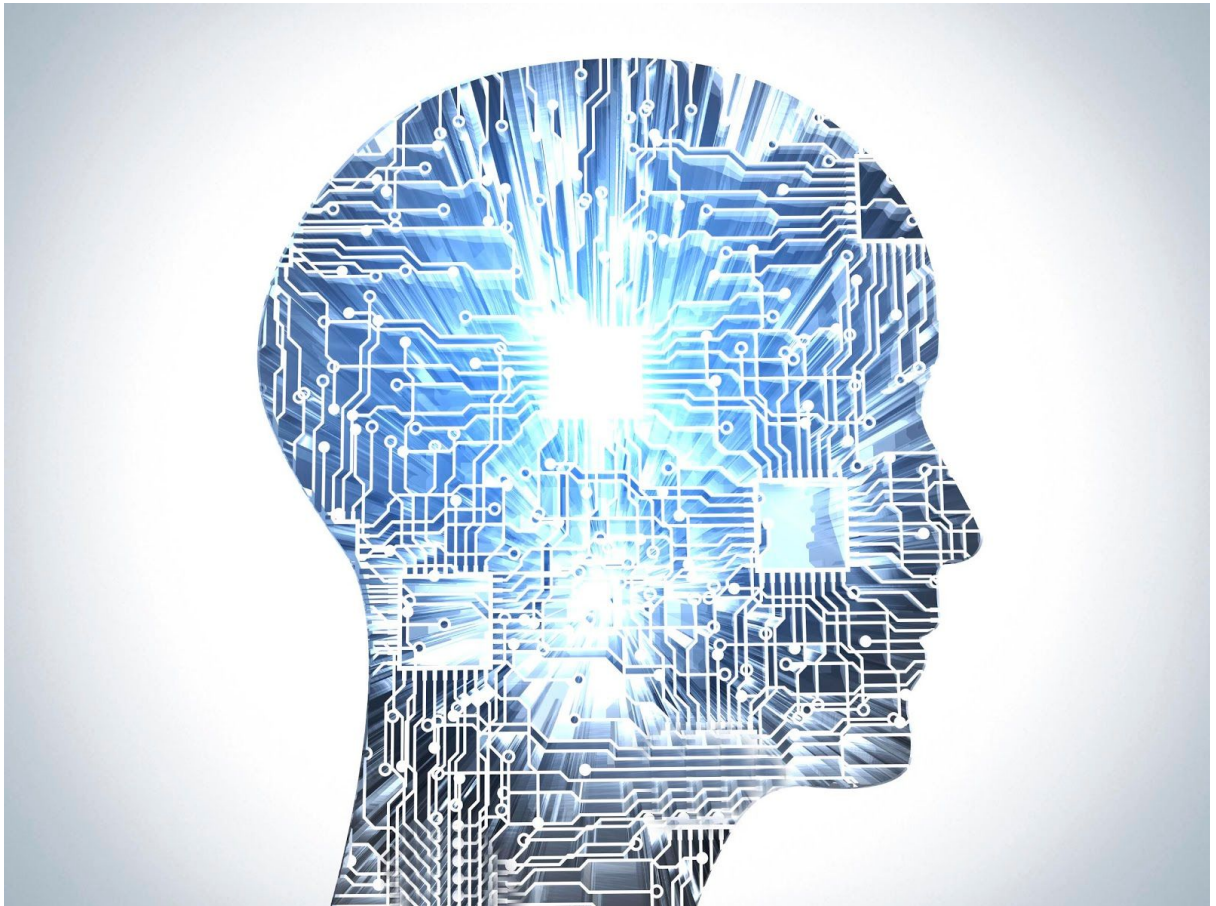

Rapport projet

IA pas de problème

Intelligence Artificielle



BERNARD Vincent	THIOLIERE Elise
-----------------	-----------------

Table des matières

Introduction	2
Gestion de projet	3
Outils utilisés	3
Répartition du travail effectué	3
Travail effectué	4
Le QCM	5
<i>Formulaire(s) du QCM</i>	5
<i>Création de la liste de questions</i>	9
L'Algorithme de Dijkstra	9
<i>Exercice étape par étape (Ouverts/Fermés)</i>	12
<i>Exercice pour compléter l'arbre</i>	16
Pistes d'amélioration	18
QCM	18
Algorithme de Dijkstra	19
Fonctionnalités respectées	19
Conclusion	20
Annexes	21
Liste des questions	21
Sauvegarde des graphes	25

I. Introduction

Considéré comme « le grand mythe de notre temps » par la CNIL, les mots “Intelligence Artificielle” reviennent sur toutes les bouches depuis plusieurs décennies. Notion décriée par certains prophétisant un futur sombre tandis que d’autres y voient le moyen d’atteindre un nouvel Âge d’or. Concept peu précis, il est nécessaire de distinguer les différents thèmes qui s’y recoupent pour plus de précision. Ainsi, il est d’usage de distinguer l’IA faible de l’IA forte. La première correspond à une approche pragmatique visant à l’automatisation des systèmes, c’est à dire que la machine simule l’intelligence, tandis que l’autre n’est que pure chimère pour la plupart, puisqu’elle implique une conscience et une personnalité. A la croisée de la neurobiologie computationnelle, de la logique mathématique et de l’informatique, on la classe généralement dans le domaines des sciences cognitives.

D’où la présence de cours d’IA dans la formation de l’ingénieur cognitif ; il s’agit donc ici de découvrir le domaine de l’Intelligence Artificielle et son utilisation dans la résolution de problème. L’objet de ce projet est donc de renforcer nos connaissances sur la notion et d’appliquer des algorithmes d’IA sur la résolution d’un problème : trouver le plus court chemin dans un graphe. Le travail produit est composé d’une solution WinForm dans laquelle est présent un Questionnaire à Choix Multiples sur l’IA où l’utilisateur peut tester ses connaissances sur le sujet. Une partie centrée sur l’algorithme de Dijkstra est également disponible et vise à évaluer de manière spécifique la connaissance de l’algorithme en proposant à l’utilisateur de faire tourner Dijkstra à la main pour trouver le plus court chemin dans un graphe non orientée.

Nous détaillerons dans ce rapport la manière dont nous avons réalisé le projet, en commençant par une partie gestion projet, laquelle est suivie de la description du travail effectué. Nous avons décidé d’y adjoindre les pistes d’amélioration dans l’optique d’améliorer la facilité et l’expérience d’usage. Enfin, le dernier moment de ce rapport établit un bilan du projet avec une matrice des fonctionnalités de la solution et la liste des questions susceptibles d’être posées dans notre QCM.

N.B : Pour lancer notre lancer la solution : *ProjetIA* avec *Partiel* comme projet de démarrage.

