

Projet Résolution de Problèmes

1GI

Problème de missionnaires & cannibales

*Réalisé par :* *Encadrant :*

* AMGHAR Souhail - Pr. Malika ADDOU

Espace d’états :

J’ai représenté les paramètres des cannibales et des missionnaires dans un état de 6 nombres :

Etat (int Dc ,int Dm, int Tc ,int Tm, int Ac ,int Am)

* **Dc**: nombres des cannibales sur la rive de départ. (0≤Dc≤3)
* **Dm** : nombres des missionnaires sur la rive de départ. (0≤Dm≤3)
* **Tc** : nombres des cannibales sur le bateau. (0≤Tc≤2)
* **Tm**: nombres des missionnaires sur le bateau. (0≤Tm≤2)
* **Ac** : nombres des cannibales sur la rive d’arrivée. (0≤Ac≤3)
* **Am** : nombres des missionnaires sur la rive d’arrivée. (0≤Am≤3)

# Etat initial : (3,3,0,0,0,0) ; Etat final : (0,0,0,0,3,3)

Arbre de Recherche :

Les règles :

* Le remplissage : Vider le bateau sur la rive de départ et prendre aux plus deux individus.
* Le vidage : Vide le bateau sur la rive d’arrivée et revenir avec au moins un dans le bateau vers la rive de départ.

Stratégie suivie :

* Stratégie A 🡺 Heuristique « H » donne un cout de valeur {0 ;1} sous la contrainte que le nombre des cannibales soit toujours inférieur à celui des missionnaires dans les deux rives.

Le code source :

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

typedef struct{

int Dc,Dm;

int Tc,Tm;

int Ac,Am;}etat;

typedef struct noeud{

struct noeud\* suivant;

etat Etat;

}Noeud;

/// Prototypes:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void R\_2Dc(etat e);void R\_2Dm(etat e);void R\_1Dc\_1Dm(etat e);

void R\_1Dc(etat e);void R\_1Dm(etat e);

void R\_1Ac(etat e);void R\_1Am(etat e);

void R\_2Ac(etat e);void R\_2Am(etat e);void R\_1Ac\_1Am(etat e);

Noeud\* NoeudsDejaTraites;

Noeud\* listeNoeuds;

int n;

int vide()

{

if(listeNoeuds==NULL) return 1;

return 0;

}

int vide1()

{

if(NoeudsDejaTraites==NULL) return 1;

return 0;

}

void Inserer(etat e,...)

{

Noeud\* tmp;

tmp=(Noeud\*)malloc(sizeof(Noeud));

tmp->Etat = e;

printf("||==>Dc=%d ||==>Dm=%d ||==>Tc=%d ||==>Tm=%d ||==>Ac=%d ||==>Am=%d\n",tmp->Etat.Dc,tmp->Etat.Dm,tmp->Etat.Tc,tmp->Etat.Tm,tmp->Etat.Ac,tmp->Etat.Am); printf("\n");

if (vide()) {tmp->suivant = NULL;}

else tmp->suivant = listeNoeuds;

listeNoeuds = tmp;

//printf("||==>%d ||==>%d ||==>%d ||==>%d ||==>%d ||==>%d\n",tmp->Etat.Dc,tmp->Etat.Dm,tmp->Etat.Tc,tmp->Etat.Tm,tmp->Etat.Ac,tmp->Etat.Am); printf("\n");

}

void Inserer1(etat e,...)

{

Noeud\* tmp;

tmp=(Noeud\*)malloc(sizeof(Noeud));

tmp->Etat = e;

//printf("||==>%d ||==>%d ||==>%d ||==>%d ||==>%d ||==>%d\n",tmp->Etat.Dc,tmp->Etat.Dm,tmp->Etat.Tc,tmp->Etat.Tm,tmp->Etat.Ac,tmp->Etat.Am); printf("\n");

if (vide1()) {tmp->suivant = NULL;}

else tmp->suivant = NoeudsDejaTraites;

NoeudsDejaTraites =tmp;

//printf("||==>%d ||==>%d ||==>%d ||==>%d ||==>%d ||==>%d\n",tmp->Etat.Dc,tmp->Etat.Dm,tmp->Etat.Tc,tmp->Etat.Tm,tmp->Etat.Ac,tmp->Etat.Am); printf("\n");

}

etat Extraire()

{

etat tmp;

tmp=listeNoeuds->Etat;

if (vide()) EXIT\_FAILURE;

listeNoeuds=listeNoeuds->suivant;

return tmp;

}

int Appartient(etat e,...)

