

EFREI - L3 - 2008/2009

Aide à la Décision -- Sujet TP #2

Algorithme de recherche

Révision 1

Partie obligatoire :

Mise en œuvre de l'algorithme de recherche A*.

Domaine d'application : jeu du taquin à 9 cases, état solution « en escargot » comme utilisé en TD.

Fonction heuristique : nombre de pièces mal placées.

Affichage du résultat :

Pour chaque test, votre programme doit afficher les informations suivantes :

- état de départ
- suite des états traversés pour passer de l'état initial à l'état solution
- longueur du chemin (nombre de mouvements)
- nombre d'états dans les ensembles 'ouverts' et 'fermés' au moment de l'atteinte de l'état solution (complexité en espace)
- nombre d'exécution de la fonction de choix d'un état dans 'ouverts' (complexité en temps).

Parties optionnelle :

Adaptation des fonctions 'g' et 'h' pour effectuer une recherche gourmande ou une recherche en coût uniforme.

Utilisation d'autres fonctions heuristiques.

Langage de programmation :

C ou C++.

Fourniture du résultat :

Au plus tard 24 heures après la fin de votre séance de TP.

Consignes impératives de constitution du mail :

Destinataire : *une adresse email qui vous sera communiquée en séance*

Sujet : EFREI L3 AD TP2 – *nom binôme 1 – nom binôme 2*

Message :

résultats d'exécution, sous forme de texte copié / collé depuis une fenêtre d'exécution de votre programme - pas de trace en pièce jointe ; pas d'image.

Pièces jointes :

fichiers code source de votre programme - pas de compression - les noms des fichiers doivent être préfixé par les noms des binômes.

Morceaux de code, à utiliser et/ou adapter si vous le souhaitez...

Description état solution

```
int etat_final [3][3] = {  
    { 1 , 2 , 3 } ,  
    { 8 , 0 , 4 } ,  
    { 7 , 6 , 5 } } ; // Valeur '0' pour absence de pièce
```

Fonction heuristique « nombre de pièces mal placées »

```
int fct_heuristique_1 ( int etat_de_jeu [3][3] )  
// Nombre de pièces mal placées (non compris la case vide)  
{  
    int resultat = 0 ;  
    if ( etat_de_jeu [0][0] != 1 ) { resultat = resultat + 1 ; } ;  
    if ( etat_de_jeu [0][1] != 2 ) { resultat = resultat + 1 ; } ;  
    if ( etat_de_jeu [0][2] != 3 ) { resultat = resultat + 1 ; } ;  
    if ( etat_de_jeu [1][0] != 8 ) { resultat = resultat + 1 ; } ;  
    if ( etat_de_jeu [1][2] != 4 ) { resultat = resultat + 1 ; } ;  
    if ( etat_de_jeu [2][0] != 7 ) { resultat = resultat + 1 ; } ;  
    if ( etat_de_jeu [2][1] != 6 ) { resultat = resultat + 1 ; } ;  
    if ( etat_de_jeu [2][2] != 5 ) { resultat = resultat + 1 ; } ;  
    return resultat ;  
};
```

Recherche des successeurs d'un état

Dans ce code, `ensembles` est un tableau de structures dont le champ `plateau` est un tableau 3x3 d'entiers représentant un plateau de jeu. La variable `etat_en_cours` est l'indice de l'état choisi à chaque itération de l'algorithme de recherche.