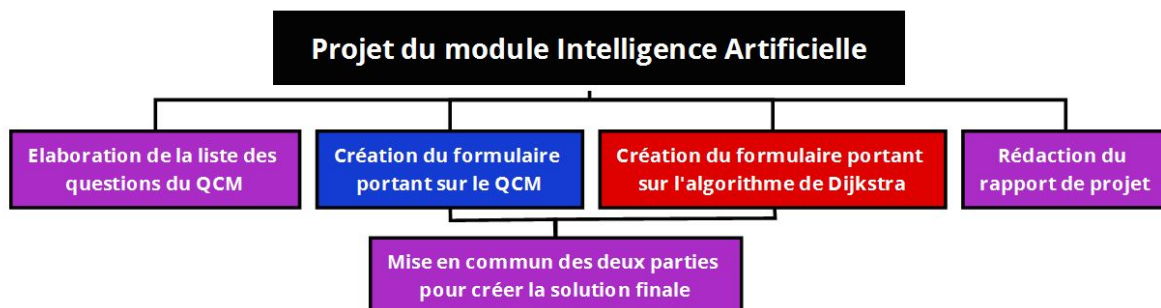
II. Gestion de projet

A. Outils utilisés

Pour piloter ce projet de manière efficace, nous avons décidé de diviser le travail en deux parties, celles déjà sous entendues par le sujet. Une première partie concernant le QCM et une deuxième basée sur l'application manuelle de l'algorithme de Dijkstra. La communication au sein d'une équipe est primordiale et elle l'est d'autant plus lorsque le travail est clairement scindé. L'utilisation d'un outil tel que GitHub facilite le suivi du projet ainsi que la mise en commun du code (les modifications sont mises en valeur). l'outil Google Drive a été privilégié pour les transferts de la documentation écrite.

Compte tenu de la courte durée du projet, nous avons décidé de ne pas élaborer de diagramme de GANTT. En effet, nous avons avancé en parallèle sur l'élaboration du code. Nous avons ensuite travaillé en pair-programming, pour le rassemblement de nos travaux respectifs et la finition du code. La rédaction du livrable a été réalisée ensemble.

B. Répartition du travail effectué



Légende :

- Travail effectué par les deux membres du groupes
- Travail effectué par Elise THIOLIERE
- Travail effectué par Vincent BERNARD

Figure 1 : Mind map présentant la répartition des tâches au sein de l'équipe

La répartition du travail effectué s'est fait comme synthétisé ci-dessus, c'est-à-dire que :

- le questionnaire du QCM, comprenant 23 questions disponibles en annexe (Annexe1), a été réalisé conjointement par les deux membres.
- la création d'un formulaire dans lequel l'utilisateur peut répondre à un QCM de 20 questions a été faite par Vincent Bernard.
- la création d'un formulaire posant des questions plus spécifiques à l'algorithme de Dijkstra a été effectué par Elise Thiolier.

- la mise en commun des deux projets pour créer la solution finale, ainsi qu'un formulaire d'accueil faisant le lien entre ceux portant sur le QCM et l'algorithme de Dijkstra a été menée ensemble, par les deux membres de l'équipe,
- enfin, le ci-présent rapport de projet a été rédigé par les deux membres, chacun s'occupant davantage de la partie qu'il/elle a réalisée.

III. Travail réalisé

Le travail produit est donc une solution comprenant plusieurs formulaires.

Lorsque le projet est lancé un formulaire d'accueil se lance. Il est alors possible de choisir entre deux exercices différents :

- Un QCM, regroupant 20 questions sur l'Intelligence Artificielle visant à tester les connaissances du joueur dans ce domaine. Le score est calculé automatiquement.
- Une application manuelle de l'algorithme de Dijkstra. Cette application peut se faire de deux manières. En remplissant les étapes une par une, ou bien sous la forme d'un arbre à compléter.



Figure 2 : Interface de démarrage de l'application

Pour cette application, nous avons considéré que l'approche pédagogique était très importante. C'est pourquoi nous avons fait le choix de séparer les deux exercices de manière distincte. En effet, il est possible pour un utilisateur d'adapter son apprentissage en fonction de ses lacunes. Si un joueur connaît parfaitement son cours, mais ne comprend pas le déroulement de l'algorithme, il lui est possible de travailler uniquement cette partie, et

inversement. S'il n'est à l'aise sur aucune des deux parties, il n'a qu'à faire les deux à la suite en commençant par celle qu'il souhaite.

A. Le QCM

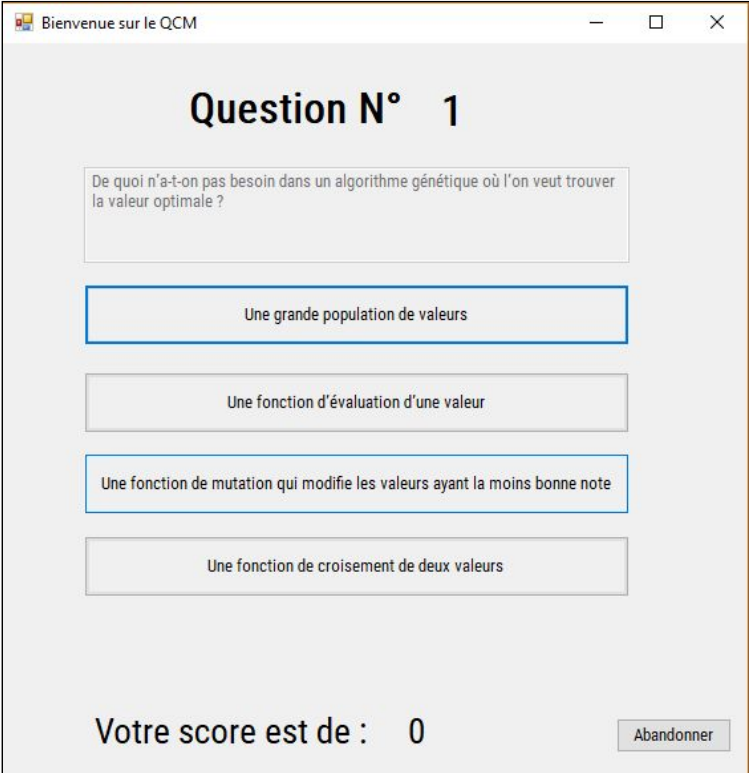
1. Formulaire(s) du Questionnaire à Choix Multiples

Dans cette première partie, nous avons choisi de proposer des questions avec quatre propositions de réponses dont une seule est valide. Nous voulions créer un QCM simple d'utilisation et très visuel. En effet, l'enjeu était pour nous d'afficher la correction de la question de la manière la plus parlante et directement à la fin de la question (et non à la fin des 20). Cela permet à l'utilisateur de bien avoir en tête le sujet et l'énoncé, lui permettant ainsi de comprendre plus facilement son erreur dans le cas où il se trompe. Aussi, lorsque l'utilisateur sélectionne une réponse, nous avons opté pour une correction avec un code couleur très simple :

- La proposition de réponse devient verte si elle est juste ...
- ... et rouge sinon.