{

Noeud\* tmp;

tmp=(Noeud\*)malloc(sizeof(Noeud));

tmp=NoeudsDejaTraites;

while (tmp != NULL)

{if( e.Dm==tmp->Etat.Dm && e.Dc==tmp->Etat.Dc && e.Tm==tmp->Etat.Tm && e.Tc==tmp->Etat.Tc && e.Am==tmp->Etat.Am && e.Ac==tmp->Etat.Ac ) return 1;

tmp=tmp->suivant;

}

return 0;

}

int etatSolution(etat etatCourant)

{

if (etatCourant.Ac==3 && etatCourant.Am==3) return 1;

return 0;

}

/// LES REGLES A SUIVRE :\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/// remplir le bateau:

void R\_1Dc(etat e)

{

e.Dc=e.Dc+e.Tc;e.Dm=e.Dm+e.Tm;e.Tc=0;e.Tm=0;

if (e.Dc>=1) {e.Tc=1; e.Dc=e.Dc-1; Inserer(e,listeNoeuds);

R\_1Ac(e);R\_1Am(e);

R\_2Ac(e);R\_2Am(e);R\_1Ac\_1Am(e);}

}

void R\_1Dm(etat e)

{

e.Dc=e.Dc+e.Tc;e.Dm=e.Dm+e.Tm;e.Tc=0;e.Tm=0;

if (e.Dm>=1) {e.Tm=1 ; e.Dm=e.Dm-1; Inserer(e,listeNoeuds);

R\_1Ac(e);R\_1Am(e);

R\_2Ac(e);R\_2Am(e);R\_1Ac\_1Am(e);}

}

void R\_2Dc(etat e)

{

e.Dc=e.Dc+e.Tc;e.Dm=e.Dm+e.Tm;e.Tc=0;e.Tm=0;

if (e.Dc>=2) {e.Tc=2; e.Dc=e.Dc-2; Inserer(e,listeNoeuds);

R\_1Ac(e);R\_1Am(e);

R\_2Ac(e);R\_2Am(e);R\_1Ac\_1Am(e);}

}

void R\_2Dm(etat e)

{

e.Dc=e.Dc+e.Tc;e.Dm=e.Dm+e.Tm;e.Tc=0;e.Tm=0;

if (e.Dm>=2) {e.Tm=2 ; e.Dm=e.Dm-2; Inserer(e,listeNoeuds);

R\_1Ac(e);R\_1Am(e);

R\_2Ac(e);R\_2Am(e);R\_1Ac\_1Am(e);}

}

void R\_1Dc\_1Dm(etat e)

{

e.Dc=e.Dc+e.Tc;e.Dm=e.Dm+e.Tm;e.Tc=0;e.Tm=0;

if (e.Dm!=0 && e.Dc!=0) {e.Tm=1 ;e.Tc=1 ; e.Dm-=1;e.Dc-=1; Inserer(e,listeNoeuds);

R\_1Ac(e);R\_1Am(e);

R\_2Ac(e);R\_2Am(e);R\_1Ac\_1Am(e);}

}

/// vider le bateau:

void R\_1Ac(etat e)

{

e.Ac=e.Ac+e.Tc;e.Am=e.Am+e.Tm;e.Tc=0;e.Tm=0;

if (e.Ac>=1) {e.Tc=1; e.Ac=e.Ac-1; Inserer(e,listeNoeuds);

R\_2Dc(e);R\_2Dm(e);R\_1Dc\_1Dm(e);

R\_1Dc(e);R\_1Dm(e);}

}

void R\_1Am(etat e)

{

e.Ac=e.Ac+e.Tc;e.Am=e.Am+e.Tm;e.Tc=0;e.Tm=0;

if (e.Am>=1) {e.Tm=1 ; e.Am=e.Am-1; Inserer(e,listeNoeuds);

R\_2Dc(e);R\_2Dm(e);R\_1Dc\_1Dm(e);

