Partie Exercice

Comment ça marche ?

La page HTML

La page Html doit contenir au minimum :

- L'appel à trois scripts :

    - le script permettant les requêtes ajax :

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

    - le script permettant l'affichage du graphe par vis :

```
<script id= "vis" type="text/javascript" src="{% static 'exercice/js/vis.min.js' %}"></script>
```

    – le script permettant la gestion de l'exercice :

```
<script type="text/javascript" src="{% static 'exercice/js/testVis.js' %}"></script>
```

    – L'appel à un fichier css de vis pour afficher le graphe :

```
<link href="{% static 'exercice/css/vis.min.css' %}" rel="stylesheet" type="text/css" />
```

    – un emplacement ou apparaîtra le graphe :

```
<div id="graphe"> </div>
```

    – un emplacement ou apparaîtra la question de l'exercice :

```
<div id="question"> </div>
```

    – un emplacement pour le qcm :

```
<input id="choix1" type="radio" name= "choix" > <span id="choix1S"></span></input>
<input id="choix2" type="radio" name= "choix" > <span  id="choix2S"></span></input>
<input id="choix3" type="radio" name= "choix" > <span  id="choix3S"></span></input>
<input id="choix4" type="radio" name= "choix" > <span  id="choix4S"></span></input>
```

    – un bouton Valider :

```
<button value="Valider" onclick="reponse()"> Valider </button>
```

    – un bouton question suivante :

```
<button value="Question suivante" onclick="runScript()"> Question suivante </button>
```

– (optionnel) des boutons permettant l'appel à des catégories d'exercices :

```
<button value="Random" onclick="runScript('random')"> Aléatoire </button>

<button value="Dijkstra" onclick="runScript('dijkstra')"> Dijkstra </button>

<button value="Bellman-Ford" onclick="runScript('bellmanford')"> Bellman-Ford </button>

<button value="Clique" onclick="runScript('clique')"> Clique </button>

<button value="Acm" onclick="runScript('acm')"> ACM </button>

<button value="Cfc" onclick="runScript('cfc')"> CFC </button>
```

Le script de gestion d'exercice

Le script *exercice.js* commence par faire une requête *AJAX* au serveur à l'adresse :

– "http://127.0.0.1:8000/graph" (changer l'adresse lors du déploiement du projet)

avec un paramètre permettant de demander un type d'exercice précis :

– *"random"* exercice aléatoire
– *"dijkstra"* exercice sur l'algorithme de Dijkstra
– *"bellmanford"* exercice sur l'algorithme de Bellman-Ford
– *"clique"* exercice sur le cardinal de la clique maximale
– *"acm"* exercice sur les arbres couvrants minimaux
– *"cfc"* exercice sur les composantes fortement connexes

Le serveur envoie ensuite l'exercice correspondant au format JSON (cf partie serveur).

Le script affiche les informations sur la page Html aux emplacements prévu à cet effet. (graphe, question, réponse).

Une fois que l'utilisateur à entrer une réponse et cliquer sur le bouton valider:

– si la réponse est fausse, il affiche une *alert* "Mauvaise réponse" et attend une nouvelle réponse de l'utilisateur
– si la réponse est bonne, il affiche une *alert* "Bonne réponse + un complément de réponse si il existe", et affiche ensuite le graphe réponse (nœuds et arcs colorés en fonction de l'exercice et de la réponse)

L'utilisation des boutons de type d'exercices et le bouton suivant ce contente de renvoyer une requête AJAX au serveur avec de nouveau paramètre (ou le paramètre de l'exercice précédent pour "suivant").

Coté serveur

La view *graph(request)* reçoit la requête AJAX et envoie le paramêtre reçu à la fonction *randomGraph(typeExercice)* du fichier *randomGraph.py* qui renvoie un exercice au format JSON.

Le fichier randomGraph.py :

Les fonctions :

- *dijkstraGraph()*
- *negatifDijkstra()*
- *bellmanFord()*
- *cardinalMaxClique()*
- *arbreCouvrantMinimal()*
- *composanteFortementConnexe()*

renvoie chacune un type d'exercice différents.

La fonction *randomGraph(typeExercice)* permet de faire appel à une de ces fonctions en fonction du paramètre *typeExercice* ou alors d'en appeler une de façon aléatoire.

Chaque fonction de création d'exercice doit respecter plusieurs conditions sur l'objet rendu.
Un objet de forme JSON possédant ses variables :

- *"question": "string"*
  // exemple : "question": "Combien de composantes fortement connexes possède ce graphe ?"