Dans le cas où la réponse n'est pas correcte, la bonne s'affiche en verte, afin d'avoir une correction instantanée. Avec ce type d'interface, il aurait été beaucoup moins instinctif pour le joueur s'il avait eu à cocher plusieurs puis à valider.

Après avoir cliqué sur le bouton "QCM" du formulaire d'accueil, le formulaire ci-dessous s'ouvre :



The screenshot shows a web application window titled "Bienvenue sur le QCM". The main heading is "Question N° 1". Below it is a text box containing the question: "De quoi n'a-t-on pas besoin dans un algorithme génétique où l'on veut trouver la valeur optimale ?". There are four buttons below the question, each representing a possible answer. The first button, "Une grande population de valeurs", is highlighted with a blue border. The other three buttons, "Une fonction d'évaluation d'une valeur", "Une fonction de mutation qui modifie les valeurs ayant la moins bonne note", and "Une fonction de croisement de deux valeurs", have grey borders. At the bottom of the window, it says "Votre score est de : 0" and there is an "Abandonner" button.

Figure 3 : Page type d'arrivée sur le formulaire du QCM

Le formulaire affiche directement une question dans la case située sous le numéro de la question. Les quatre options de réponse s'affichent dans des boutons placés en colonne sous l'intitulé de la question. Le score s'affiche tout en bas du formulaire. Cela permet au joueur de savoir où il en est. Il peut ainsi comparer son score au numéro de la question en cours : chaque bonne réponse rapporte un point.

Nous avons placé un bouton en bas à droite qui permet à l'utilisateur de fermer le formulaire à tout moment, s'il veut arrêter de répondre au QCM. Ce faisant, ce dernier disparaît et l'utilisateur revient alors au formulaire d'accueil. Nous avons choisi un bouton de petite taille pour remplir cette fonction, car bien que nous voulons laisser la possibilité à l'utilisateur de stopper le contrôle de ses connaissances, nous ne voulons pas l'inciter à ne pas aller au bout du questionnaire.

Lorsque l'on clique sur une réponse, comme précisé ci-dessus, le bouton contenant la proposition choisie change de couleur passant au vert pour une bonne réponse ou au rouge pour une mauvaise (et la bonne s'éclaire), comme on peut le voir ci-dessous :

Question N° 1

Qu'est ce que "l'espace d'états" ?

L'ensemble des positions que l'on peut attendre au prochain coup

L'ensemble des positions que l'on peut attendre depuis l'état initial avec un nombre de coups illimités

L'ensemble des positions initialement connues

L'ensemble des positions effectivement atteintes lorsque l'on est arrivé à l'état final

Question Suivante

Votre score est de : 1

Abandonner

Figure 4 : Affichage d'une bonne réponse

Question N° 2

Quelle est la différence entre un algorithme MinMax et un algorithme Alpha Beta ?

L'algorithme MinMax est plus rapide car plus optimisé

Alpha Beta évite le développement de branches inutiles

Il n'y aucune différence : les deux noms désignent le même type d'algorithme

L'algorithme Alpha Beta n'existe pas

Question Suivante

Votre score est de : 1

Abandonner

Figure 5 : Affichage d'une mauvaise réponse et de sa correction

Pour l'utilisateur, la correction est donc instantanée. De plus quand l'utilisateur répond et que la correction s'affiche, un bouton permettant de passer à la question suivante apparaît. Ce dernier était auparavant placé en dessous du score mais cette position amplifiait alors le mouvement à faire pour répondre à la question suivante. Nous avons donc inversé les positions de ces deux éléments pour une question ergonomique. Ajoutons à cela qu'une fois que l'utilisateur a répondu à une question, les boutons des quatre propositions sont désactivés pour éviter l'appel de fonction inutile et la tricherie.

Lorsque l'utilisateur arrive à la question n°20, le bouton question "Question Suivante" est remplacé par un bouton "Résultat" :

Question N° 20

Dans un graphe, quelle est la différence entre un arc et une arête

Aucune, les deux termes désignent la même chose

L'arête désigne une relation dans un graphe orienté alors que pour l'arc le graphe est non orienté

L'arête désigne une relation dans un graphe non orienté alors que pour l'arc le graphe est orienté

Un arc relie trois points alors qu'une arête en relie deux

Résultats

Votre score est de : 17

Abandonner

Figure 6 : Affichage du formulaire lors de la dernière question

En cliquant sur ce bouton, le formulaire contenant les questions du QCM se ferme et s'ouvre un formulaire présentant les résultats de l'utilisateur sous forme d'une note sur 20. Un bouton permet de fermer la fenêtre "Résultats" et de revenir à l'écran d'Accueil.