R\_1Dc(e);R\_1Dm(e);}

}

void R\_2Ac(etat e)

{

e.Ac=e.Ac+e.Tc;e.Am=e.Am+e.Tm;e.Tc=0;e.Tm=0;

if (e.Ac>=2) {e.Tc=2; e.Ac=e.Ac-2; Inserer(e,listeNoeuds);

R\_2Dc(e);R\_2Dm(e);R\_1Dc\_1Dm(e);

R\_1Dc(e);R\_1Dm(e);}

}

void R\_2Am(etat e)

{

e.Ac=e.Ac+e.Tc;e.Am=e.Am+e.Tm;e.Tc=0;e.Tm=0;

if (e.Am>=2) {e.Tm=2 ; e.Am=e.Am-2; Inserer(e,listeNoeuds);

R\_2Dc(e);R\_2Dm(e);R\_1Dc\_1Dm(e);

R\_1Dc(e);R\_1Dm(e);}

}

void R\_1Ac\_1Am(etat e)

{

e.Ac=e.Ac+e.Tc;e.Am=e.Am+e.Tm;e.Tc=0;e.Tm=0;

if (e.Am>=1 && e.Ac>=1) {e.Tm=1; e.Tc=1; e.Am=e.Am-1; e.Ac=e.Ac-1; Inserer(e,listeNoeuds);

R\_2Dc(e);R\_2Dm(e);R\_1Dc\_1Dm(e);

R\_1Dc(e);R\_1Dm(e);}

}

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void genereSuccesseurs(etat e)

{

R\_2Dc(e);R\_2Dm(e);R\_1Dc\_1Dm(e);

R\_1Dc(e);R\_1Dm(e);

/\*R\_1Ac(e);R\_1Am(e);

R\_2Ac(e);R\_2Am(e);R\_1Ac\_1Am(e);\*/

}

/// \*\*\*\*\*\*\*Heuristique\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int H(etat e)

{

if(e.Dc+e.Tc <= e.Dm+e.Tm && e.Ac+e.Tc <= e.Am+e.Tm ) return 1;

else return 0;

}

void Tri(Noeud\* N)

{

Noeud\* tmp=N;

etat e;

while (tmp!=NULL)

{

e=tmp->Etat;

if (H(e)==0){

Noeud\* tmp1=tmp->suivant;

tmp->Etat=tmp1->Etat;

tmp->suivant=tmp1->suivant;

}

tmp=tmp->suivant;

}

}

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void rechercheLargeur(etat etatInitial,etat etatFinal,void genereSuccesseurs())

{/// ajouter Heuristique <<<<<<

listeNoeuds=NULL;

printf("l'etat initial : ");

Inserer(etatInitial,listeNoeuds);

//printf("||==>%d ||==>%d ||==>%d ||==>%d ||==>%d ||==>%d",listeNoeuds->Etat.Dc,listeNoeuds->Etat.Dm,listeNoeuds->Etat.Tc,listeNoeuds->Etat.Tm,listeNoeuds->Etat.Ac,listeNoeuds->Etat.Am); printf("\n");

NoeudsDejaTraites=NULL;

etat etatCourant;

printf("\nAttendez qu'on traite les etats courants \*~~\*... \n");

while (!vide())

{etatCourant=Extraire();n++;//printf("||==>%d\n",n);

if (etatSolution(etatCourant))

{

printf("Success ! Arret sur l'Etat: (%d,%d,%d,%d,%d,%d)\n",etatCourant.Dc,etatCourant.Dm,etatCourant.Tc,etatCourant.Tm,etatCourant.Ac,etatCourant.Am);

printf("Nombre de noeuds explorer est : %d \n",n);

exit(0);

}

else if (!Appartient(etatCourant,NoeudsDejaTraites))

{

genereSuccesseurs(etatCourant);

Inserer1(etatCourant,NoeudsDejaTraites);

Tri(listeNoeuds);

}

}

printf("\nPAS DE SOLUTION\n");

}

int main()

{

etat etatInitial={3,3,0,0,0,0},etatFinal={0,0,0,0,3,3};

rechercheLargeur(etatInitial,etatFinal,genereSuccesseurs);

return 0;

}

Exécution :