- Les données d'un graphe au format Json (gérer par *networkX* dans la fonction graphToJson(G)) comportant généralement les variables *"multigraph"*, *"graph"*, *"nodes"*, et *"links"*
  //exemple : *"multigraph": "False"*,
  "graph": {},
  "nodes": [{"id": 0}, {"id": 1}, {"id": 2}, {"id": 3}, {"id": 4}, {"id": 5}, {"id": 6}, {"id": 7}],
  "links": [{"id": 0, "source": 0, "target": 6}, {"id": 1, "source": 1, "target": 2}, {"id": 2, "source": 6, "target": 4}, {"id": 3, "source": 7, "target": 1}]

  Attention : *NetworkX* ne crée pas un format json correct il est nécessaire de remplacer les guillemets simple " ' " par des doubles grâce à la fonction *graphToJson(G)*.

- *"ponderate": "True/False"* //si le graphe est pondéré ou non

- *"directed": "True/False"* //si le graphe est dirigé ou non

- *"true_answer": "String/Tableau de noeuds/liste d'arcs/etc …"*
  //exemple : *"true_answer": "A-D-E",*

- *"wrong_answer": "trois String séparé par le caractère 'z' "*
  //exemple : *"wrong_answer": "A-B-C-D-EzA-B-C-D-F-Ez-A-B-C-H-D-E",*

  NB : le caractère séparateur z peut être modifié si une réponse comporte ce caractère.

- *"complementreponse": "String apportant une information à la bonne réponse/None"*

- *"colorbase": "Tableau de noeuds a coloré à l'affichage de la question/None"*
  //exemple : "colorbase": "{0,4}",

- *"colorreponse": " {un autre format json comportant les variables "nodes" et "edges"}*
  //exemples : "colorreponse": {

"nodes": "{yellow,4,red,6,green,0,orange,2,pink,1,purple,3,white,5,brown,7}",
"edges": "{red,0,4,blue,5,2}"},

Si une couleur (en brut ex : "red", ou en hexadécimal) est fournit devant un ou plusieurs id d'un nœud ou arcs, ils seront colorer de la couleur définit.

Exemple complet :


```
{
	"question": "Combien de composantes fortement connexes possède ce graphe ?",
	"ponderate": "False",
	"colorbase": "None",
	"colorreponse": {
	"nodes":
	"{yellow,0,#ff0210,2,#1AD723,3,orange,5,pink,4,#b302d7,7,white,1,#45a9d7,6}"
	},
	"complementreponse": "None",
	"directed": "True",
	"multigraph": "False",
	"graph": {},
	"nodes": [{
		"id": 0
	}, {
		"id": 1
	}, {
		"id": 2
	}, {
		"id": 3
	}, {
		"id": 4
	}, {
		"id": 5
	}, {
		"id": 6
	}, {
		"id": 7
	}],
	"links": [{
		"id": 0,"source": 1,"target": 2
	}, {
		"id": 1,"source": 1,"target": 4
	}, {
		"id": 2,"source": 1,"target": 7
	}, {
		"id": 3,"source": 2,"target": 0
	}, {
		"id": 4,"source": 4,"target": 2
	}, {
		"id": 5,"source": 4,"target": 3
	}, {
		"id": 6,"source": 4,"target": 5
	}, {
		"id": 7,"source": 6,"target": 5
	}, {
		"id": 8,"source": 7,"target": 2
	}],
	"true_answer": "8",
	"wrong_answer": "1z10z5z"
}
```

Conclusion

Pour la création de nouveaux exercices il est donc nécessaire de créer une nouvelle fonction dans le fichier *randomGraph.py* générant un exercice mis sous la forme vue précédemment, puis d'ajouter un appel à cette fonction dans la fonction *randomGraph(typeExercice)* lié à un mot clé. Une fois cela effectué il faut modifer le script et la page Html pour gérer son appel via la requête AJAX.

Amélioration possible :

- Utilisation de variable globales ou paramètre de fonctions pour la création des graphes, non utilisé ici (valeur écrites en "dur") pour créer des exercices d'un niveau de difficultés constant.

  Exemple 1 : pour l'instant les questions sur Dijkstra et Bellman-Ford demanderont toujours le plus court chemin entre le nœud A et le nœud E.

  Exemple 2 : la question sur les composantes fortement connexes comportera toujours 8 nœuds pour ne pas avoir à gérer la génération de plus de 8 couleurs différentes.

- Faire une deuxième requête AJAX au serveur pour obtenir la bonne réponse, si on observe la réponse actuelle du serveur avant de répondre à l'exercice on peut y lire la bonne réponse.

- Création de niveau de difficultés pour les exercices (plus de nœuds, d'arcs, changement de parcours, etc ...)

- S'assurer que les réponses proposés dans le QCM ne soit pas identiques (peut arriver très rarement pour les questions sur les composantes fortement connexes et l'arbre couvrant minimal.

- Gestion d'un système de score avec le script.

- Coloration en rouge du cycle négatif lorsqu'il est impossible d'utiliser l’algorithme de Bellman-Ford.

- Possibilité qu'un cycle négatif ne soit pas relié au graphe principal mais génère quand même la réponse "Utilisation de l’algorithme impossible" pour l'exercice sur Bellman-Ford.