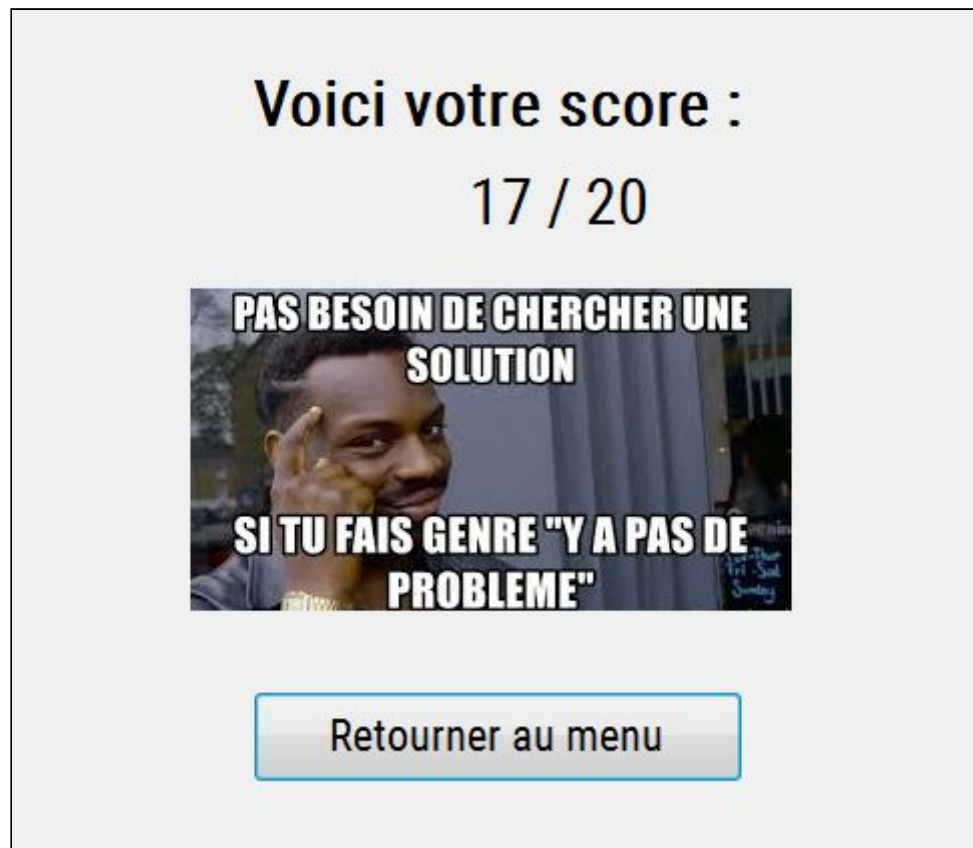


Figure 7 : Affichage du résultat du QCM

2. La liste de question

Le QCM présente à l'utilisateur une série de 20 questions. Ces questions sont sélectionnées de manière aléatoire dans une liste de 23 questions, stockées sous la forme d'un fichier xml nommé *questionnaire.xml*. Le fichier est généré à partir d'un projet de la solution, nommé "CréationQuestionnaire" qu'il faut lancer si le fichier n'existe pas ; un utilisateur peut donc l'utiliser pour ajouter des questions au QCM. Il peut également ajouter des questions directement dans le fichier *questionnaire.xml* lorsque celui-ci existe, situé dans *ProjetIA/Partie1/bin/Debug/*, et peut le faire à tout moment, y compris lorsque l'application est lancée. La nouvelle liste de questions sera alors chargée lors du prochain lancement d'un QCM, c'est-à-dire la prochaine fois qu'il cliquera sur le bouton "QCM" du formulaire d'Accueil.

B. L'algorithme de Dijkstra

Lorsque l'utilisateur choisit l'exercice sur l'algorithme de Dijkstra, celui-ci démarre dans un formulaire annexe. Nous avons choisi de privilégier l'ouverture de ce nouveau formulaire afin que l'utilisateur puisse revenir à n'importe quel instant sur le menu de démarrage, lui laissant alors une plus grande flexibilité.

Lorsque le joueur clique sur le bouton “Dijkstra” dans le formulaire d’accueil, cette interface suivant se présente à lui :



Figure 8 : Interface d’accueil de l’exercice de Dijkstra

A ce stade, l'utilisateur choisit le graphe avec lequel il souhaite travailler. Il nous a semblé important qu'au démarrage le joueur est uniquement les informations visuelles essentielles. Une fois le graphe sélectionné, les noeuds de départ et d'arrivée sont affichés de manière aléatoire. En cas d'égalité lors du tirage aléatoire, le noeud d'arrivée est modifié de manière à ce qu'il soit différent. Le graphe est généré automatiquement et est affiché dans le rectangle blanc sous forme écrite. Sa visualisation spatiale apparaît sous forme d'une image à droite. Une fois de plus, l'information visuelle apportée à l'utilisateur est limitée. Son attention est alors portée sur le graphe et les deux points à joindre.

L'utilisateur se trouve alors face à l'interface ci-contre :

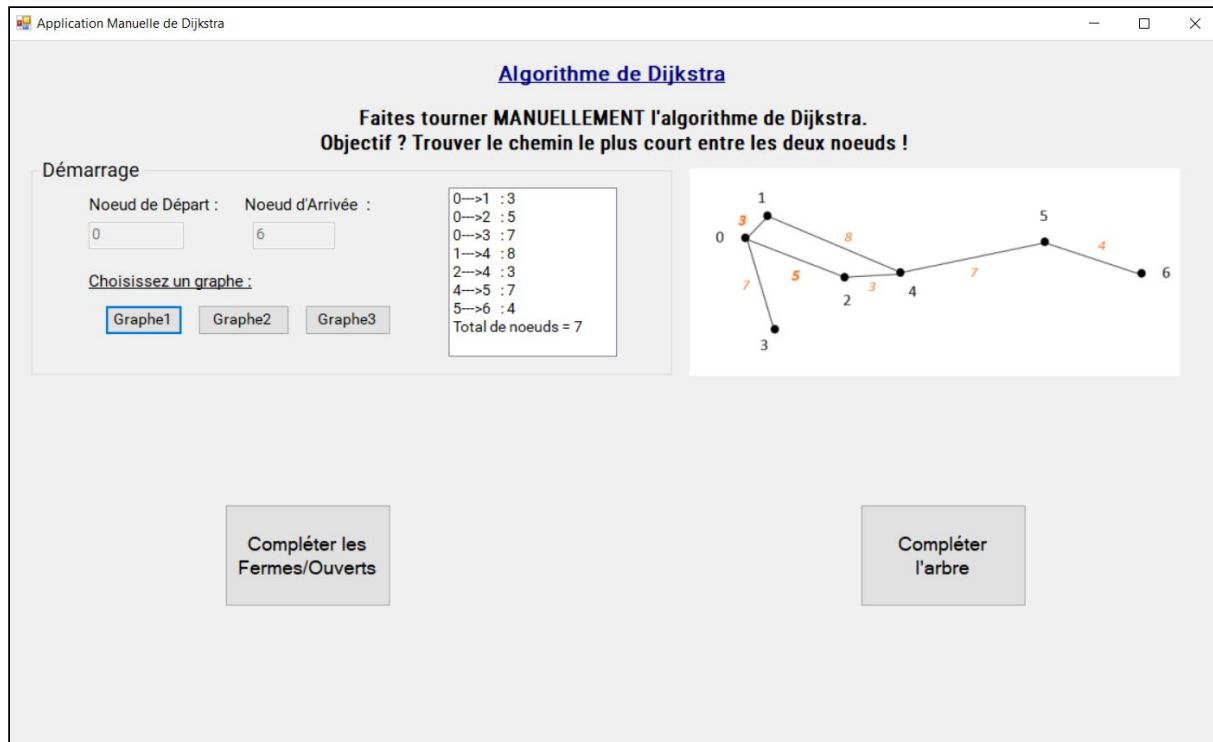


Figure 9 : Interface suite au choix du graphe, ici le graphe n°1

Une fois le choix effectué, deux possibilités s'offrent à l'utilisateur. Dans les deux cas il lui sera nécessaire de faire tourner manuellement l'algorithme de Dijkstra.

- Le joueur peut décider de compléter le domaine des fermés et des ouverts étape par étape.
- Le joueur peut décider de compléter un arbre, dont la structure existe déjà, en indiquant le numéro des noeuds correspondant à chaque emplacement libre.

1. Exercice étape par étape (Ouverts et Fermés)

Lorsque l'on choisit d'effectuer l'exercice étape par étape, on obtient l'interface ci-contre :

Figure 10 : Interface pour l'exercice étape par étape.

Dans cet exercice, des zones de texte affichent respectivement le domaine des fermés et des ouverts. Pour ajouter une étape, il suffit de saisir le numéro des noeuds dans les champs prévus à cet effet. Afin de connaître les contraintes de saisie, il est possible de lire les consignes de l'exercice.

Figure 11 : Consignes pour l'exercice étape par étape.

Lorsque le joueur a pris connaissance des consignes, il peut commencer à jouer. En cas de mauvaise saisie, l'utilisateur reçoit un message d'alerte lui rappelant les règles à suivre. Il peut et doit modifier sa saisie jusqu'à ce qu'elle soit conforme. Ce contrôle nous a semblé essentiel puisqu'il permet à l'utilisateur d'éviter toutes fautes de frappe, qui auraient pu engendrer des erreurs involontaires.

