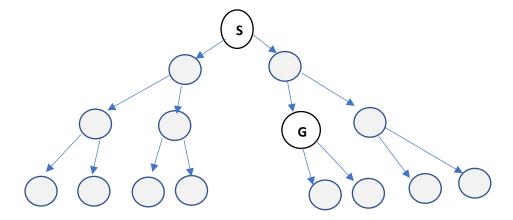
- 1- L'algorithme de recherche en largeur (Breadth First Search) appartient à la classe
 - a. N'importe quel chemin non informé
 - b. N'importe quel chemin informé
 - c. Optimal informé
 - d. Optimal non informé
- 2- L'algorithme de recherche A* appartient à la classe
 - a. N'importe quel chemin non informé
 - b. N'importe quel chemin informé
 - c. Optimal informé
 - d. Optimal non informé
- 3- La différence entre l'algorithme A* et le coût uniforme est que :
 - a. A * est optimal et le coût uniforme ne l'est pas
 - b. Le coût uniforme est non informé et A* est informé
 - c. Les deux sont informé mais A* est optimal
 - d. Les deux sont non optimaux mais le coût uniforme est informé
- 4- La différence entre l'algorithme du coût uniforme avec une liste visitée et l'algorithme du coût uniforme sans liste visitée est que :
 - a. Le coût uniforme sans liste visitée peut trouver une solution optimale
 - b. Le coût uniforme avec liste visitée trouve toujours une solution optimale plus rapidement que le coût uniforme sans liste visitée
 - c. Le coût uniforme avec liste visitée trouve toujours une solution optimale
 - d. Le coût uniforme sans liste visitée trouve toujours une solution optimale
- 5- Soit le graphe de recherche, allant de S et cherchant G dans l'image, cochez tous ce qui est vrai :
 - a. Un algorithme de recherche en profondeur développera plus de nœuds qu'un algorithme de recherche en largeur pour arriver à G
 - b. Un algorithme de recherche en largeur développera plus de nœuds qu'un algorithme de recherche en profondeur pour arriver à G
 - c. Un algorithme de recherche en A* avec la meilleure heuristique peut trouver la solution G en développant au minimum 3 nœuds
 - d. Pour ce type de graphe, un algorithme de recherche en largeur sera toujours meilleur ou égal en termes de performance qu'un algorithme de recherche A* et cela pour n'importe quel heuristique.



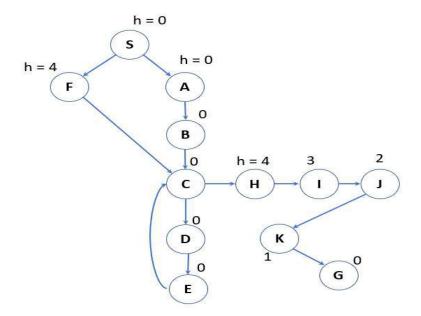
Soit le graphe de recherche dans la figure suivante

Faites les assomptions suivantes :

- La liste visitée ne doit pas être utilisée.
- Tous les liens sont de longueur 1.
- La valeur heuristique d'un nœud à l'objectif est donnée pour chaque nœud.
- Aucun algorithme de recherche ne génère un chemin avec un cycle.
- La recherche en profondeur et la recherche en largeur explore les nœuds par ordre alphabétique.
- Les algorithmes de recherche utilisent une queue.

En cas d'ex-aequo faites les étapes suivantes :

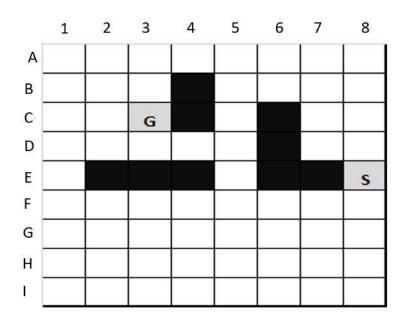
- Prendre, en premier en ordre alphabétique dans le chemin.
- Après, du premier au dernier dans l'ordre de la liste.
- 1- Montrer la séquence des nœuds développés par une recherche en profondeur.
 (1 pt)
- 2- Recherche A*.
 - a. Est-ce que l'heuristique est admissible, expliquez. (1 pt)
 - b. Montrer la séquence des nœuds développés pour la recherche A* (2 pts)
 - c. Quel est le chemin final trouvé par A* (1 pt)



Soit le problème de recherche par un robot d'un chemin dans l'image montré ci-Haut de la **position s (Start)** à la position **g (Goal)**. Le robot peut bouger dans le tableau horizontalement et verticalement, une case à la fois. La zone ombrée est interdite.

Supposons que l'ordre des mouvements est (1) Bas, (2) Gauche, (3) Droit puis (4) Haut en assumant qu'il y a une méthode pour la vérification des cycles ;

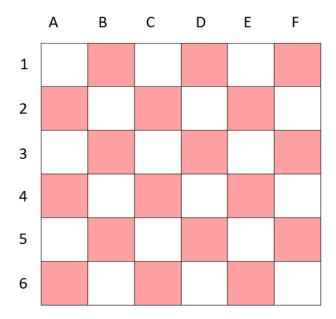
- 1- (1 pt) Appliquer une recherche en profondeur afin de numéroter en ordre les nœuds qui seront développés. (Ecrire seulement les cases à développer par exemple 8F, 8G, 8H, 7H et ainsi de suite ...)
- 2- (2 pts) Appliquer une recherche de meilleur d'abord (Best First) en utilisant l'heuristique de la distance de Manhattan (vu en cours). Quel est le premier chemin trouvé ? (Ecrire seulement les cases à développer par exemple 8F, 8G, 8H, 7H et ainsi de suite ...)



Résoudre le problème de l'affectation des 6 reines sur un échiquier 6x6 en expliquant les affectations faites par l'algorithme de recherche en profondeur (**backtracking**) appartenant à la classe des problèmes de satisfaction de contrainte en expliquant en détail par rédaction écrite les affectations.

Le problème consiste à placer 6 reines sur un 'échiquier 6 × 6 sans que deux d'entre elles ne se menacent mutuellement), en tenant compte du fait qu'il y a exactement une reine par ligne.

- Explication par rédaction écrite le déroulement des affectations de l'algorithme (1 pt)
- 2. Donner une solution pour une affectation si possible, sinon montrer qu'aucune solution n'est possible (1 pt)



Une solution: A3; B6; C2; D5; E1; F4