**Exercice 1**

États : 8 variables (une par élément du taquin) avec pour valeur le couple (i,j) de ses coordonnées sur le tableau et une variable pour le trou (VE), où i Î[1 ;3] et j Î[1 ;3]

État initial : V1=(1,2), V2=(1,1), V3=(3,1), V4=(3,2), V5=(3,3), V6=(2,2), V7=(1,3), V8=(2,1),VE=(2,3)

État but : V1=(1,1), V2=(2,1), V3=(3,1), V4=(3,2), V5=(3,3), V6=(2,3), V7=(1,3), V8=(2,3), VE=(2,2)

H : déplacer vers le haut le trou. Condition : le trou ne doit pas être sur la ligne 1. Dans l’état, c’est la variable représentant la case juste au-dessus qui change de valeur. Plus formellement :

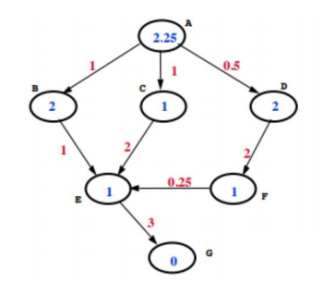
Condition : VE=(C,L) avec L >1 (C = colonne, L = ligne)

Modification de l’état :

-VE [qui était égal à (C,L)] <-VE=(C,L-1)[le « vide » monte d’une ligne, ne change pas de colonne];

-Vk [qui est la pièce qui descend, avait pour valeur (C,L-1)] <-Vk=(C,L [elle descend d’une ligne, ne change pas de colonne]) (les autres variables ne changent pas)

Vous devez examiner l'espace d'état complet sans heuristique. Vous pouvez appliquer un opérateur à chaque état (ou non) et recommencer avec l'état acquis. Si nous arrivons à l'état final, nous avons atteint notre objectif, et la solution est le chemin tracé par une série d'opérateurs à partir de l'état initial. Nous devons continuer à étudier les états de l'espace si nous voulons trouver la meilleure réponse.



* FERMÉ : A (2.25)
* OUVERT : C(2), D(2.5), B(3)
* FERMÉ : C(2), A(2.25)
* OUVERT : D (2.5), B(3), E(4)
* FERMÉ : C(2), A(2.25), D(2.5), B(3), E(3) par B
* OUVERT : F(3.5), G(5)
* FERMÉ : C(2), A(2.25), D(2.5)
* OUVERT : B(3), F(3.5), E(4)
* FERMÉ : C(2), A(2.25), D(2.5), B(3)
* OUVERT : E(3) par B, F(3.5), E(4) par C est éliminé
* FERMÉ : C(2), A(2.25), D(2.5), B(3), E(3) par B, F(3.5)
* OUVERT : E(3.75) par F, mais pas meilleur que E(3) par B de FERMÉ, donc pas de ré-actualisation de E ni donc de G(5). Et comme G(5) est le seul à rester dans OUVERT et que c'est le but, Succès par le chemin A-B-E-G

**Exercice 2**

1. Coût uniforme : S, A, B, C, C, D, D, D, G2
2. Profondeur itérative : S, S, A, C, S, A, B, E, C, D, G2
3. Best First Search (BFS): S, A, B, G1
4. A\*: S, A, B, D, G2