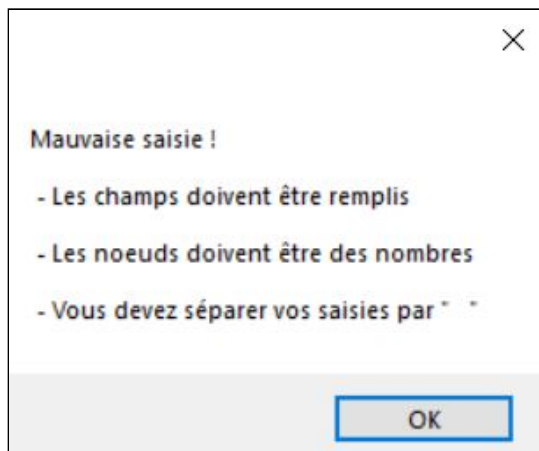


Figure 12: Message en cas de mauvaise saisie pour l'exercice étape par étape.

Nous avons volontairement décidé de ne pas faire de test sur la longueur de la saisie, dans le cas où les graphes comprendraient des nombres à plusieurs chiffres.

Si la saisie est conforme, l'utilisateur peut enregistrer son étape en cliquant sur étape suivante. Ce bouton est en vert afin d'utiliser les normes selon lesquelles le vert correspond à la couleur de la validation.

Lorsque de cette étape, la saisie du joueur s'ajoute automatiquement dans les zones de texte regroupant l'ensemble des étapes des fermés et des ouverts. En parallèle, le clic sur ce bouton, déclenche la comparaison entre les deux listes (Ouverts/Fermés) comprenant les étapes du joueur et les deux listes comprenant les résultats. On a donc un résultat étape par étape, nous indiquant ce qui est juste et ce qui est faux. Afin de s'affranchir de l'ordre de saisie, ces listes sont triées par ordre croissant avant comparaison.

On obtient alors l'affichage suivant :

Figure 13 : Interface de l'exercice étape par étape après ajout de la première étape.

Sur la Figure 13 plusieurs cas essentiels sont représentés. Tout d'abord, on remarque que l'étape des fermés est fausse, un message l'indiquant est affiché. En revanche, l'étape dans les ouverts correspond à la correction. On remarque que si l'algorithme est rigoureusement appliqué, l'ordre des noeuds sur la Figure 13 n'est pas correct. L'IA nous indique pourtant que l'étape est juste. Cela illustre le fait que l'ordre de saisie n'est pas pris en compte. L'affichage de la correction étape par étape change à chaque tour en fonction des données saisies. Une fois de plus, nous avons choisi d'utiliser le code couleur commun à tout utilisateur :

- Vert pour les étapes correctes
- Rouge pour les étapes fausses

Lorsqu'il pense avoir tout rempli, il demande à voir la correction globale. Il obtient alors l'interface suivante :

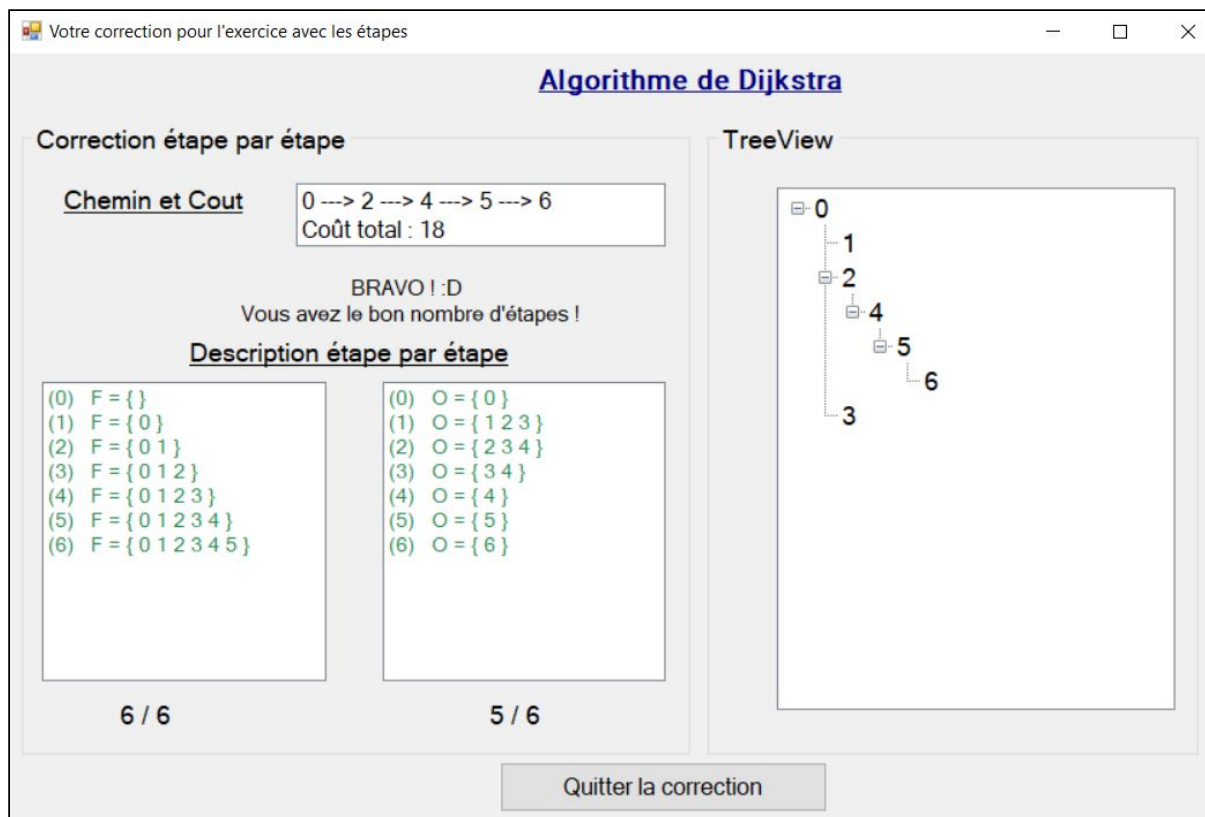


Figure 14.: Interface du formulaire de correction pour l'exercice étape par étape.

A l'ouverture de la correction, le joueur voit directement le chemin final ainsi que le coût total. Elle apparaît dans un nouveau formulaire, afin que l'utilisateur puisse comparer l'ensemble des étapes qu'il a saisi avec les étapes du corrigé. Si l'utilisateur n'a pas trouvé le bon nombre d'étape, un message lui indique pour qu'il soit capable de le refaire. Le nombre de réponses justes, autrement dit son score est également affiché pour le domaine des fermés et des ouverts.

