes, difficulté {question['difficulte']}\n")

print("Sujets générés et sauvegardés avec succès.")

```

### Conclusion pour le jury

Pour expliquer au jury :

1. \*\*Description de l'algorithme\*\* : Présentez l'algorithme comme une solution qui tente de respecter au mieux les contraintes de points, de durée et de distribution de la difficulté des questions.

2. \*\*Limites de l'algorithme\*\* : Expliquez que bien que l'algorithme fasse de son mieux, il ne garantit pas que toutes les contraintes seront respectées à 100% en raison de la variabilité des questions disponibles.

3. \*\*Importance de la validation\*\* : Soulignez l'importance de la phase de validation par arbre de décision pour vérifier si les sujets générés respectent suffisamment les contraintes, justifiant ainsi l'utilisation de cette deuxième phase dans votre travail.

Cela montre que votre approche est robuste et flexible, capable de générer des sujets d'examen qui respectent les contraintes autant que possible, tout en fournissant un mécanisme pour valider et évaluer la qualité des sujets générés.

Resolution algorithmique

Bien sûr, je vais vous fournir une explication détaillée de l'algorithme en utilisant un langage clair et structuré que vous pouvez inclure dans votre mémoire et dans vos diapositives pour le jury.

### Algorithme pour Générer des Sujets d'Examen

\*\*Objectif :\*\* Générer des sujets d'examen qui respectent les contraintes de points, de durée, et de distribution des niveaux de difficulté.

\*\*Paramètres :\*\*

- `questions` : Liste des questions disponibles.

- `n\_sujets` : Nombre de sujets d'examen à générer.

- `max\_points` : Nombre maximal de points pour un sujet.

- `max\_duration` : Durée maximale pour un sujet (en minutes).

- `difficulty\_distribution` : Distribution souhaitée des niveaux de difficulté (par exemple, [30, 50, 20] pour 30% de questions faciles, 50% de questions moyennes, et 20% de questions difficiles).

\*\*Étapes de l'algorithme :\*\*

1. \*\*Initialisation :\*\*

- Charger le jeu de données de questions.

- Initialiser une liste vide pour stocker les sujets générés.

2. \*\*Vérification de la Validité d'un Sujet :\*\*

- Calculer le total des points pour le sujet.

- Calculer la durée totale pour répondre aux questions du sujet.

- Compter le nombre de questions pour chaque niveau de difficulté.

- Vérifier que les totaux des points et de la durée ne dépassent pas les valeurs maximales spécifiées.

- Vérifier que la distribution des niveaux de difficulté respecte la distribution souhaitée.

3. \*\*Génération des Sujets :\*\*

- Pour chaque sujet à générer :

- Réinitialiser le sujet et les compteurs de points, durée, et niveaux de difficulté.

- Mélanger les questions pour introduire de la variabilité.

- Ajouter des questions au sujet tant que les contraintes de points, de durée, et de distribution des niveaux de difficulté sont respectées.

- Répéter le processus jusqu'à ce qu'un sujet valide soit trouvé ou qu'un nombre maximal de tentatives soit atteint.

4. \*\*Sauvegarde des Sujets :\*\*

- Pour chaque sujet généré, sauvegarder les questions dans un fichier texte, en incluant les points et la durée pour chaque question.

### Pseudo-code de l'algorithme :

```pseudo

charger\_questions()

initialiser\_liste\_sujets()

fonction est\_valide(sujet, max\_points, max\_duration, difficulty\_distribution):

calculer\_total\_points(sujet)

calculer\_duree\_totale(sujet)

compter\_difficultes(sujet)

retourner (total\_points <= max\_points et

total\_duree <= max\_duration et

distribution\_difficultes\_respectee)

fonction generer\_sujets(questions, n\_sujets, max\_points, max\_duration, difficulty\_distribution):

pour i de 1 à n\_sujets:

initialiser\_sujet()

tenter 1000 fois:

mélanger\_questions(questions)

pour chaque question dans questions:

si ajouter\_question\_ne\_viol\_pas\_contraintes(question, sujet, max\_points, max\_duration, difficulty\_distribution):

ajouter\_question(sujet, question)

si sujet\_est\_valide(sujet, max\_points, max\_duration, difficulty\_distribution):

sortir\_boucle

ajouter\_sujet(liste\_sujets, sujet)

retourner liste\_sujets

fonction ajouter\_question(sujet, question):

ajouter question à sujet

mettre à jour total\_points, total\_duree, distribution\_difficultes

pour chaque sujet dans generer\_sujets(questions, n\_sujets, max\_points, max\_duration, difficulty\_distribution):

sauvegarder\_sujet(sujet)

```

\*\*Titre :\*\* Algorithme de Génération de Sujets d'Examen

\*\*Slide Content :\*\*

1. \*\*Objectif :\*\*

- Générer des sujets d'examen respectant les contraintes de points, de durée et de distribution de difficulté.

2. \*\*Paramètres :\*\*

- Nombre de sujets à générer.

- Nombre maximal de points et durée maximale.

- Distribution des niveaux de difficulté.

3. \*\*Étapes de l'algorithme :\*\*

- Charger les questions.

- Vérifier la validité des sujets.

- Générer les sujets en mélangeant les questions et en respectant les contraintes.

- Sauvegarder les sujets générés.

4. \*\*Pseudo-code :\*\*

- Présenter un pseudo-code simplifié pour illustrer le processus.

5. \*\*Conclusion :\*\*

- L'algorithme vise à respecter au mieux les contraintes.

- Utilisation d'une phase de validation pour s'assurer du respect des contraintes.

- Importance de la validation pour garantir des sujets de qualité.

\*\*Illustration :\*\*

- Ajouter un diagramme ou un flowchart pour visualiser le processus de génération des sujets.

Cette explication vous permet de montrer au jury que vous avez une approche structurée et bien pensée pour générer des sujets d'examen tout en respectant les contraintes spécifiques.