La valeur `ensembles[etat_en_cours].plateau` est donc le tableau correspondant à l'état pour lequel on cherche des successeurs. Vous devez adapter ce code en fonction des structures de données que vous utilisez dans votre programme.

```
typedef struct elt_liste_succ {
    bool existe ;
    // Permet de reconnaître les éléments valides du tableau
    int plateau[3][3] ;
    // Description d'un successeur possible
} elt_liste_succ ;
elt_liste_succ liste_des_successeurs [4] ;
// Max 4 successeurs pour le jeu du taquin (cf. champ 'existe')
int nb_successeurs ;
// Pour parcourir uniquement les cases valides dans le tableau

void rechercher_successeurs_possibles ()
// Mise à jour des variables globales 'liste_des_successeurs'
// et 'nb_successeurs'
{
    int lv ; int cv ; //case vide
    int l ; int c ;
    nb_successeurs = 0 ;

    // Copie de l'état en analyse (état choisi) dans les 4 cases du tableau des successeurs
    // On ne modifiera ensuite que les cases concernées par les déplacements possibles

    for (int i=0; i<4; i++)
    {
        for (l=0; l<dimension_plateau; l++)
        {
            for (c=0; c<dimension_plateau; c++)
            {
                liste_des_successeurs[i].plateau[l][c] = ensembles[etat_en_cours].plateau[l][c];
            };
        };
    };

    // Recherche de l'emplacement de la case vide dans l'état en cours

    for (int l=0; l<dimension_plateau; l++)
    {
        for (int c=0; c<dimension_plateau; c++)
        {
            if ( ensembles[etat_en_cours].plateau[l][c]==0 )
            {
                lv=l; cv=c;
            };
        };
    };
}
```

```
};  
};  
  
// Recherche des successeurs possibles selon l'emplacement de la case vide  
  
if ( (lv==0) && (cv==0) ) {  
// coin haut gauche  
// droite_gauche ;  
    liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv][cv+1];  
    liste_des_successeurs[nb_successeurs].plateau[lv][cv+1]=0;  
    nb_successeurs = nb_successeurs + 1 ;  
// bas_haut ;  
    liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv+1][cv];  
    liste_des_successeurs[nb_successeurs].plateau[lv+1][cv]=0;  
    nb_successeurs = nb_successeurs + 1 ;  
};  
if ( (lv==0) && (cv==dimension_plateau-1) ) {  
// coin haut droite  
// gauche_droite ;  
    liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv][cv-1];  
    liste_des_successeurs[nb_successeurs].plateau[lv][cv-1]=0;  
    nb_successeurs = nb_successeurs + 1 ;  
// bas_haut ;  
    liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv+1][cv];  
    liste_des_successeurs[nb_successeurs].plateau[lv+1][cv]=0;  
    nb_successeurs = nb_successeurs + 1 ;  
};  
if ( (lv==0) && (cv!=0) && (cv!=dimension_plateau-1) ) {  
// milieu ligne du haut  
// gauche_droite ;  
    liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv][cv-1];  
    liste_des_successeurs[nb_successeurs].plateau[lv][cv-1]=0;  
    nb_successeurs = nb_successeurs + 1 ;  
// droite_gauche ;  
    liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv][cv+1];  
    liste_des_successeurs[nb_successeurs].plateau[lv][cv+1]=0;  
    nb_successeurs = nb_successeurs + 1 ;  
// bas_haut ;  
    liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv+1][cv];  
    liste_des_successeurs[nb_successeurs].plateau[lv+1][cv]=0;  
    nb_successeurs = nb_successeurs + 1 ;  
};  
if ( (lv==dimension_plateau-1) && (cv==0) ) {  
// coin bas gauche  
// haut_bas ;  
    liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv-1][cv];
```

```
        liste_des_successeurs[nb_successeurs].plateau[lv-1][cv]=0;
        nb_successeurs = nb_successeurs + 1 ;
    // droite_gauche ;
        liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv][cv+1];
        liste_des_successeurs[nb_successeurs].plateau[lv][cv+1]=0;
        nb_successeurs = nb_successeurs + 1 ;
};
if ( (lv==dimension_plateau-1) && (cv==dimension_plateau-1) ) {
// coin bas droite
    // haut_bas ;
        liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv-1][cv];
        liste_des_successeurs[nb_successeurs].plateau[lv-1][cv]=0;
        nb_successeurs = nb_successeurs + 1 ;
    // gauche_droite ;
        liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv][cv-1];
        liste_des_successeurs[nb_successeurs].plateau[lv][cv-1]=0;
        nb_successeurs = nb_successeurs + 1 ;
};
if ( (lv==dimension_plateau-1) && (cv!=0) && (cv!=dimension_plateau-1) ) {
// milieu ligne bas
    // haut_bas ;
        liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv-1][cv];
        liste_des_successeurs[nb_successeurs].plateau[lv-1][cv]=0;
        nb_successeurs = nb_successeurs + 1 ;
    // gauche_droite ;
        liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv][cv-1];
        liste_des_successeurs[nb_successeurs].plateau[lv][cv-1]=0;
        nb_successeurs = nb_successeurs + 1 ;
    // droite_gauche ;
        liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv][cv+1];
        liste_des_successeurs[nb_successeurs].plateau[lv][cv+1]=0;
        nb_successeurs = nb_successeurs + 1 ;
};
if ( (lv!=0) && (lv!=dimension_plateau-1) && (cv!=0) && (cv!=dimension_plateau-1) ) {
// milieu plateau
    // haut_bas ;
        liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv-1][cv];
        liste_des_successeurs[nb_successeurs].plateau[lv-1][cv]=0;
        nb_successeurs = nb_successeurs + 1 ;
    // gauche_droite ;
        liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv][cv-1];
        liste_des_successeurs[nb_successeurs].plateau[lv][cv-1]=0;
        nb_successeurs = nb_successeurs + 1 ;
    // droite_gauche ;
        liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv][cv+1];
        liste_des_successeurs[nb_successeurs].plateau[lv][cv+1]=0;
```

```

        nb_successeurs = nb_successeurs + 1 ;
    // bas_haut ;
    liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv+1][cv];
    liste_des_successeurs[nb_successeurs].plateau[lv+1][cv]=0;
    nb_successeurs = nb_successeurs + 1 ;
};
if ( (lv!=0) && (lv!=dimension_plateau-1) && (cv==0) ) {
// colonne gauche ligne milieu
    // haut_bas ;
    liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv-1][cv];
    liste_des_successeurs[nb_successeurs].plateau[lv-1][cv]=0;
    nb_successeurs = nb_successeurs + 1 ;
    // droite_gauche ;
    liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv][cv+1];
    liste_des_successeurs[nb_successeurs].plateau[lv][cv+1]=0;
    nb_successeurs = nb_successeurs + 1 ;
    // bas_haut ;
    liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv+1][cv];
    liste_des_successeurs[nb_successeurs].plateau[lv+1][cv]=0;
    nb_successeurs = nb_successeurs + 1 ;
};
if ( (lv!=0) && (lv!=dimension_plateau-1) && (cv==dimension_plateau-1) ) {
// colonne droite ligne milieu
    // haut_bas ;
    liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv-1][cv];
    liste_des_successeurs[nb_successeurs].plateau[lv-1][cv]=0;
    nb_successeurs = nb_successeurs + 1 ;
    // gauche_droite ;
    liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv][cv-1];
    liste_des_successeurs[nb_successeurs].plateau[lv][cv-1]=0;
    nb_successeurs = nb_successeurs + 1 ;
    // bas_haut ;
    liste_des_successeurs[nb_successeurs].plateau[lv][cv]=ensembles[etat_en_cours].plateau[lv+1][cv];
    liste_des_successeurs[nb_successeurs].plateau[lv+1][cv]=0;
    nb_successeurs = nb_successeurs + 1 ;
};
};

```