Il peut également consulter l'arbre, qui est affiché directement ouvert grâce à la fonction ExpandAll(). Nous avons choisi d'afficher l'arbre également dans cette correction pour des raisons pédagogiques. En effet, si l'utilisateur ne comprend pas l'ordre selon lequel il doit remplir un arbre, il lui est possible de faire un parallèle avec le remplissage du domaine des ouverts et des fermés,, lui donnant ainsi de nouvelles clés pour maîtriser cette approche graphique.

Une fois ses résultats obtenu, le joueur quitte l'affichage du score et la partie. Il revient alors au menu principal.

2. Exercice pour compléter l'arbre

Lorsque l'on choisit d'effectuer l'exercice de l'arbre, on obtient l'interface ci-contre :

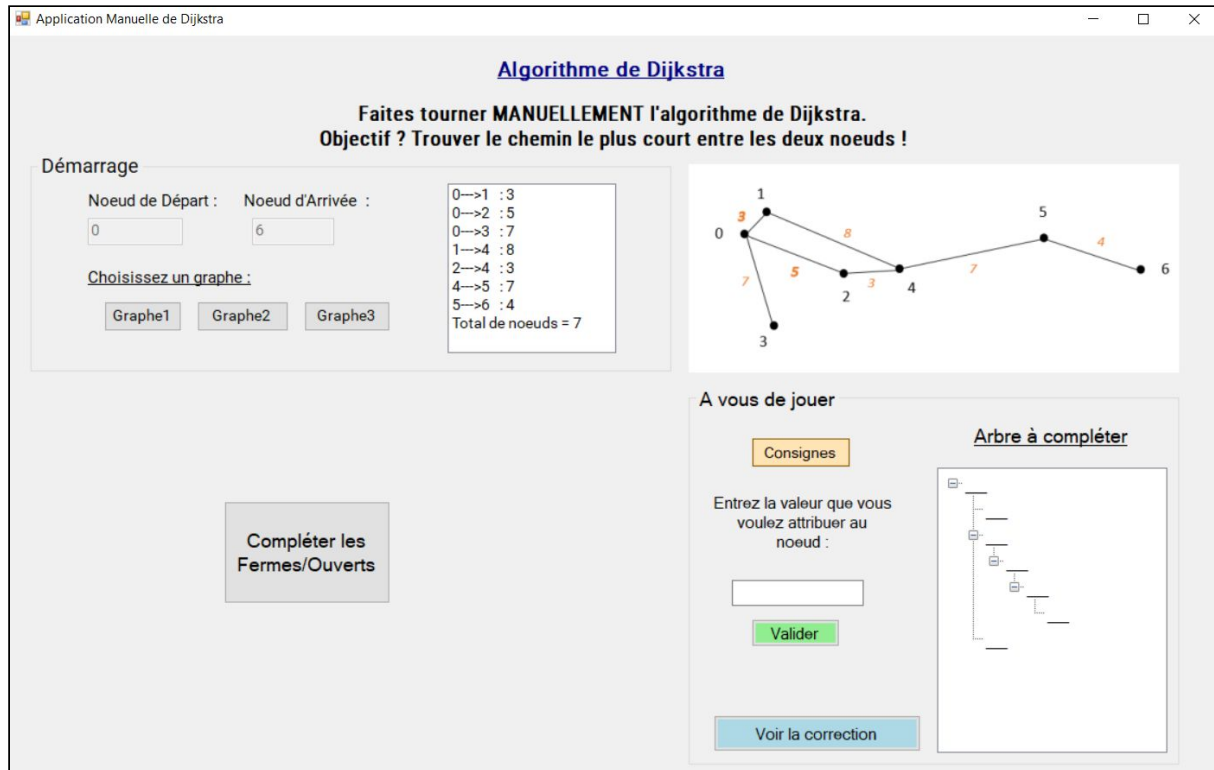


Figure 15 : Interface de l'exercice de l'arbre

Dans cet exercice, un arbre structuré et déjà ouvert (utilisation de la fonction `ExpandAll()`) s'affiche. Pour compléter un noeud, il suffit de saisir le numéro du noeud que vous souhaitez ajouter à l'arbre dans le champ prévu à cet effet. Sélectionnez ensuite sur l'arbre le noeud auquel vous souhaitez attribuer la valeur saisie au préalable. Cliquez ensuite sur Valider. On retrouve également une fois la coloration verte du bouton. Afin de connaître le déroulement de l'exercice, il est possible de lire les consignes de l'exercice.

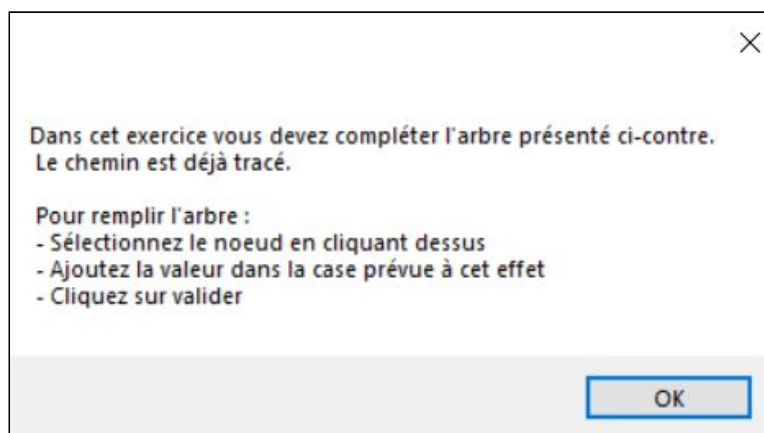


Figure 16 : Consignes pour l'exercice de l'arbre.

Lorsque le joueur a pris connaissances des consignes, il peut commencer à jouer. En cas de mauvaise saisie, l'utilisateur reçoit un message d'alerte lui rappelant les règles de saisie à suivre. Il peut et doit modifier sa saisie jusqu'à ce qu'elle soit conforme. Tout comme pour le premier exercice, ce contrôle nous a semblé essentiel puisqu'il permet à l'utilisateur d'éviter toutes fautes de frappe, qui auraient pu engendrer des erreurs involontaires.

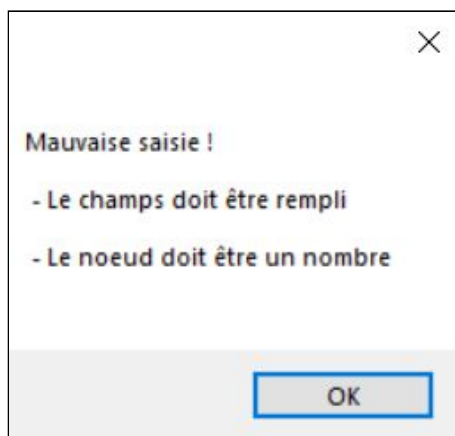


Figure 17 : Message en cas de mauvaise saisie pour l'exercice de l'arbre

Nous avons volontairement décidé de ne pas faire de test sur la longueur de la saisie, dans le cas où les graphes comprendraient des nombres à plusieurs chiffres.

Lorsque la totalité de l'arbre est rempli, le joueur demande à voir la correction, un nouveau formulaire s'ouvre. Une fois de plus, ce choix est volontaire. En effet, nous avons les méthodes permettant de comparer l'arbre du joueur et l'arbre de correction affichent un résultat visuel :

- Les étapes différentes, à savoir fausses ou vides, sont colorées en rouge sur les deux arbres.
- Les étapes identiques, donc correctes, restent en noir sur les deux arbres.

Pour que l'utilisateur puisse comparer les deux arbres, il était essentiel que les deux arbres soient affichés en même temps. Cela n'aurait pas été le cas si nous avions remplacé le formulaire de l'exercice (là où il a rempli son arbre) par l'affichage de la correction.

Sur la Figure 18, le formulaire de correction de l'arbre est placé à côté du formulaire de l'exercice, ce qui nous donne l'interface ci-contre :

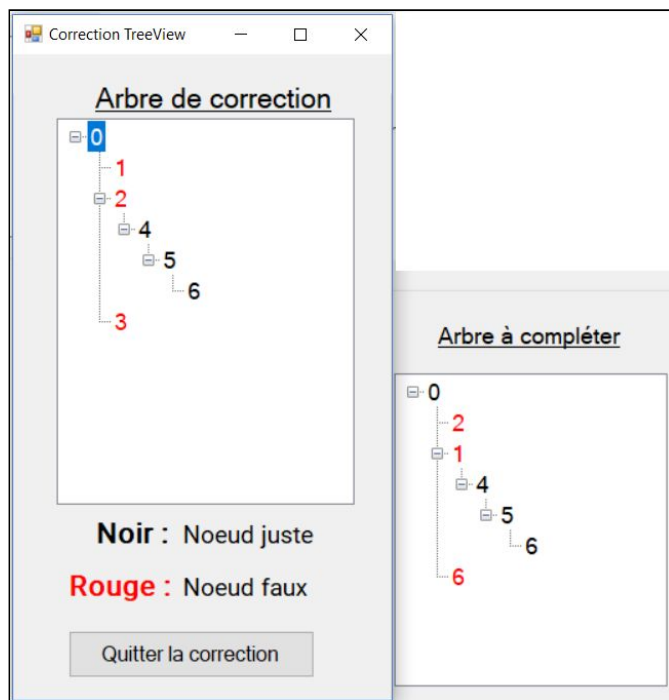


Figure 18 : Correction pour l'exercice de l'arbre.

Une fois ses résultats obtenu, le joueur quitte la correction et la partie. Il revient alors au menu principal.

IV. Pistes d'amélioration

A.QCM

Action	Bénéfices
Modifier la classe Question et le formulaire (modification de Correction(), ajout d'un bouton "corriger" pour soumettre sa réponse) pour pouvoir avoir plusieurs bonnes réponses	Possibilité de questions plus complexes avec entre 1 et 4 bonne(s) réponse(s)
Ajout de davantage de questions au fichier questionnaire.xml	Avoir une plus grande variété de questions pour tester davantage les connaissances de l'utilisateur et pour qu'il puisse recommencer plusieurs fois le test sans tomber sur les mêmes questions

B. Algorithme de Dijkstra

Action	Bénéfices
Ajout d'un bouton "Annuler l'étape précédente"	Permet plus de souplesse à l'utilisateur
Ajout d'un bouton "Tout recommencer"	Le joueur n'est pas obligé de relancer tout le jeu pour recommencer.

V. Fonctionnalités respectées

Priorité	Fonctionnalité	Respectée
Partie I : QCM		
1	Les questions et réponses doivent être stockées dans un fichier xml.	✓
2	Il est possible de rajouter de nouvelles questions (et réponses), sans avoir besoin de recompiler le programme.	✓
3	A l'exécution, le programme choisit au hasard une question et la présente à l'utilisateur.	✓
4	Le programme évalue la réponse et indique la correction éventuelle.	✓
5	20 questions sont posées par session	✓
6	La note finale est affichée	✓
Partie II : Algorithme de Dijkstra		
1	A chaque étape, le programme vérifie que F et O sont justes, sans tenir compte de l'ordre.	✓
2	Un arbre final déjà bien structuré mais sans information sur les nœuds est proposé	✓
3	Le programme doit tourner sur des graphes différents.	✓

VI. Conclusion

Comme point d'entrée dans le domaine de l'IA, ce projet a été très enrichissant. En effet, il a apporté à l'équipe la pratique des technologies Winform et XML ; nous avons donc pu nous familiariser davantage avec ces outils. De plus, la conception du contenu du projet nous a obligé à retravailler les notions vues en cours, portant sur l'Intelligence Artificielle et son utilisation dans la résolution de problèmes : en particulier sur le fonctionnement de l'algorithme Dijkstra. La création du questionnaire à choix multiples et du formulaire d'application de Dijkstra dans un graphe non orienté nécessitait une bonne compréhension de ces éléments pour pouvoir être mis correctement en application dans la solution produite. Nous avons néanmoins rencontré des difficultés, notamment dans la sérialisation XML et sur la correction par étape de la partie sur Dijkstra mais nous avons fait le nécessaire pour pallier à ces problèmes tout en respectant les exigences du projet.

La solution finale est donc fonctionnelle et permet à n'importe qui de tester ses connaissances sur l'IA et/ou l'algorithme de Dijkstra.

VII. Annexes

Annexe 1 : Liste de questions

Nous avons rédigé 23 questions pour notre QCM, les voici : (la bonne réponse est en gras)

1. De combien de degré de liberté est pourvu une liaison de type “rotule” ?
 - a. 1
 - b. 2
 - c. 3**
 - d. 4
2. Qu'est-ce qui caractérise une IA forte ?
 - a. Une conscience et une personnalité**
 - b. La complexité du programme
 - c. Un algorithme génétique
 - d. Un apprentissage automatique
3. La stratégie FIFO est une stratégie qui...
 - a. Parcourt en longueur d'abord
 - b. Parcourt en profondeur d'abord
 - c. Parcourt en largeur d'abord**
 - d. Parcourt en surface d'abord
4. Quelle est la différence entre un algorithme MinMax et un algorithme Alpha Beta ?
 - a. L'algorithme MinMax est plus rapide car plus optimisé
 - b. Alpha Beta évite le développement de branches inutiles**
 - c. Il n'y aucune différence : les deux noms désignent le même type d'algorithme.
 - d. L'algorithme Alpha Beta n'existe pas.
5. Un arbre d'exploration correspond à :
 - a. L'ensemble des possibilités d'exploration de l'algorithme**
 - b. L'ensemble des possibilités l'algorithme que l'algorithme a exploré
 - c. L'ensemble des possibilités amenant au résultat voulu
 - d. L'ordre dans lequel les possibilités vont être explorées
6. Un des objectifs de l'algorithme de Dijkstra est :
 - a. De permettre la reconnaissance faciale.
 - b. Trouver le plus court chemin entre une état initial et un état final.**
 - c. Trouver la solution mathématique de polynômes du troisième degré.
 - d. Calculer le pourcentage de réussite d'un processus.
7. Le problème de Dijkstra utilise une exploration :
 - a. Longitudinale

- b. Pyramidale
 - c. Radiale**
 - d. En profondeur

- 8. "L'algorithme de Dijkstra prend en compte le contexte ou les informations de haut niveau pour guider la recherche ou décomposer le problème en sous-problèmes". Cette affirmation est :
 - a. Toujours vraie
 - b. Toujours fausse**
 - c. Vraie uniquement si le nombre de successeurs est très grand
 - d. Vraie uniquement si on indique explicitement à l'algorithme de le faire

- 9. Une matrice d'adjacence est :
 - a. toujours symétrique.
 - b. symétrique si le graphe est orienté
 - c. symétrique si le graphe n'est pas orienté.**
 - d. jamais symétrique

- 10. Un graphe non orienté est composé de trois points A, B, C. Il y a une arête uniquement entre A et B, et B et C. La relation entre A et B vaut 2 et la relation entre B et C vaut 1. Quelle valeur y a-t-il dans les cases [A;C] et [C;A] de la matrice d'adjacence ?
 - a. -1**
 - b. 1
 - c. 3
 - d. On a besoin de connaître l'angle ABC pour calculer cette valeur

- 11. Dans quel cas, l'algorithme A* n'est pas équivalent à Dijkstra ?
 - a. Pour la stratégie de classement des nœuds qui sont placés dans les Fermés.
 - b. Pour la stratégie de classement des nœuds qui sont placés dans les Ouverts.**
 - c. Les deux algorithmes sont toujours équivalents.
 - d. Il n'est jamais équivalent à Dijkstra.

- 12. Enlever une heuristique consiste à
 - a. Compliquer la résolution du problème.
 - b. Modifier le sens d'exploration de l'arbre d'exploration.
 - c. Supprimer une contrainte pour simplifier le problème.**
 - d. Enlever une partie des possibilités pour se focaliser sur l'autre partie.

- 13. L'heuristique $h(N)$ permet...
 - a. d'afficher le chemin restant pour atteindre le but
 - b. d'afficher le chemin entre le point de départ et le point d'arrivée
 - c. d'estimer le coût du chemin entre le point de départ et le point d'arrivée.
 - d. d'estimer le coût du chemin restant pour atteindre le but.**

14. Quelle est la formule de Tri des ouverts en fonction de $f(N)$?
 - a. **$f(N) = \text{coût_chemin}(N) + h(N)$**
 - b. $f(N) = \text{coût_chemin}(N) * h(N)$
 - c. $f(N) = (\text{coût_chemin}(N) + h(N))/2$
 - d. $f(N) = h(N) - \text{coût_chemin}(N)$

15. Sous quelle condition A^* garantit que le chemin trouvé est le plus court ?
 - a. **Si $h(N)$ est un minorant du coût du chemin restant réel**
 - b. Si $h(N)$ est un majorant du coût du chemin restant réel
 - c. Si $h(N) = 0$
 - d. Si $h(N)$ est un entier.

16. Quel est le point fort d'un algorithme génétique ?
 - a. Le code évolue de lui-même
 - b. **Il permet d'effectuer de multiples recherches simultanées**
 - c. Il est facilement applicable à une situation présentant de nombreux variables
 - d. Il s'adapte à son environnement

17. On pose $h(N)=0$ pour tout N , on a alors
 - a. **A^* équivalent à Dijkstra**
 - b. On ne peut pas utiliser Dijkstra
 - c. On ne peut pas utiliser A^*
 - d. On doit utiliser MinMax

18. Dans un graphe, quelle est la différence entre un arc et une arête.
 - a. Aucune, les deux termes désignent la même chose
 - b. L'arête désigne une relation dans un graphe orienté alors que pour l'arc le graphe est non orienté
 - c. **L'arête désigne une relation dans un graphe non orienté alors que pour l'arc le graphe est orienté**
 - d. Un arc relie trois points alors qu'une arête en relie deux

19. De quoi n'a-t-on pas besoin dans un algorithme génétique où l'on veut trouver la valeur optimale ?
 - a. Une grande population de valeurs
 - b. Une fonction d'évaluation d'une valeur
 - c. **Une fonction de mutation qui modifie les valeurs ayant la moins bonne note**
 - d. Une fonction de croisement de deux valeurs

20. Quelle est la bonne orthographe :
 - a. Djikstra
 - b. Dijsktra
 - c. Djisktra
 - d. **Dijkstra**

21. Parmi les assertions suivantes, laquelle est un principe de base de la théorie des jeux ?

- a. On considère que l'adversaire est aussi doué que soi-même.
- b. On considère que la victoire réside dans la compréhension des règles du jeu.
- c. On considère que l'adversaire a plus de chance que nous.
- d. **On considère que l'adversaire joue de façon optimale.**

22. Si on compare au "model based" le "data based", on peut dire que ce dernier est

- a. Globalement identique
- b. Complètement différent
- c. Basé sur un faible nombre de données
- d. **Basé sur un grand nombre de données**

23. Qu'est ce que "l'espace d'états" ?

- a. L'ensemble des positions que l'on peut attendre au prochain coup
- b. **L'ensemble des positions que l'on peut attendre depuis l'état initial avec un nombre de coups illimité**
- c. L'ensemble des positions initialement connues

L'ensemble des positions effectivement atteintes lorsque l'on est arrivé à l'état final

Annexe 2 : Fichiers de sauvegarde des graphes

Pour bien vous expliquer comment sont rédigés les fichiers contenant les informations sur un graphe, nous allons prendre l'exemple du premier graphe (*fichier Graphe1.txt*). L'ensemble des fichiers sont stockés sous la forme présentée ci-dessous..

Chaque paquet de trois chiffres correspond à un lien entre deux points. La virgule fait office de séparateur. Le dernier nombre correspond quant à lui au nombre de noeuds du graphe.

Lorsque l'on ouvre le fichier *Graphe1.txt*, on lit :

013,025,037,148,243,457,564,7

Pour chaque paquet, chaque chiffre correspond à une information précise. Sur un paquet de 3 chiffres :

- Le **1er** correspond au numéro du 1er noeud
- Le **2ème** correspond au numéro du 2eme noeud
- Le **3ème** correspond au coût entre les deux noeuds indiqués ci-dessus.

Dans cet exemple, le graphe comporte 7 noeuds au total.

Il est essentiel de noter que pour l'ensemble du code, nous avons établi que les relations dans les deux sens. Autrement dit, pour le premier paquet, l'algorithme prend en compte que le coût entre les noeuds 0 et 1 est identique au coût entre les noeuds 1 et 0.

VIII. Bibliographie

https://www.cnil.fr/sites/default/files/atoms/files/cnil_rapport_garder_la_main_web.pdf